



Informatica® Big Data Management
10.2.2

ユーザーガイド

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。

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。

Informatica、Informatica ロゴ、Big Data Management、および PowerExchange 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。

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。

目次

序文	13
Informatica のリソース.....	13
Informatica Network.....	13
Informatica ナレッジベース.....	13
Informatica マニュアル.....	13
Informatica 製品可用性マトリックス.....	14
Informatica Velocity.....	14
Informatica Marketplace.....	14
Informatica グローバルカスタマサポート.....	14
 第 1 章 : Informatica Big Data Management の概要	15
Informatica Big Data Management の概要.....	15
例.....	15
Big Data Management コンポーネントアーキテクチャ.....	16
クライアントおよびツール.....	16
アプリケーションサービス.....	16
リポジトリ.....	17
Hadoop の統合.....	18
Databricks の統合.....	19
Big Data Management エンジン.....	19
Blaze エンジン上のランタイム処理.....	20
Spark エンジン上のランタイム処理.....	22
Databricks Spark エンジン上のランタイム処理.....	22
Big Data プロセス.....	23
手順 1。データの収集.....	23
手順 2。データのクレンジング.....	24
手順 3。データの変換.....	24
手順 4。データの処理.....	24
手順 5。ジョブの監視.....	24
データウェアハウス最適化のマッピングの例.....	25
 第 2 章 : マッピング	27
マッピングの概要.....	27
マッピングランタイムプロパティ.....	28
検証環境.....	28
実行環境.....	30
複数のマッピングのランタイムプロパティの更新.....	33
Hadoop 環境での Sqoop マッピング.....	34
Sqoop マッピングレベル引数.....	34
Sqoop マッピングの差分データ抽出.....	37

マッピングの Sqoop プロパティの設定.	38
マッピングでの Sqoop 引数のパラメータの設定.	38
非ネイティブ環境のマッピングのルールとガイドライン.	39
Blaze エンジンでのマッピングのルールとガイドライン.	39
Spark エンジンでのマッピングのルールとガイドライン.	39
Databricks Spark エンジンでのマッピングのルールとガイドライン.	40
非ネイティブ環境でマッピングを実行するワークフロー.	40
非ネイティブ環境で実行するマッピングの設定.	41
マッピング実行プラン.	41
Blaze エンジンの実行プランの詳細.	42
Spark エンジンの実行プランの詳細.	43
Databricks Spark エンジンの実行の詳細.	44
実行プランの表示.	44
Hadoop 環境向けの最適化.	45
Blaze エンジン高可用性.	46
一時ステー징テーブルでのデータ圧縮の有効化.	46
Hive ターゲットでのパーティションの切り詰め.	47
スケジュール、キュー、およびノードのラベル適用.	48
Big Data ジョブリカバリ.	50
Sqoop パススルーマッピング向けの Spark エンジン最適化.	50
非ネイティブ環境でのマッピングのトラブルシューティング.	51
ネイティブ環境でのマッピング.	52
データプロセッサのマッピング.	52
HDFS マッピング.	52
Hive マッピング.	54
ソーシャルメディアマッピング.	55
ネイティブ環境の最適化.	56
第 3 章: ソース.	59
ソースの概要.	59
PowerExchange アダプタソース.	59
Databricks のソース.	60
ADLS の複合ファイルソース.	61
Azure Blob の複合ファイルソース.	61
Databricks ソースのルールとガイドライン.	61
ファイルソース Hadoop 上.	62
Amazon S3 の複合ファイルソース.	62
ADLS の複合ファイルソース.	63
Azure Blob の複合ファイルソース.	64
MapR-FS の複合ファイルソース.	65
HDFS の複合ファイルソース.	65
フラットファイルソース Hadoop 上.	65
Hadoop のリレーショナルソース.	66

Hive ソース Hadoop 上.	67
PreSQL コマンドおよび PostSQL コマンド.	67
Blaze エンジン上の Hive ソースのルールとガイドライン.	68
Hadoop の Sqoop ソース.	70
Sqoop を使用した Vertica ソースからのデータ読み取り.	70
Sqoop ソースのルールとガイドライン.	71
Sqoop クエリのルールとガイドライン.	71
第 4 章: ターゲット.	72
ターゲットの概要.	72
PowerExchange アダプターターゲット.	72
Databricks のターゲット.	73
ADLS の複合ファイルターゲット.	74
Azure Blob の複合ファイルターゲット.	74
Databricks ターゲットのルールとガイドライン.	74
Hadoop でのファイルターゲット.	75
Amazon S3 の複合ファイルターゲット.	76
ADLS の複合ファイルターゲット.	76
Azure Blob の複合ファイルターゲット.	77
MapR-FS の複合ファイルターゲット.	78
HDFS の複合ファイルターゲット.	78
フラットファイルターゲット Hadoop 上.	79
Hadoop でのメッセージターゲット.	79
Hadoop のリレーショナルターゲット.	80
Hadoop での Hive ターゲット.	80
PreSQL コマンドおよび PostSQL コマンド.	81
Hive ターゲットの切り詰め.	82
アップデートストラテジトランスフォーメーションを使用した Hive ターゲットの更新.	82
Blaze エンジン上の Hive ターゲットのルールとガイドライン.	83
Hadoop の Sqoop ターゲット.	84
Sqoop ターゲットのルールとガイドライン.	85
第 5 章: トランスフォーメーション.	86
トランスフォーメーションの概要.	86
非ネイティブ環境でのアドレスバリデータトランスフォーメーション.	88
Blaze エンジンでのアドレスバリデータトランスフォーメーション.	88
Spark エンジンでのアドレスバリデータトランスフォーメーション.	89
非ネイティブ環境でのアグリゲータトランスフォーメーション.	89
Blaze エンジンでのアグリゲータトランスフォーメーション.	89
Spark エンジンでのアグリゲータトランスフォーメーション.	90
Databricks Spark エンジンでのアグリゲータトランスフォーメーション.	90
非ネイティブ環境での大文字小文字変換トランスフォーメーション.	91
非ネイティブ環境での分類子トランスフォーメーション.	91

非ネイティブ環境での比較トランスフォーメーション.....	91
非ネイティブ環境での統合トランスフォーメーション.....	92
Blaze エンジンでの統合トランスフォーメーション.....	92
Spark エンジンでの統合トランスフォーメーション.....	92
非ネイティブ環境でのデータマスキングトランスフォーメーション.....	92
Blaze エンジンでのデータマスキングトランスフォーメーション.....	92
Spark エンジンでのデータマスキングトランスフォーメーション.....	93
非ネイティブ環境でのデータプロセッサトランスフォーメーション.....	94
Blaze エンジンでのデータプロセッサトランスフォーメーション.....	94
非ネイティブ環境でのディシジョントランスフォーメーション.....	95
非ネイティブ環境での式トランスフォーメーション.....	95
Blaze エンジンでの式トランスフォーメーション.....	95
Spark エンジンでの式トランスフォーメーション.....	95
Databricks Spark エンジンでの式トランスフォーメーション.....	95
非ネイティブ環境でのフィルタトランスフォーメーション.....	96
Blaze エンジンでのフィルタトランスフォーメーション.....	96
非ネイティブ環境での Java トランスフォーメーション.....	96
Blaze エンジンでの Java トランスフォーメーション.....	96
Spark エンジンでの Java トランスフォーメーション.....	97
非ネイティブ環境でのジョイナトランスフォーメーション.....	98
Blaze エンジンでのジョイナトランスフォーメーション.....	98
Spark エンジンでのジョイナトランスフォーメーション.....	99
Databricks Spark エンジンでのジョイナトランスフォーメーション.....	99
非ネイティブ環境でのキージェネレータトランスフォーメーション.....	99
非ネイティブ環境でのラベラトランスフォーメーション.....	99
非ネイティブ環境でのルックアップトランスフォーメーション.....	100
Blaze エンジンでのルックアップトランスフォーメーション.....	100
Spark エンジンでのルックアップトランスフォーメーション.....	100
Databricks Spark エンジンでのルックアップトランスフォーメーション.....	101
非ネイティブ環境での一致トランスフォーメーション.....	101
Blaze エンジンでの一致トランスフォーメーション.....	101
Spark エンジンでの一致トランスフォーメーション.....	101
非ネイティブ環境でのマージトランスフォーメーション.....	102
非ネイティブ環境でのノーマライザトランスフォーメーション.....	102
非ネイティブ環境でのパーサトランスフォーメーション.....	102
非ネイティブ環境での Python トランスフォーメーション.....	103
Spark エンジンでの Python トランスフォーメーション.....	103
非ネイティブ環境でのランクトランスフォーメーション.....	103
Blaze エンジンでのランクトランスフォーメーション.....	103
Spark エンジンでのランクトランスフォーメーション.....	104
Databricks Spark エンジンでのランクトランスフォーメーション.....	104
非ネイティブ環境でのルータートランスフォーメーション.....	104

シーケンスジェネレータトランスフォーメーション非ネイティブ環境で.	105
Blaze エンジンでのシーケンスジェネレータトランスフォーメーション.	105
Spark エンジンでのシーケンスジェネレータトランスフォーメーション.	105
非ネイティブ環境でのソータートランスフォーメーション.	105
Blaze エンジンでのソータートランスフォーメーション.	106
Spark エンジンでのソータートランスフォーメーション.	106
Databricks Spark エンジンでのソータートランスフォーメーション.	107
非ネイティブ環境での標準化トランスフォーメーション.	108
非ネイティブ環境での共有体トランスフォーメーション.	108
非ネイティブ環境でのアップデートストラテジトランスフォーメーション.	108
Blaze エンジンでのアップデートストラテジトランスフォーメーション.	108
Spark エンジンでのアップデートストラテジトランスフォーメーション.	110
非ネイティブ環境での加重平均トランスフォーメーション.	111

第 6 章: クラスタワークフロー. 112

クラスタワークフローの概要.	112
クラスタワークフローのコンポーネント.	113
クラスタワークフロープロセス.	114
クラスタの作成タスクのプロパティ.	115
Azure HDInsight の詳細プロパティ.	116
Amazon EMR の詳細プロパティ.	117
Blaze エンジンの詳細プロパティ.	120
Hive メタストアデータベースの詳細プロパティ.	121
Databricks の詳細プロパティ.	121
マッピングタスクのプロパティ.	123
クラスタの削除タスクの追加.	123
ワークフローのデプロイと実行.	124
Azure HDInsight クラスタワークフロージョブの監視.	124

第 7 章: プロファイル. 125

プロファイルの概要.	125
ネイティブ環境.	125
Hadoop 環境.	126
Sqoop データソースのカラムプロファイル.	126
Informatica Developer での単一のデータオブジェクトプロファイルの作成.	127
Informatica Developer でのエンタープライズ検出プロファイルの作成.	128
Informatica Analyst でのカラムプロファイルの作成.	129
Informatica Analyst でのエンタープライズ検出プロファイルの作成.	130
Informatica Analyst でのスコアカードの作成.	131
プロファイルの監視.	132
プロファイリング機能のサポート.	133
トラブルシューティング.	134

第 8 章 : 監視	135
監視の概要	135
Hadoop 環境のログ	135
YARN Web ユーザーインターフェース	136
監視 URL へのアクセス	137
Administrator ツールでの Hadoop 環境ログの表示	137
マッピングの監視	138
Blaze エンジンの監視	141
Blaze ジョブ監視アプリケーション	143
Blaze サマリレポート	144
Blaze エンジンのログ	147
Blaze ログの表示	148
オーケストレータのサンセット時間	148
Blaze 監視のトラブルシューティング	148
Spark エンジン監視	149
Hive タスクの表示	153
Spark エンジンのログ	153
Spark ログの表示	153
Spark エンジン監視のトラブルシューティング	154
第 9 章 : 階層データ処理	155
階層データ処理の概要	155
マッピングの開発方法階層データの処理	156
復号データ型	158
array データ型	159
Map データ型	161
Struct データ型	161
複合データ型のルールとガイドライン	162
複合ポート	163
トランスフォーメーション内の複合ポート	164
複合ポートのルールとガイドライン	164
複合ポートの作成	165
複合データ型定義	165
ネストされたデータ型定義	167
複合データ型定義のルールとガイドライン	167
複合データ型定義の作成	167
複合データ型定義のインポート	168
型設定	169
配列ポートのタイプ設定の変更	170
マップポートのタイプ設定の変更	172
構造ポートの型設定の指定	173
複合演算子	174

添字演算子を使用した配列要素の抽出.	175
ドット演算子を使用した構造要素の抽出.	175
複合関数.	176
階層データプレビュー.	177
階層データプレビュープロセス.	177
データプレビューインタフェーステーブル.	178
階層データのプレビュー.	180
階層データプレビューのルールおよびガイドライン.	180
第 10 章 : 階層データ処理設定.	181
階層データの変換.	181
リレーショナルデータまたは階層データの構造データへの変換.	182
構造ポートの作成.	182
リレーショナルデータまたは階層データのネストされた構造データへの変換.	184
ネストされた複合ポートの作成.	185
階層データからの要素の抽出.	192
複合ポートからの要素の抽出.	192
階層データのフラット化.	194
複合ポートのフラット化.	195
第 11 章 : スキーマが変更された階層データの処理.	197
スキーマが変更された階層データの処理の概要.	197
階層データのスキーマの変更を処理する動的マッピングの開発方法.	198
動的複合ポート.	199
動的ポートと動的複合ポート.	199
トランスフォーメーション内の動的複合ポート.	201
動的複合ポートの入力ルール.	201
動的配列の入力ルール.	201
動的マップの入力ルール.	203
動的構造の入力ルール.	203
動的複合ポートのポートセクタ.	204
動的式.	204
例 - 動的構造を構築する動的式.	204
複合演算子.	205
複合関数.	206
動的複合ポートのルールおよびガイドライン.	206
最適化されたマッピング.	206
第 12 章 : インテリジェント構造モデル.	208
インテリジェント構造モデルの概要.	208
インテリジェント構造検出プロセス.	209
ユースケース.	209
マッピングでのインテリジェント構造モデルの使用.	210

インテリジェント構造モデルのルールとガイドライン.	211
作業を開始する前に.	211
インテリジェント構造モデルを使用してデータを処理するためのマッピングの開発および実行方法	212
マッピングの例.	213
Informatica Intelligent Cloud Services アカウントの作成.	214
インテリジェント構造モデルの作成.	215
インテリジェント構造モデルのエクスポート.	215
データ損失のチェック.	215
第 13 章 : ステートフルコンピューティング.	216
ステートフルコンピューティングの概要.	216
ウィンドウ化構成.	217
フレーム.	217
パーティションキーおよびオーダーキー.	218
ウィンドウ化構成のルールとガイドライン.	220
ウィンドウ関数.	220
LEAD.	221
LAG.	221
ウィンドウ関数としての集計関数.	221
ウィンドウ関数のルールとガイドライン.	225
ウィンドウ化例.	225
財務プランの例.	225
GPS ping の例.	227
ウィンドウ関数としての集計関数例.	229
付録 A : 接続.	232
接続.	233
クラウドプロビジョニング設定.	233
AWS クラウドプロビジョニング設定のプロパティ.	234
Azure クラウドプロビジョニング設定のプロパティ.	236
Databricks クラウドプロビジョニング設定のプロパティ.	239
Amazon Redshift 接続のプロパティ.	239
Amazon S3 接続のプロパティ.	241
.	242
.	244
Google Analytics 接続のプロパティ.	245
Google BigQuery 接続のプロパティ.	246
Google Cloud Spanner 接続のプロパティ.	247
Google Cloud Storage 接続のプロパティ.	248
Hadoop 接続プロパティ.	249
Hadoop クラスタプロパティ.	249
共通プロパティ.	251
拒否ディレクトリのプロパティ.	252

Hive プッシュダウン設定.	252
Blaze 設定.	252
Spark 設定.	253
HDFS 接続プロパティ.	254
HBase 接続プロパティ.	256
MapR-DB の HBase 接続のプロパティ.	256
Hive 接続のプロパティ.	257
JDBC 接続のプロパティ.	260
Sqoop の接続レベルの引数.	263
Kafka 接続のプロパティ.	265
Microsoft Azure Blob ストレージ接続のプロパティ.	266
Microsoft Azure Cosmos DB SQL API 接続のプロパティ.	267
Microsoft Azure Data Lake Store 接続プロパティ.	267
Microsoft Azure SQL Data Warehouse 接続プロパティ.	268
Snowflake 接続プロパティ.	269
ソースまたはターゲットにアクセスするための接続の作成.	271
Hadoop 接続の作成.	271
Hadoop 接続プロパティの設定.	272
クラスタ環境変数.	273
クラスタのライブラリパス.	273
共通する詳細プロパティ.	273
Blaze エンジン詳細プロパティ.	274
Spark の詳細プロパティ.	275
付録 B: データ型リファレンス.	279
データ型リファレンスの概要.	279
非ネイティブ環境でのトランスフォーメーションデータ型のサポート.	280
複合ファイルデータ型とトランスフォーメーションデータ型.	281
Avro データ型とトランスフォーメーションデータ型.	281
JSON のデータ型とトランスフォーメーションデータ型.	282
Parquet のデータ型とトランスフォーメーションデータ型.	283
Hive のデータ型とトランスフォーメーションデータ型.	284
Hive の複合データ型.	285
Sqoop のデータ型.	286
Aurora のデータ型.	286
IBM DB2 および DB2 for z/OS のデータ型.	286
Greenplum のデータ型.	287
Microsoft SQL Server データ型.	287
Netezza のデータ型.	288
Oracle のデータ型.	288
Teradata のデータ型.	289
Sqoop 用の TDCH 専用コネクタを備えた Teradata のデータ型.	290
Vertica のデータ型.	290

付録 C: 関数リファレンス	292
非ネイティブ環境での関数サポート.....	292
関数とデータ型の処理.....	294
Spark エンジン処理のルールとガイドライン.....	295
索引	297

序文

『*Informatica Big Data Management(R)*ユーザーガイド』は、ネイティブおよび非ネイティブ環境での、マッピングの設定および実行に関する情報を提供します。

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。

Informatica はマニュアルポータルに加えて、Informatica ナレッジベースでも多くの製品に関するドキュメントを管理しています。マニュアルポータルで製品または製品バージョンに関するマニュアルが見つからない場合は、<https://search.informatica.com> でナレッジベースを検索してください。

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

Informatica Big Data Management の概要

この章では、以下の項目について説明します。

- [Informatica Big Data Management の概要, 15 ページ](#)
- [Big Data Management コンポーネントアーキテクチャ, 16 ページ](#)
- [Big Data Management エンジン, 19 ページ](#)
- [Big Data プロセス, 23 ページ](#)
- [データウェアハウス最適化のマッピングの例, 25 ページ](#)

Informatica Big Data Management の概要

Informatica Big Data Management は、急速に変化する多様なビッグデータを組織が大量に処理し、データに対する洞察を得ることを可能にします。Big Data Management を使用して、外部コードを記述または保守せずに、ビッグデータの統合とトランスフォーメーションを実行します。

Big Data Management を使用して、多様なデータを迅速に収集し、仮想環境にビジネスロジックを構築して、手作業でコードを作成せずにデータに対する洞察を得ます。次のような状況で、ビッグデータプロジェクトの実装を検討します。

- 処理対象となるデータの量が 10 テラバイトを超える。
- データの変更をミリ秒単位で分析またはキャプチャする必要がある。
- 構造化されていないテキストからソーシャルメディアデータまで、幅広いデータソースを扱う。

ランタイム処理は、ネイティブ環境または非ネイティブ環境で実行できます。ネイティブ環境は、データ統合サービスがすべてのランタイム処理を実行する Informatica ドメインです。ネイティブのランタイム環境を使用して、10 テラバイトより少ないデータを処理します。非ネイティブ環境は、Hadoop や Databricks などの Informatica ドメイン外部の分散クラスターで、データ統合サービスがランタイム処理をプッシュすることができます。非ネイティブランタイム環境を使用すると、マッピングのパフォーマンスが最適化され、10 テラバイトを超えるデータを処理できます。

例

あなたは投資銀行家であり、株式の人気とリスクを見積もり、顧客の好みに基づいて、各顧客を株式と適合させる必要があります。あなたの CIO は、各株式の人気とリスクの計算処理を自動化し、株式を各顧客と関連付け、全顧客に推奨株式のリストを記載した電子メールを送信することを求めています。

プロジェクトに次の要件を検討します。

- 各株式によって生成されるデータ量は、10 テラバイトを超えます。
- 株式への変更をマイクロ秒で分析する必要があります。
- 株式は Twitter フィードと企業の株式取引 Web サイトに含まれるため、これらのソーシャルメディアソースを分析する必要があります。

要件に基づき、IT 部門と共同で、株式の人気を判断するためのマッピングを作成します。1 つのマッピングは、株式が Twitter フィードに含まれる回数を追跡し、もう 1 つのマッピングは、顧客が企業の株式取引 Web サイトの株式に関して問い合わせた回数を追跡します。

Big Data Management コンポーネントアーキテクチャ

Big Data Management コンポーネントには、Big Data Management がビッグデータプロジェクトに使用するクライアントツール、アプリケーションサービス、リポジトリ、およびサードパーティツールが含まれます。特定の関連コンポーネントは、実行するタスクにより異なります。

クライアントおよびツール

製品ライセンスに応じて、複数の Informatica ツールおよびクライアントを使用すると、ビッグデータプロジェクトを管理できます。

次のツールを使用すると、ビッグデータプロジェクトを管理できます。

Informatica Administrator

Administrator ツールの [監視] タブのプロファイル、マッピング、MDM ビッグデータリレーション管理ジョブのステータスを監視します。Administrator ツールの [監視] タブでは、Monitoring ツールが呼び出されます。また、Administrator ツールで Vibe Data Stream ワークフローを設計することもできます。

Informatica Analyst

ビッグデータソースでのプロファイルの作成と実行、プロジェクトと連携するマッピング仕様の作成、ビッグデータターゲットとデータを統合するビジネスロジックの定義を行います。

Informatica Developer

ビッグデータソースに対するプロファイルの作成と実行、Developer tool の Hadoop クラスタ上でのマッピングとワークフローの実行を行います。

アプリケーションサービス

Big Data Management では、データ処理のために Informatica ドメインでアプリケーションサービスが使用されます。

Administrator ツールを使用して接続の作成、ジョブの監視、Big Data Management で使用するアプリケーションサービスの管理を行います。

Big Data Management では、次のアプリケーションサービスが使用されます。

アナリストサービス

アナリストサービスでは、Analyst ツールが Informatica ドメインで実行されます。アナリストサービスにより、サービスコンポーネントと Analyst ツールにアクセス権があるユーザー間の接続が管理されます。

データ統合サービス

データ統合サービスは、ネイティブ環境でマッピングを処理したり、処理のためにマッピングを非ネイティブ環境の計算クラスタにプッシュしたりできます。また、Developer tool のマッピングまたはワークフローを実行する場合、データ統合サービスはモデルリポジトリからメタデータも取得します。Analyst ツールおよび Developer tool をデータ統合サービスに接続すると、プロファイルジョブを実行し、プロファイリングウェアハウスにプロファイル結果を格納できます。

一括取り込みサービス

一括取り込みサービスは、一括取り込みツールで作成する一括取り込み仕様を管理および検証します。一括取り込みサービスは、データ統合サービスに仕様をデプロイします。仕様が実行されると、一括取り込みサービスで取り込み統計が生成されます。

メタデータアクセスサービス

メタデータアクセスサービスでは、Developer tool を使用して Hadoop クラスタからメタデータをインポートし、プレビューできます。

Hadoop クラスタが Kerberos 認証を使用する場合、メタデータアクセスサービスには、サービスプリンシパル名 (SPN) とキータブ情報に関する情報が含まれます。1 つ以上のメタデータアクセスサービスをノード上で作成できます。ライセンスに基づき、メタデータアクセスサービスの高可用性を実現できます。

HBase、HDFS、Hive、および MapR-DB の接続は、Hadoop クラスタからオブジェクトをインポートするときにメタデータアクセスサービスを使用します。HBase、HDFS、Hive、および MapR-DB の接続を作成する前に、メタデータアクセスサービスを作成および設定します。

Model Repository サービス

Model Repository サービスでは、モデルリポジトリが管理されます。マッピング、マッピング仕様、プロファイル、またはワークフローを実行する場合、モデルリポジトリサービスはモデルリポジトリに接続します。

Rest Operations Hub

Rest Operations Hub サービスは、REST API を介して Informatica 製品の機能を外部クライアントに公開する Informatica ドメインのアプリケーションサービスです。

リポジトリ

Big Data Management は、リポジトリやその他のデータベースを使用して、接続、ソースメタデータ、データドメイン、データプロファイリング、データマスキング、およびデータリネージに関連するデータを格納します。Big Data Management は、Informatica ドメインのアプリケーションサービスを使用して、リポジトリ内のデータにアクセスします。

Big Data Management は次のデータベースを使用します。

モデルリポジトリ

モデルリポジトリには、Developer tool で管理するプロファイル、データドメイン、マッピング、およびワークフローが格納されます。また、モデルリポジトリには、Analyst ツールで管理するプロファイル、データドメイン、およびマッピング仕様も格納されます。

プロファイリングウェアハウス

データ統合サービスはプロファイルを実行し、プロファイル結果をプロファイリングウェアハウスに格納します。

Hadoop の統合

Big Data Management は、さまざまな Hadoop ディストリビューションを実行するクラスタに接続できます。Hadoop はオープンソースのソフトウェアフレームワークであり、複数のマシンのクラスタにわたって大規模なデータセットを分散処理できます。サードパーティのソフトウェアクライアントを使用して Hadoop クラスタを設定および管理する必要がある場合もあります。

Big Data Management は、HDFS、HBase、または Hive などの Hadoop 環境でサポートされているデータソースに接続し、ジョブ処理を Hadoop クラスタにプッシュできます。クラスタ全体のファイルへの高パフォーマンスアクセスを有効にするために、HDFS ソースに接続することができます。また、HDFS に接続するデータウェアハウスである Hive ソースに接続することもできます。

HBase などの NoSQL データベースに接続することもできます。これは、リアルタイムで操作を実行する、Hadoop 上のキーと値のペアを含むデータベースです。データ統合サービスは、マッピングジョブを Spark または Blaze エンジンにプッシュしたり、プロファイルジョブを Hadoop 環境の Blaze エンジンにプッシュしたりできます。

Big Data Management は、一部の Hadoop ディストリビューションの複数のバージョンをサポートしています。デフォルトでは、クラスタ設定ウィザードには最新のサポート対象バージョンが設定されています。

Hadoop のユーティリティ

Big Data Management は、Sqoop などのサードパーティの Hadoop ユーティリティを使用してデータを効率的に処理します。

Sqoop は、MapReduce プログラムを介してリレーショナルデータベースと HDFS の間でデータを処理するための、Hadoop コマンドラインプログラムです。Sqoop を使用してデータをインポートおよびエクスポートできます。Sqoop を使用するときは、Hadoop クラスタのノードにリレーショナルデータベースのクライアントおよびソフトウェアをインストールする必要はありません。

Sqoop を使用するには、JDBC 接続で Sqoop プロパティを設定し、Hadoop 環境でマッピングを実行する必要があります。JDBC 準拠のデータベースに基づくリレーショナ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、および論理データオブジェクトに Sqoop 接続を設定できます。例えば、次のデータベースに対して、Sqoop 接続を設定できます。

- Aurora
- Greenplum
- IBM DB2
- IBM DB2 for z/OS
- Microsoft SQL Server
- Netezza
- Oracle
- Teradata
- Vertica

モデルリポジトリサービスは、JDBC を使用してメタデータをインポートします。データ統合サービスは、Hadoop ランタイム環境でマッピングを実行し、Sqoop にジョブ処理をプッシュします。その後、Sqoop が Hadoop クラスタ内で map-reduce ジョブを作成します。このジョブは、インポートとエクスポートのジョブを同時に実行します。

Sqoop 専用コネクタ

Sqoop からマッピングを実行するときは、次の専用コネクタを使用できます。

OraOop

Sqoop で OraOop を使用すると、Oracle との間でデータの読み書きを行うときのパフォーマンスを最適化できます。OraOop は、ネイティブプロトコルを使用して Oracle データベースに接続する Oracle 用の特殊な Sqoop プラグインです。

Spark エンジンで Sqoop マッピングを実行するときに、OraOop を設定できます。

Teradata Connector for Hadoop (TDCH) の Sqoop 専用コネクタ

次の TDCH の Sqoop 専用コネクタを使用して、Teradata との間でデータの読み書きを行うことができます。

- Cloudera Connector Powered by Teradata
- Hortonworks Connector for Teradata (Teradata Connector for Hadoop 提供)
- MapR Connector for Teradata

これらのコネクタは、Cloudera、Hortonworks、および MapR が Teradata に提供している Sqoop 専用のプラグインです。ネイティブプロトコルを使用して、Teradata データベースに接続します。

Informatica は、Blaze エンジンと Spark エンジンで、Cloudera Connector Powered by Teradata と Hortonworks Connector for Teradata をサポートしています。Blaze エンジンでの Sqoop マッピングを実行するときは、これらのコネクタを設定する必要があります。Sqoop マッピングを Spark エンジンで実行すると、デフォルトではデータ統合サービスによってこれらのコネクタが呼び出されます。

Informatica は、Spark エンジンで、MapR Connector for Teradata をサポートしています。Sqoop マッピングを Spark エンジンで実行すると、デフォルトではデータ統合サービスによってコネクタが呼び出されます。

注: Sqoop でのネイティブ Teradata マッピングの実行の詳細については、『*Informatica PowerExchange for Teradata Parallel Transporter API ユーザーガイド*』を参照してください。

Databricks の統合

Big Data Management は、マッピングを Azure Databricks 環境にプッシュすることができます。Azure Databricks は、Microsoft Azure クラウドサービス用に最適化された分析クラウドプラットフォームです。オープンソースの Apache Spark クラスタのテクノロジーと機能が組み込まれています。

データ統合サービスは、Informatica ドメインと Databricks 環境の統合に必要なバイナリを自動的にインストールします。統合には Informatica 接続オブジェクトおよびクラスタ設定が必須です。クラスタ設定とは、Databricks クラスタからインポートする設定パラメータを含むドメインオブジェクトです。続いてクラスタ設定を接続に関連付けて Databricks 環境にアクセスします。

Big Data Management エンジン

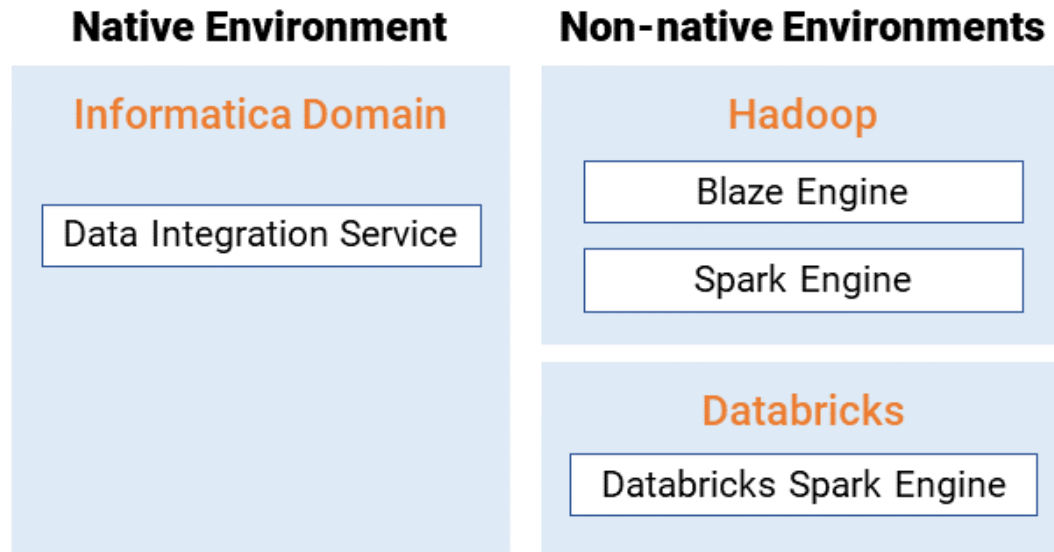
ビッグデータマッピングを実行する場合は、ネイティブ環境または Hadoop や Databricks などの非ネイティブ環境でマッピングを実行できます。マッピングを検証するときは、1 つのエンジンまたはすべてのエンジンに対して検証できます。Developer tool は、各エンジンの検証メッセージを返します。

ネイティブ環境でマッピングを実行する場合は、Informatica ドメインのデータ統合サービスがマッピングを実行します。非ネイティブ環境でマッピングを実行する場合は、データ統合サービスがランタイム処理を非ネイティブ環境の計算クラスタにプッシュします。

非ネイティブ環境でマッピングを実行する場合は、データ統合サービスが独自のルールベースの方法を使用して、マッピングを実行する最適なエンジンを決定します。このルールベースの方法では、マッピングソースと

マッピングロジックが評価されてエンジンが決定されます。データ統合サービスは、マッピングロジックをエンジンが処理できるコードに変換し、そのコードをエンジンに転送します。

次の図は、処理環境とそれらの環境のランタイムエンジンを示しています。



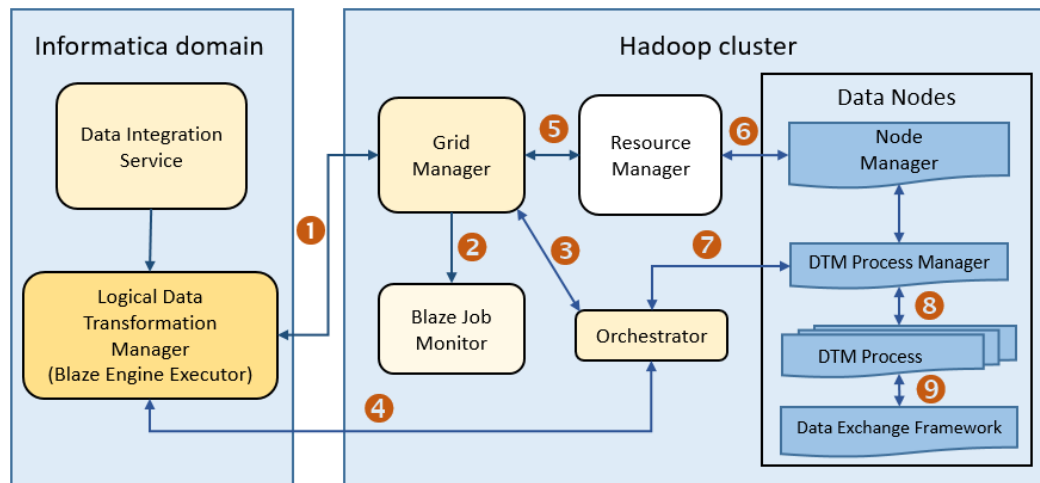
Blaze エンジン上のランタイム処理

Informatica Blaze エンジンでマッピングを実行するには、データ統合サービスが Blaze エンジン実行プログラムにジョブを送信します。Blaze エンジン実行プログラムは、Hadoop クラスタ上のデータ統合サービスと Blaze エンジンコンポーネント間の通信を有効にするソフトウェアコンポーネントです。

次の Blaze エンジンコンポーネントが Hadoop クラスタ上に表示されます。

- グリッドマネージャ。一括処理のタスクを管理します。
- オークストレータ。クラスタ上でデータ処理タスクをスケジュールして同時に処理します。
- Blaze ジョブ監視。クラスタ上で Blaze エンジンジョブを監視します。
- DTM プロセスマネージャ。DTM プロセスを管理します。
- DTM プロセス。DTM インスタンスの実行が開始されたオペレーティングシステムプロセス。
- データ交換フレームワーク。クラスタノード上のデータを処理するさまざまなプロセス間でデータをシャッフルします。

次の図に、Hadoop クラスタが Blaze エンジン実行プログラムから送信されたジョブを処理する方法について示します。



データ統合サービスが Blaze エンジン実行プログラムにジョブを送信する場合、次のイベントが発生します。

1. Blaze エンジン実行プログラムはグリッドマネージャと通信し、Hadoop クラスタの Blaze エンジンコンポーネントを初期化して、利用可能なオーケストレータのグリッドマネージャに対してクエリを実行します。
2. グリッドマネージャは Blaze ジョブ監視を開始します。
3. グリッドマネージャはオーケストレータを開始し、オーケストレータ情報を LDTM に返信します。
4. LDTM はオーケストレータと通信します。
5. グリッドマネージャはリソースマネージャと通信して、オーケストレータに利用可能なリソースがあるかを確認します。
6. リソースマネージャは、ノードマネージャを介してデータノード上のリソース割り当てを処理します。
7. オーケストレータは、DTM プロセスマネージャを介して DTM プロセスにタスクを送信します。
8. DTM プロセスマネージャは DTM プロセスと継続的に通信します。
9. DTM プロセスはデータ交換フレームワークと継続的に通信し、クラスタノードで実行される処理ユニット全体のデータの送受信を行います。

アプリケーションタイムラインサーバー

Hadoop アプリケーションタイムラインサーバーは、完了したアプリケーションプロセスの基本情報を収集します。また、タイムラインサーバーは、完了済みおよび実行中の YARN アプリケーションの情報も提供します。

デフォルトでは、グリッドマネージャは、Yarn 設定でアプリケーションタイムラインサーバーを開始します。

Blaze は、Blaze ジョブ監視ステータスを格納するためにアプリケーションタイムラインサーバーを使用します。デフォルトでタイムラインサーバーが有効になっていない Hadoop ディストリビューションでは、グリッドマネージャはアプリケーションタイムラインサーバープロセスを現在のノードで開始しようとします。

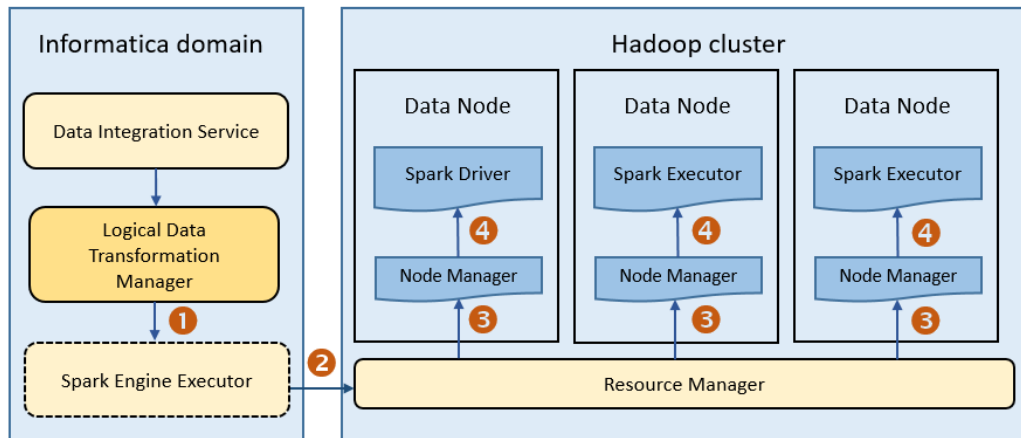
保護された Kerberos クラスタでアプリケーションタイムラインサーバーを有効にしている場合、グリッドマネージャはアプリケーションタイムラインサーバープロセスを HTTP モードで開始しようとします。

Spark エンジン上のランタイム処理

データ統合サービスは、Hadoop クラスタで Spark エンジンを使用して、モデルリポジトリマッピングを実行できます。

Spark エンジンでマッピングを実行するために、データ統合サービスはマッピングアプリケーションを Spark 実行プログラムに送信します。Spark 実行プログラムは、ジョブを実行するために Hadoop クラスタに送信します。

次の図は、Hadoop クラスタが Spark 実行プログラムから送信されたジョブをどのように処理するかを示しています。



データ統合サービスが Spark エンジンでマッピングを実行すると、次のイベントが発生します。

1. 論理 Data Transformation マネージャがマッピングを Scala プログラムに変換し、アプリケーションとしてパッケージ化して、Spark 実行プログラムに送信します。
2. Spark 実行プログラムがこのアプリケーションを Hadoop クラスタ内のリソースマネージャに送信し、アプリケーションを実行するためのリソースを要求します。

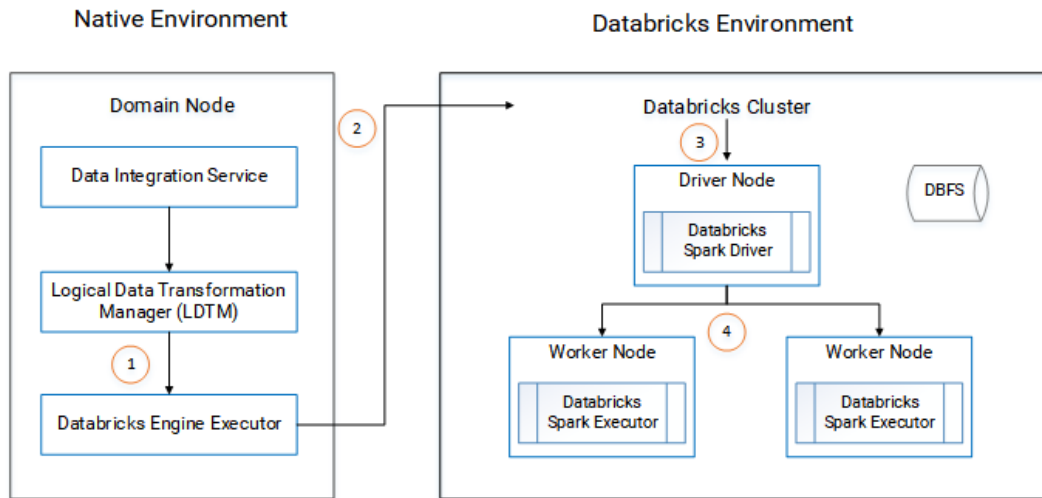
注: HDInsight クラスタでマッピングを実行すると、Spark 実行プログラムが spark-submit スクリプトを起動します。このスクリプトが、アプリケーションを実行するためのリソースを要求します。

3. リソースマネージャが、リソースを提供できるノードマネージャを識別し、ジョブをデータノードに割り当てます。
4. Spark アプリケーションが実行されるデータノードで、ドライバプロセスと実行プログラムプロセスが起動されます。

Databricks Spark エンジン上のランタイム処理

Databricks Spark エンジンでジョブを実行する場合、データ統合サービスが処理を Databricks クラスタにプッシュし、Databricks Spark エンジンがジョブを実行します。

次の図は、Informatica 環境と Databricks 環境のコンポーネントを示しています。



1. 論理 Data Transformation マネージャがマッピングを Scala プログラムに変換し、アプリケーションとしてパッケージ化して、データ統合サービスマシン上の Databricks エンジン実行プログラムに送信します。
2. Databricks エンジン実行プログラムは、REST API を介して、アプリケーションを Databricks クラスタに送信し、アプリケーションの実行を要求して、ランタイム中のアクセスのためにファイルをステージングします。
3. Databricks クラスタは、ドライバノード上の Databricks Spark ドライバに要求を渡します。
4. Databricks Spark ドライバは、ワーカーノードにある 1 つ以上の Databricks Spark 実行プログラムにジョブを分散します。
5. 実行プログラムはジョブを実行し、ワークスペースの Databricks File System (DBFS) にランタイムデータをステージングします。

Big Data プロセス

ビッグデータプロジェクトの一環として、データを各種データソースから収集します。データのプロファイリング、クレンジング、照合を実行できます。データのビジネスロジックを構築し、変換したデータをデータウェアハウスにプッシュします。次に、データのビューでビジネスインテリジェンスを実行できます。

ビッグデータプロジェクトの要件に基づき、次の高度なタスクを実行できます。

1. データの収集
2. データのクレンジング
3. データの変換
4. データの処理
5. ジョブの監視

手順 1。データの収集

データの収集元とする必要があるデータソースを識別します。

Big Data Management には、データ型、データ量、およびデータ待ち時間に基づいて Hadoop 内外のデータにアクセスするためのいくつかの方法が提供されています。

PowerExchange アダプタを使用して、複数のビッグデータソースに接続できます。データのステージングを行う必要なく、データを複数のソースシステムから HDFS に移動するためのバッチロードをスケジュール設定できます。リレーショナルシステムまたはメインフレームシステムから、HDFS または Hive ウェアハウスに変更されたデータを移動できます。リアルタイムのデータフィードを実現するために、データをメッセージキューから HDFS に移動できます。

以下の種類のデータを収集することができます。

- トランザクショナル
- 対話型
- ログファイル
- センサーデバイス
- ドキュメントおよびファイル
- 業界の形式

手順 2。データのクレンジング

データのプロファイリング、クリーニング、および一致を実行して、データをクレンジングします。データに対するデータリネージュを表示できます。

データプロファイリングを実行し、見つからない値や記述的な統計情報を表示して、データに含まれる外れ値や異常値を識別できます。値やパターンの頻度を参照して、データ内の不整合や予期しないパターンを分離できます。不整合のデータにドリルダウンして、データセット全体に渡って結果を表示できます。

それらのデータドメインや関係の検出を自動化できます。社会保障番号やクレジットカード番号などの機密データを検出して、コンプライアンスのためにデータをマスキングできます。

データの質が要件を満たすようになったら、データからビジネス用語集を作成することもできます。Analyst ツールまたは Developer tool を使用して、データプロファイリングタスクを実行できます。Analyst ツールを使用して、データ検出タスクを実行します。Metadata Manager を使用して、データリネージュタスクを実行します。

手順 3。データの変換

Developer tool でデータを解析するためのビジネスロジックを構築できます。事前に作成されたデータ変換用の Informatica のトランスフォーメーションを使用することで、トランスフォーメーションロジックのコードを手作業で作成する必要がありません。

手順 4。データの処理

ビジネスロジックに基づいて、データを処理するために最適なランタイム環境を判断できます。データが 10 テラバイトより少ない場合は、データをネイティブ環境で処理することを検討します。データが 10 テラバイトを超える場合は、データを計算クラスターで処理することを検討します。

手順 5。ジョブの監視

処理ジョブのステータスを監視します。Monitoring ツールで処理ジョブの監視統計を表示できます。処理ジョブの完了後、データからビジネスインテリジェンスと分析を得ることができます。

データウェアハウス最適化のマッピングの例

Hadoop システムを使用するエンタープライズデータウェアハウスを最適化して、ウェアハウス内により多くのデータを低コストで含めることができます。

例えば、24 時間の期間内に変更されたレコードを処理して、顧客のポートフォリオを分析する必要があるとします。Hadoop 上のデータをオフロード処理し、過去 24 時間に挿入、削除、および更新された顧客レコードを見つけ、データウェアハウス内のそれらのレコードを更新できます。ソースファイル内でカラム数が変更されたり、キーが変更された場合でも、これらの変更をキャプチャできます。

変更をキャプチャするために、Developer tool で次のマッピングを作成できます。

Mapping_Day1

最初の 24 時間の期間について、ローカルファイルシステムのフラットファイルから顧客データを読み取り、HDFS ターゲットに書き込むためのマッピングを作成します。

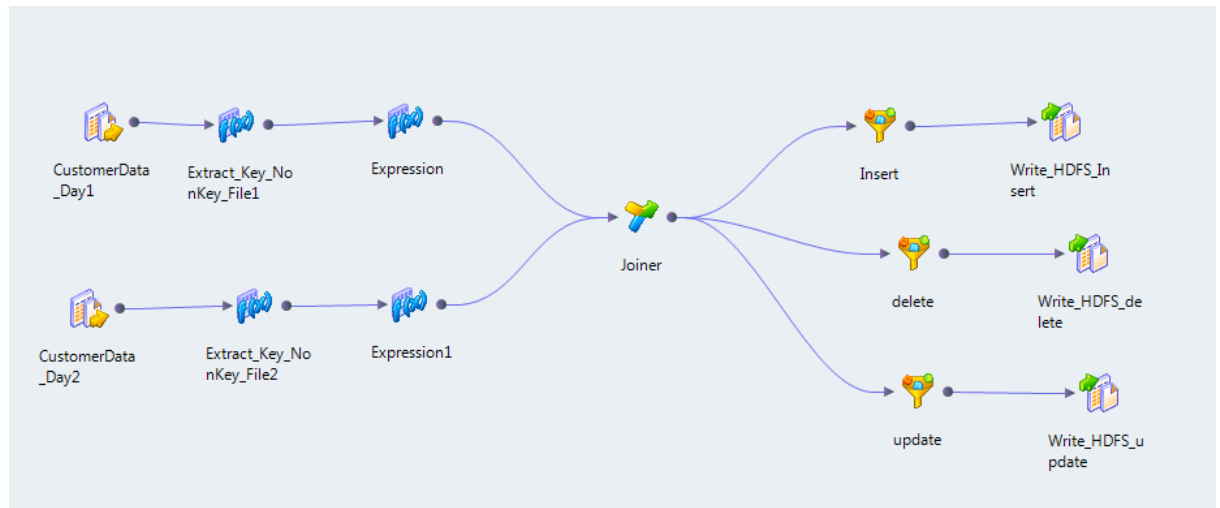
Mapping_Day2

次の 24 時間の期間について、ローカルファイルシステムのフラットファイルから顧客データを読み取り、HDFS ターゲットに書き込むためのマッピングを作成します。

m_CDC_DWHOptimization

変更されたデータをキャプチャするためのマッピングを作成します。このマッピングは、HDFS からデータを読み取り、変更されたデータを識別します。パフォーマンスを向上させるために、マッピングを Hadoop 環境の Hadoop クラスターノードで実行するように設定します。

次の図は、m_CDC_DWHOptimization のマッピングを示します。



マッピングには次のオブジェクトが含まれます。

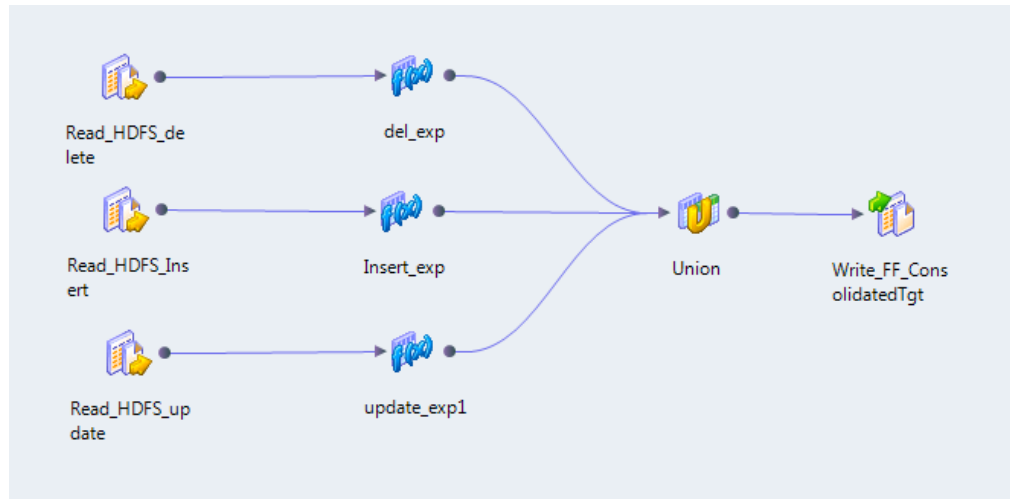
- 読み取りトランスフォーメーション。Mapping_Day1 と Mapping_Day2 のターゲットだった HDFS ファイルからデータを読み取るトランスフォーメーションです。データ統合サービスは、すべてのデータを単一のカラムとして読み取ります。
- 式トランスフォーメーション。データの非キー値からキーを抽出します。式では、INSTR 関数と SUBSTR 関数を使用して、キー値の抽出を実行します。
- ジョイナトランスフォーメーション。式トランスフォーメーションで生成されたキーに基づいて、2 つのソース上で完全な外部結合を実行します。
- フィルタトランスフォーメーション。行を更新、削除、または挿入するべきかどうかに基づいて、ジョイナトランスフォーメーションの出力を使用して行をフィルタします。

- 書き込みトランスフォーメーション。データが挿入、削除、または更新されたかどうかに基づいて、3つの HDFS ファイルにデータを書き込むトランスフォーメーションです。

Consolidated_Mapping

HDFS ファイルのデータを統合し、データをデータウェアハウスにロードするためのマッピングを作成します。

次の図は、Consolidated_Mapping のマッピングを示します。

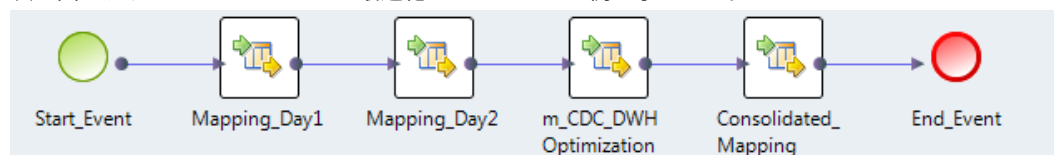


マッピングには次のオブジェクトが含まれます。

- 読み取りトランスフォーメーション。以前のマッピングのターゲットだった HDFS ファイルからデータを読み取るトランスフォーメーションが、このマッピングのソースになります。
- 式トランスフォーメーション。データ行に、削除、更新、または挿入されたタグを追加します。
- 共有体トランスフォーメーション。レコードを結合します。
- 書き込みトランスフォーメーション。ローカルファイルシステムでステージング場所として機能するフラットファイルにデータを書き込むトランスフォーメーションです。

各マッピングを開いて、右クリックしてマッピングを実行できます。すべてのマッピングを順番に実行するには、ワークフローを使用します。

次の図は、データウェアハウスの最適化ワークフローの例を示します。



ワークフローを実行するには、infacmd wfs startWorkflow コマンドを使用します。

第 2 章

マッピング

この章では、以下の項目について説明します。

- [マッピングの概要, 27 ページ](#)
- [マッピングランタイムプロパティ, 28 ページ](#)
- [Hadoop 環境での Sqoop マッピング, 34 ページ](#)
- [非ネイティブ環境のマッピングのルールとガイドライン, 39 ページ](#)
- [非ネイティブ環境でマッピングを実行するワークフロー, 40 ページ](#)
- [非ネイティブ環境で実行するマッピングの設定, 41 ページ](#)
- [マッピング実行プラン, 41 ページ](#)
- [Hadoop 環境向けの最適化, 45 ページ](#)
- [非ネイティブ環境でのマッピングのトラブルシューティング, 51 ページ](#)
- [ネイティブ環境でのマッピング, 52 ページ](#)

マッピングの概要

マッピングのパフォーマンスを最適化し、10 テラバイトより大きいデータを処理するために、Developer tool でランタイム環境を設定します。マッピングは、ネイティブ環境または非ネイティブ環境での実行を選択できます。ネイティブ環境でマッピングを実行する場合は、データ統合サービスがマッピングを処理および実行します。非ネイティブ環境でマッピングを実行する場合は、データ統合サービスは Hadoop や Databricks などの計算クラスタに処理をプッシュします。

スタンドアロンのマッピングや、ワークフローに含まれるマッピングを非ネイティブ環境で実行できます。

Hadoop 環境を選択するとき、エンジンを選択して、マッピングロジックを Hadoop クラスタにプッシュすることもできます。Hadoop 環境では、データ統合サービスがマッピングロジックに基づいて、マッピングロジックを次のいずれかのエンジンにプッシュできます。

- Informatica Blaze エンジン。Hadoop での分散処理用の Informatica 所有のエンジン。
- Spark エンジン。Hadoop クラスタまたは Spark スタンドアロンモードクラスタで実行できる、バッチ処理用の高パフォーマンスエンジン。

Databricks 環境を選択すると、統合サービスはマッピングロジックを Databricks Spark エンジン (Databricks 向けにパッケージ化された Apache Spark エンジン) にプッシュします。

複数のエンジンを選択すると、データ統合サービスが検証中にマッピングの実行に最適なエンジンを決定します。また、データ統合サービスがどのエンジンを使用するかを選択することもできます。エンジンが特定のトランスフォーメーションをサポートするかどうかや、エンジンがデータを返す形式に基づいて、エンジンを選択できます。

非ネイティブ環境でマッピングを実行するときは、環境にアクセスするための接続を設定する必要があります。環境と、マッピングを実行するエンジンのランタイムプロパティを設定できます。

非ネイティブ環境で実行するマッピングの実行プランを表示できます。データ統合サービスがマッピングの実行に選択するエンジンの実行プランを表示します。

Monitoring ツールで、Hive クエリと Hadoop ジョブを監視できます。YARN Web ユーザーインターフェースまたは Blaze ジョブ監視 Web アプリケーションを使用して、Hadoop クラスタのジョブを監視します。

データ統合サービスは、Blaze、Spark、および Databricks Spark エンジンと DTM からのメッセージのログを記録します。

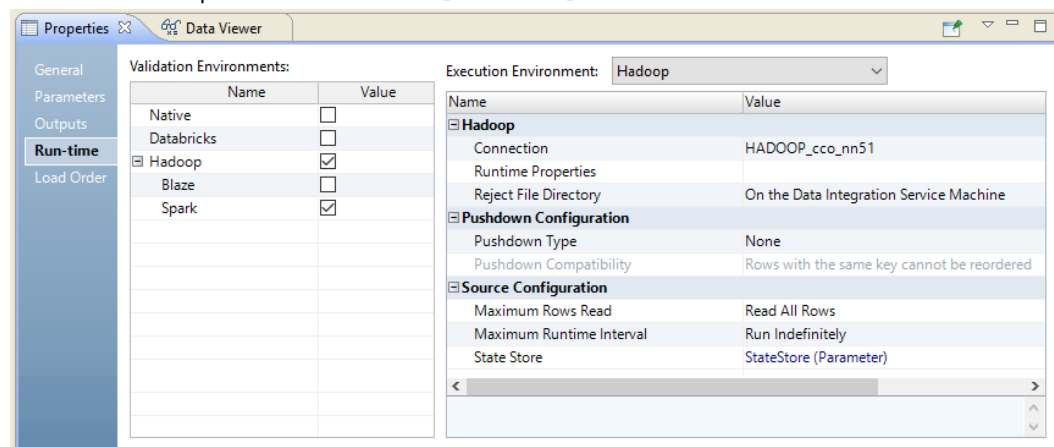
マッピングランタイムプロパティ

マッピングランタイムプロパティはマッピングに選択した環境により異なります。

マッピングプロパティには、**【検証環境】** 領域と **【実行環境】** 領域が含まれます。**【検証環境】** のプロパティは、Developer tool がネイティブ実行環境、非ネイティブ実行環境、または両環境のマッピング定義のいずれを確認するかを示します。ネイティブ環境でマッピングを実行する場合は、データ統合サービスがマッピングを処理します。

非ネイティブ環境でマッピングを行う場合、データ統合サービスはランタイム接続を使用してマッピング実行を計算クラスタにプッシュします。計算クラスタのエンジンがマッピングを処理します。

次の図は、Hadoop 環境のマッピングの **【ランタイム】** プロパティを示します。



検証環境

【検証環境】 のプロパティは、Developer tool がネイティブ実行環境または非ネイティブ実行環境のどちらのマッピング定義を確認するかを示します。

【検証環境】 では、次のプロパティを設定できます。

ネイティブ

デフォルト環境です。データ統合サービスは、ネイティブ環境でマッピングを実行します。

Hadoop

Hadoop 環境でマッピングを実行します。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 Hadoop 接続を通じて Hadoop クラスタにプッシュします。マッピングを処理するエンジンを選択します。Blaze または Spark エンジンを選択できます。

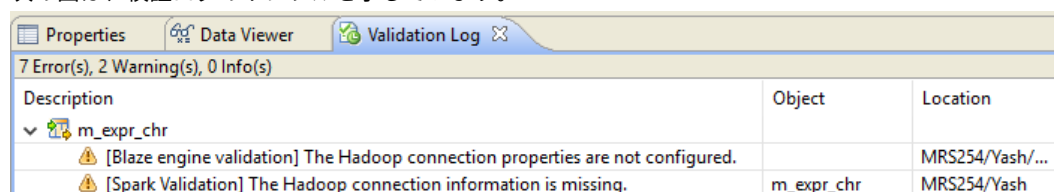
Databricks

Databricks 環境でマッピングを実行します。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 Databricks 接続を通じて Databricks クラスタにプッシュします。Databricks クラスタは、Databricks Spark エンジンでマッピングを処理します。

マッピングパラメータを使用してマッピングの実行環境を示すことができます。実行環境を選択するときに、**[パラメータの割り当て]** をクリックします。文字列パラメータを設定します。デフォルト値を native、hadoop、または spark-databricks に設定します。

マッピングを検証する際に、**[検証環境]** で選択した各エンジンに対して検証が実行されます。各エンジンに特有の検証エラーが、検証ログに記録される場合があります。少なくとも 1 つのエンジンに対してマッピングが有効な場合に、マッピングが有効になります。その他のエンジンのエラーは、検証ログに警告として表示されます。マッピングが複数のエンジンに有効な場合は、実行プランを参照してどのエンジンでジョブを実行するかを決定できます。実行プランは、**[データビューア]** ビューで参照できます。

次の図は、検証ログのサンプルを示しています。



7 Error(s), 2 Warning(s), 0 Info(s)		
Description	Object	Location
▼ m_expr_chr		
⚠ [Blaze engine validation] The Hadoop connection properties are not configured.		MRS254/Yash/...
⚠ [Spark Validation] The Hadoop connection information is missing.	m_expr_chr	MRS254/Yash

実行環境

非ネイティブプロパティ、プッシュダウン設定プロパティ、ソース設定プロパティを【**実行環境**】領域で設定します。

次の表に、Hadoop および Databricks 環境で設定できるプロパティを示します。

名前	説明
接続	Hadoop および Databricks 環境向けの設定。 データ統合サービスがマッピングの実行を計算クラスタにプッシュするために必要な接続情報を定義します。計算クラスタでマッピングを実行するには、非ネイティブ接続を選択します。非ネイティブ接続にユーザー定義のパラメータを割り当てることができます。
ランタイムプロパティ	Hadoop 環境向けの設定。 Hadoop 環境のランタイムプロパティは、データ統合サービス、Hadoop 接続、およびマッピングで構成できます。下位レベルで値を設定することにより、上位レベルで構成されたプロパティをオーバーライドできます。例えば、データ統合サービスのカスタムプロパティでプロパティを構成する場合は、Hadoop 接続またはマッピングでオーバーライドできます。データ統合サービスは、次の優先順位に基づいてプロパティのオーバーライドを処理します。 1. infacmd ms runMapping で -cp オプションを指定して設定された、マッピングのカスタムプロパティ 2. Hadoop 環境のマッピングのランタイムプロパティ 3. ランタイムエンジン用の Hadoop 接続の詳細プロパティ 4. Hadoop 接続の詳細な全般プロパティ、環境変数、およびクラスパス 5. データ統合サービスのカスタムプロパティ
拒否ファイルディレクトリ	Hadoop 環境向けの設定。 Hadoop 環境でマッピングを実行するときの HDFS 上の Hadoop マッピングファイルのディレクトリ。 Blaze エンジンは、フラットファイル、HDFS、および Hive ターゲットの Hadoop 環境に拒否ファイルを書き込むことができます。Spark エンジンは、フラットファイルおよび HDFS ターゲットの Hadoop 環境に拒否ファイルを書き込むことができます。 次のいずれかのオプションを選択します。 - Hadoop クラスタ上。拒否ファイルは、Hadoop 接続で設定された拒否ディレクトリに移動します。ディレクトリが設定されていない場合、マッピングは失敗します。 - Hadoop 接続に従います。拒否ファイルは、Hadoop 接続プロパティで拒否ディレクトリが有効になっているかどうかに基づいて移動します。拒否ディレクトリが有効になっている場合、拒否ファイルは Hadoop 接続で設定されている拒否ディレクトリに移動します。そうでない場合は、データ統合サービスは、RejectDir システムパラメータに基づいて拒否ファイルを格納します。

次のプッシュダウン設定プロパティを設定できます。

名前	説明
プッシュ ダウン タイプ	Hadoop 環境向けの設定。 次のいずれかのオプションを選択します。 - 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、トランスフォーメーションロジックのソースデータベースへ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 ダウン互 換性	Hadoop 環境向けの設定。 完全なプッシュダウンの最適化を選択していて、マッピングにアップデートストラテジトランスフォーメーションが含まれている場合は、必要に応じて、プッシュダウン互換性オプションを選択するか、プッシュダウン互換性パラメータを割り当てることができます。 次のいずれかのオプションを選択します。 - 複数の行に同じキーがない。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トランスフォーメーションは、同じキーを持たない複数の行を受け取ります。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ターゲットにプッシュできます。 - 同じキーの複数の行を並べ替えることができる。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は、並べ替えが可能な同じキーの複数の行を受け取ります。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非ネイティブ環境にプッシュできます。 - 同じキーの複数の行を並べ替えることができない。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は、並べ替えが不可能な同じキーの複数の行を受け取ります。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非ネイティブ環境にプッシュできません。

Hadoop および Databricks 環境では、次のソースプロパティを設定できます。

名前	説明
読み取られる最大行数	将来の使用のために予約済み。
最大ランタイム間隔	将来の使用のために予約済み。
状態ストア	将来の使用のために予約済み。

Spark エンジンの JSON レコードの解析

マッピングのランタイムプロパティで、Spark エンジンがマッピング内の JSON ソースから読み取る際に、破損したレコードや複数行のレコードをどのように解析するかを設定できます。

次の Spark ランタイムプロパティを設定します。

`infaspark.json.parser.mode`

破損した JSON レコードの処理方法をパーサーに指定します。次のいずれかのモードに値を設定できます。

- DROPMALFORMED。パーサーは破損したすべてのレコードを無視します。デフォルトのモードです。
- PERMISSIVE。パーサーは、破損したレコードの標準ではないフィールドを NULL として受け取ります。
- FAILFAST。パーサーが破損したレコードを検出し、Spark アプリケーションがダウンした場合、パーサーは例外を生成します。

infaspark.json.parser.multiLine

JSON ファイルの複数行のレコードをパーサーが読み取れるかどうかを指定します。この値は、true または false に設定できます。デフォルトは false です。Spark バージョン 2.2.x 以上を使用する非ネイティブのディストリビューションにのみ適用します。

拒否ファイルディレクトリ

拒否ファイルは、データ統合サービスマシンまたは Hadoop クラスタに書き込むことができます。または、Hadoop 接続設定に従うこともできます。Blaze エンジンでは、フラットファイルターゲット、HDFS ターゲット、および Hive ターゲットの拒否ファイルを Hadoop 環境に書き込むことができます。Spark エンジンでは、フラットファイルおよび HDFS ターゲットの Hadoop 環境に拒否ファイルを書き込むことができます。

Hadoop 接続に従うようにマッピングランタイムプロパティを設定すると、アクティブな Hadoop 接続の拒否ファイルを Hadoop に書き込むように選択しているかどうかに基づいて、この設定が施されたすべてのマッピングの拒否ファイルが移動されます。拒否ファイルディレクトリを変更するためにマッピングランタイムプロパティを手動で変更する必要はありません。

例えば、拒否ファイルが現在データ統合サービスマシンに移動されていて、それらのファイルを Hadoop 接続に設定されたディレクトリに移動する場合は、拒否ファイルを Hadoop に書き込むように Hadoop 接続プロパティを編集します。Hadoop 接続に従うように設定されているすべてのマッピングの拒否ファイルが、設定したディレクトリに移動されます。

また、複数の Hadoop 接続を交互に使用するように接続がパラメータ化されている場合にも、Hadoop 接続に従うように選択することもできます。例えば、拒否ファイルをデータ統合サービスマシンに移動するように設定されている Hadoop 接続と、拒否ファイルを Hadoop 接続に設定されたディレクトリに移動するように設定されている別の Hadoop 接続を交互に使用するようにパラメータを指定できます。Hadoop 接続に従うように選択すると、接続パラメータのアクティブな Hadoop 接続に応じて、拒否ファイルが移動されます。

実行時に拒否ファイルディレクトリを作成することはできません。拒否ファイルを保存するためには、選択したオプションに応じて、設定された拒否ファイルディレクトリがデータ統合サービスマシンまたは Hadoop 環境に存在する必要があります。

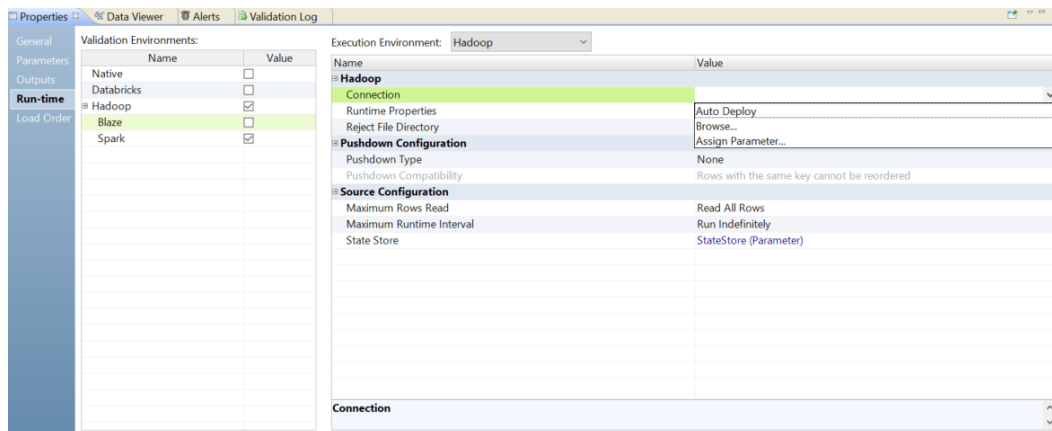
マッピング実行のための計算クラスタの変更

マッピングを実行する計算クラスタを変更できます。

マッピングを別の計算クラスタタイプや別のバージョンで実行するために、計算クラスタを変更する場合があります。

マッピングを実行する計算クラスタを変更するには、マッピングのランタイムプロパティでクラスタが表示される接続をクリックします。**【参照】** をクリックし、マッピングを実行するクラスタのクラスタ設定を使用する非ネイティブ接続を選択します。

次の図は、非ネイティブ接続に対する**【参照】**の選択肢を示します。



マッピングで使用する接続を変更したら、変更を有効にするためにデータ統合サービスを再起動する必要があります。

複数のマッピングのランタイムプロパティの更新

検証環境を有効または無効にしたり、複数のマッピングの実行環境を設定したりできます。Developer tool から実行する複数のマッピングや、データ統合サービスにデプロイする複数のマッピングを更新できます。コマンドラインインターフェースを使用して、これらの更新を実行します。

次の表では、マッピングのランタイムプロパティを更新するコマンドについて説明します。

コマンド	説明
dis disableMappingValidationEnvironment	データ統合サービスにデプロイされたマッピングのマッピング検証環境を無効にします。
mrs disableMappingValidationEnvironment	Developer tool で実行したマッピングのマッピング検証環境を無効にします。
dis enableMappingValidationEnvironment	データ統合サービスにデプロイされたマッピングのマッピング検証環境を有効にします。
mrs enableMappingValidationEnvironment	Developer tool で実行したマッピングのマッピング検証環境を有効にします。
dis setMappingExecutionEnvironment	データ統合サービスにデプロイされたマッピングのマッピング実行環境を指定します。
mrs setMappingExecutionEnvironment	Developer tool で実行したマッピングのマッピング実行環境を指定します。

Hadoop 環境での Sqoop マッピング

Sqoop 接続を有効にした JDBC 接続を使用して、Sqoop ソースまたは Sqoop ターゲットをインポートしてマッピングを作成できます。Sqoop マッピングは、Blaze および Spark の各エンジンで実行できます。

Cloudera Connector Powered by Teradata または Hortonworks Connector for Teradata を使用する場合は、Blaze エンジンまたは Spark エンジン上でマッピングを実行できます。MapR Connector for Teradata を使用する場合は、Spark エンジンでマッピングを実行できます。

マッピングで、追加の Sqoop 引数を指定し、Sqoop コネクタを無効化できます。

注: externaljdbcjars ディレクトリからの Sqoop 接続に必要な Type 4 JDBC ドライバ.jar ファイルを追加または削除した場合、変更が有効になるのはデータ統合サービスを再起動してからです。マッピングを Blaze エンジンで実行した場合、変更が有効になるのはデータ統合サービスと Blaze グリッドマネージャを再起動してからです。マッピングを初めて実行する場合には、データ統合サービスと Blaze グリッドマネージャを再起動する必要はありません。データ統合サービスと Blaze グリッドマネージャを再起動する必要があるのは、それに続いてマッピングが実行された場合のみです。

Sqoop マッピングレベル引数

データオブジェクトで Sqoop が使用されている場合、Sqoop マッピング内の対応する**読み取り**トランスフォーマーまたは**書き込み**トランスフォーマーをクリックして、データ処理に Sqoop が使用する必要がある引数を定義できます。データ統合サービスは、マッピングに指定した追加の Sqoop 引数と JDBC 接続に指定した引数をマージして、Sqoop コマンドを構築します。

マッピングに指定する Sqoop 引数は、JDBC 接続に指定した引数より優先されます。ただし、JDBC 接続で Sqoop コネクタを有効にしていない場合にマッピングで Sqoop コネクタが有効になっていると、データ統合サービスは Sqoop を使用してマッピングを実行しません。データ統合サービスは、JDBC を使用してマッピングを実行します。

Sqoop マッピングで次の Sqoop 引数を設定できます。

- m または num-mappers
- split-by
- batch
- infaoptimize
- infaownername
- スキーマ
- verbose

設定できる Sqoop 引数の完全なリストについては、Sqoop のマニュアルを参照してください。

m または num-mappers

m または num-mappers 引数は、データのインポートとエクスポートを並行して行うために Sqoop が使用する必要があるマップタスクの数を定義します。

以下の構文を使用します。

-m <マップタスクの数>

--num-mappers <マップタスクの数>

m 引数または num-mappers 引数を設定した場合、Sqoop が作業単位を分割するときに基準にするカラムを指定するために、split-by 引数も指定する必要があります。

m 引数または num-mappers 引数を使用して、並行処理の度合いを増やします。パフォーマンスを最適にするために、異なる値を試す必要がある場合もあります。

m 引数または num-mappers 引数を設定して Spark エンジンまたは Blaze エンジン上で Sqoop マッピングを実行する場合、Sqoop は、ファイルサイズに基づいてパーティションを動的に作成します。

注: Blaze または Spark エンジンでデータをエクスポートするために num-mappers 引数を設定する場合、Sqoop は引数を無視します。Blaze または Spark エンジンが作成する中間ファイルの数に基づいて、Sqoop はマップタスクを作成します。

split-by

split-by 引数は、Sqoop が作業単位を分割する際に基準とするカラムを定義します。

以下の構文を使用します。

```
--split-by <column_name>
```

split-by 引数を設定して、パフォーマンスを向上させることができます。プライマリキーの範囲の上限と下限の間で値の均等分布がない場合、データの均等分布がある別のカラムを指定して作業単位を分割するように、split-by 引数を設定できます。

split-by カラムを定義しない場合、Sqoop では次の条件に基づいて作業単位が分割されます。

- データオブジェクトに単一プライマリキーが含まれる場合、Sqoop はプライマリキーを split-by カラムとして使用します。
- データオブジェクトにコンポジットプライマリキーが含まれる場合、Sqoop のデフォルトは、split-by 引数を使用せずにコンポジットプライマリキーを処理する Sqoop の動作になります。詳細については、Sqoop のマニュアルを参照してください。
- データオブジェクトにプライマリキーがない場合、m 引数の値と num-mappers 引数は、デフォルトで 1 になります。

split-by 引数のルールとガイドライン

split-by 引数を設定する場合は、次の制限事項を考慮します。

- split-by 引数を設定しており、split-by カラムに NULL 値が含まれる場合、Sqoop は NULL 値が含まれる行をインポートしません。ただし、マッピングは正しく実行され、YARN ログにエラーは記録されません。
- split-by 引数を設定しており、split-by カラムに特殊文字が含まれる場合、Sqoop のインポートプロセスは失敗します。
- split-by 引数は、次のシナリオで必要になります。
 - Cloudera Connector Powered by Teradata または Hortonworks Connector for Teradata を使用し、Teradata テーブルにプライマリキーがない場合。
 - Sqoop ソースからデータをインポートする際に、デフォルトクエリをオーバーライドするカスタムクエリを作成した場合。

batch

batch 引数は、Sqoop がデータを行単位ではなくバッチでエクスポートする必要があることを示します。

以下の構文を使用します。

```
--batch
```

batch 引数は、パフォーマンスを向上する目的で設定できます。

batch 引数は、Sqoop を次のシナリオで設定するときに必要です。

- データを Vertica ターゲットに書き込む。
- データを IBM DB2 for z/OS に書き込む。
- Sqoop 用の TDCH 専用コネクタではなく一般的な JDBC コネクタを使用したマッピングを実行することで、データを Teradata ターゲットに書き込む。一般的な JDBC コネクタは、JDBC 接続の--driver および--connection-manager Sqoop 引数を設定することで起動できます。

infaoptimize

infaoptimize 引数によって、Spark エンジンでの Sqoop パススルーマッピングのパフォーマンスの最適化を無効にするかどうかが定義されます。

Sqoop パススルーマッピングを Spark エンジンで実行すると、データ統合サービスでは次のシナリオでマッピングパフォーマンスを最適化します。

- Sqoop ソースからデータを読み取り、テキスト形式を使用する Hive ターゲットにデータを書き込みます。
- Sqoop ソースからデータを読み取り、Flat、Avro、または Parquet 形式を使用する HDFS ターゲットにデータを書き込みます。

パフォーマンスの最適化を無効にする場合、--infaoptimize 引数を false に設定します。例えば、最適化した Sqoop マッピングの実行後に、データ型の問題が発生したことがわかった場合、パフォーマンスの最適化を無効にできます。

以下の構文を使用します。

```
--infaoptimize false
```

infaownername

infaownername 引数は、Sqoop でデータオブジェクトのオーナー名を使用するかどうかを指定します。

データオブジェクトのオーナー名を使用するように Sqoop を設定するには、infaownername 引数の値を true に設定します。

以下の構文を使用します。

```
--infaownername true
```

注: Microsoft SQL Server データベースのオーナー名を使用するには、-- --schema 引数を設定して、オーナー名を指定します。以下の構文を使用します。

```
-- --schema <owner_name>
```

スキーマ

schema 引数は、Sqoop が Microsoft SQL Server データベースを表すデータオブジェクトのオーナー名を使用する必要があるかどうかを示します。

Microsoft SQL Server ソースデータベースのオーナー名を使用するには、Sqoop マッピングの **【Sqoop import の追加の引数】** フィールドにオーナー名を指定します。以下の構文を使用します。

```
-- --schema <owner_name>
```

Microsoft SQL Server ターゲットデータベースのオーナー名を使用するには、Sqoop ターゲットデータオブジェクトの **【所有者】** フィールドを空白のままにする必要があります。Sqoop マッピングの **【Sqoop export の追加の引数】** フィールドにオーナー名を指定します。以下の構文を使用します。

```
-- --schema <owner_name>
```

verbose

デバッグの目的でログの詳細な情報が必要なときは、`--verbose` 引数を設定します。

以下の構文を使用します。

```
--verbose
```

注: `--verbose` 引数を設定した場合、特に Sqoop マッピングを Spark エンジンで実行した場合に、パフォーマンスが非常に低下することがあります。

Sqoop マッピングの差分データ抽出

Sqoop マッピングを設定すると、ID またはタイムスタンプベースのソースカラムに基づいた差分データ抽出を実行できます。差分データ抽出では、Sqoop は最後のデータ抽出以降変更されたデータのみを抽出します。差分データ抽出により、マッピングのパフォーマンスが向上します。

差分データ抽出を実行するには、Sqoop マッピングの読み取りトランスフォーメーションの **[Sqoop import の追加の引数]** テキストボックスで引数を設定します。

差分データ抽出では、次の引数のすべてを設定する必要があります。

`--infa-incremental-type`

差分データ抽出を ID とタイムスタンプカラムのどちらに基づいて実行するかを指定します。

ソーステーブルに、プライマリキーまたは一意キーカラムとして動作する ID カラムが含まれている場合は、ID カラムに基づく差分データ抽出を設定できます。`--Infa-incremental-type` 引数の値を ID として設定します。Sqoop は、最後に抽出された ID より大きな ID を持つ行を抽出します。

ソーステーブルに、すべての行の最後に更新された時刻を含むタイムスタンプカラムが含まれている場合は、タイムスタンプカラムに基づく差分データ抽出を設定できます。`--Infa-incremental-type` 引数の値を `timestamp` として設定します。Sqoop は、最後に読み取られたタイムスタンプ値または最大タイムスタンプ値より大きなタイムスタンプを持つ行を抽出します。

以下の構文を使用します。

```
--infa-incremental-type <ID or timestamp>
```

`--infa-incremental-key`

Sqoop が差分データ抽出を実行するカラム名を指定します。

以下の構文を使用します。

```
--infa-incremental-key <column_name>
```

`--infa-incremental-value`

Sqoop が差分データ抽出を実行するためにベースライン値として使用するカラム値を指定します。Sqoop は、`--infa-incremental-value` 引数に定義された値より大きな値を持つすべての行を抽出します。

カラム値は二重引用符で囲みます。

以下の構文を使用します。

```
--infa-incremental-value "<column_value>"
```

`--infa-incremental-value-format`

タイムスタンプカラムに基づく差分データ抽出を設定する場合に適用できます。`--infa-incremental-value` 引数で定義されたカラム値のタイムスタンプ形式を指定します。

形式値は二重引用符で囲みます。

以下の構文を使用します。

```
--infa-incremental-value-format "<format>"
```

デフォルトは「MM/dd/yyyy HH:mm:ss.SSSSSSSS」です。

注: 必要なすべての引数を設定していないのに差分データ抽出を実行した場合、Sqoop マッピングは失敗します。引数値の大文字と小文字は区別されません。

例

組織の営業部門は、Salesforce に新規顧客に関する情報を保存しています。特定の日付後に作成された顧客レコードを読み取り、そのデータを SAP に書き込む Sqoop マッピングを作成しようとしています。

この Sqoop マッピングでは、次のように Sqoop import 引数を定義できます。

```
--infa-incremental-type timestamp --infa-incremental-key CreatedDate --infa-incremental-value "10/20/2018 00:00:00.000000000" --infa-incremental-value-format "MM/dd/yyyy HH:mm:ss.SSSSSSSS"
```

データ統合サービスは、10/20/2018 00:00:00.000000000 より大きいタイムスタンプ値を含む CreatedDate カラムを持つすべての顧客レコードを読み取ります。データの読み取り中、Sqoop マッピングは 10/20/2018 00:00:00.000000000 形式を使用します。

マッピングの Sqoop プロパティの設定

マッピングレベルで、追加の Sqoop 引数を指定し、Sqoop コネクタを無効化できます。マッピングレベルで指定する Sqoop 引数は、JDBC 接続に指定した引数より優先されます。

1. Sqoop プロパティを設定するデータオブジェクトが含まれるマッピングを開きます。
2. データオブジェクトと関連付けられている、読み取りまたは書き込みのトランスフォーメーションを選択します。
3. **【詳細】** タブをクリックします。
4. データオブジェクトの Sqoop コネクタを無効化するには、**【Sqoop コネクタを無効にします】** チェックボックスを選択します。
5. 次のいずれかの手順に従います。
 - データオブジェクトに対して追加の Sqoop import の引数を指定するには、**【Sqoop import の追加の引数】** テキストボックスにインポートの引数を入力します。
 - データオブジェクトに対して追加の Sqoop export の引数を指定するには、**【Sqoop export の追加の引数】** テキストボックスにエクスポートの引数を入力します。

データ統合サービスは、マッピングに指定した追加の Sqoop 引数と JDBC 接続に指定した引数をマージして、Sqoop コマンドを構築します。その後、データ統合サービスは、Hadoop ノードで Sqoop を起動します。

マッピングでの Sqoop 引数のパラメータの設定

Sqoop マッピングでは、Sqoop import および export の引数をパラメータ化できます。

1. Sqoop 引数をパラメータ化する Sqoop マッピングを開きます。
2. **【パラメータ】** タブをクリックして、パラメータ名とパラメータ値を定義します。
例えば、パラメータ名を「Param_splitByColumn」として入力します。
3. データオブジェクトと関連付けられている、読み取りまたは書き込みのトランスフォーメーションを選択します。
4. **【詳細】** タブをクリックします。
5. 次のいずれかの手順に従います。

- データオブジェクトの Sqoop import の引数をパラメータ化するには、**[Sqoop import の追加の引数]** テキストボックスをクリックして、**[パラメータ]** タブで定義したパラメータ名を入力します。
- データオブジェクトの Sqoop export の引数をパラメータ化するには、**[Sqoop export の追加の引数]** テキストボックスをクリックして、**[パラメータ]** タブで定義したパラメータ名を入力します。

次の形式を使用して Sqoop import または export の引数のパラメータ名を指定します。

```
--<Sqoop_argument_name> $<parameter_name>
```

例えば、「--split-by \$Param_splitByColumn」と入力します。

非ネイティブ環境のマッピングのルールとガイドライン

非ネイティブ環境でマッピングを実行する際は、異なる処理や構成が適用されます。マッピングは、Blaze、Spark、または Databricks Spark エンジンの非ネイティブ環境で実行できます。ネイティブおよび非ネイティブ環境でマッピングを実行する際には、次のルールとガイドラインを考慮します。

Blaze エンジンでのマッピングのルールとガイドライン

Blaze エンジンでは、次の実行時の違いを考慮します。

- Hadoop 環境では、カラムにデータエラーのあるソースでは、そのカラムの値は NULL になります。ネイティブ環境では、カラムにデータエラーのある行はデータ統合サービスによって処理されません。
- フラットファイルソースから読み取るマッピングをキャンセルすると、フラットファイルデータを HDFS にコピーするファイルコピープロセスは引き続き実行される場合があります。データ統合サービスはこのプロセスを強制終了するコマンドを Hive セッションログに記録し、HDFS にコピーされたすべてのデータをクリーンアップします。必要に応じて、ファイルのコピープロセスを強制終了するコマンドを実行できます。

Spark エンジンでのマッピングのルールとガイドライン

Spark エンジンでは、次の実行時の違いを考慮します。

- マッピングで検証が実行されても実行に失敗する場合は、最適化レベルをなしまたは最小限に設定します。コストベースまたは準結合の最適化方式を使用するように最適化レベルを設定した場合、データ統合サービスは実行時にこれを無視し、デフォルト値を使用します。
- ランタイムエンジンは、すべてのケースで、以前のプロジェクションの最適化方式を使用しません。データ統合サービスが使用されていないポート間のリンクを削除した場合、ランタイムエンジンはポートを再接続する場合があります。
- Hadoop 環境では、カラムにデータエラーのあるソースでは、そのカラムの値は NULL になります。ネイティブ環境では、カラムにデータエラーのある行はデータ統合サービスによって処理されません。
- フラットファイルソースから読み取るマッピングをキャンセルすると、フラットファイルデータを HDFS にコピーするファイルコピープロセスは引き続き実行される場合があります。データ統合サービスはこのプロセスを強制終了するコマンドを Hive セッションログに記録し、HDFS にコピーされたすべてのデータをクリーンアップします。必要に応じて、ファイルのコピープロセスを強制終了するコマンドを実行できます。

Spark エンジンは、マッピングを実行するときに、次の場合に HiveServer2 を使用してクラスタ上のジョブを処理します。

- マッピングが、char 型または varchar 型のフィールド上にバケット化された Hive テーブルのターゲットに書き込む場合。
- マッピングが、Hive トランザクション対応テーブルで読み取りまたは書き込みを行う場合。
- マッピングが、カラムレベルのセキュリティが有効な Hive テーブルで読み取りまたは書き込みを行う場合。
- マッピングが、Hive ターゲットに書き込み、実行時にテーブルを作成または置き換えるように設定されている場合。

Databricks Spark エンジンでのマッピングのルールとガイドライン

Databricks Spark エンジンでは、次の実行時の違いを考慮します。

- マッピングで検証が実行されても実行に失敗する場合は、最適化レベルをなしまたは最小限に設定します。コストベースまたは準結合の最適化方式を使用するように最適化レベルを設定した場合、データ統合サービスは実行時にこれを見落とし、デフォルト値を使用します。
- ランタイムエンジンは、すべてのケースで、以前のプロジェクションの最適化方式を使用しません。データ統合サービスが使用されていないポート間のリンクを削除した場合、ランタイムエンジンはポートを再接続する場合があります。

非ネイティブ環境でマッピングを実行するワークフロー

非ネイティブ環境で実行するように設定したマッピングを、ワークフローのマッピングタスクに追加することができます。ワークフローをデプロイして実行すると、マッピングタスクがマッピングを実行します。

ワークフローの実行中に決定できるようにするために、ワークフローからマッピングを実行するように決定する場合があります。複数のマッピングを順番に、または並行して実行するようにワークフローを設定できます。マッピングタスクのステータスをユーザーに通知する電子メールを送信するようにワークフローを設定できます。

Spark または Databricks Spark エンジンで実行されるワークフローにマッピングを追加するには、マッピング出力も設定する必要があります。モデルリポジトリでマッピング出力を保持し、保持した出力をマッピングタスク入力に割り当てることができます。また、マッピング出力とワークフロー変数をバインドして、ワークフロー内で出力の値を他のタスクに渡すことができます。

注: Spark または Databricks Spark エンジンでは、マッピング出力内で SUM 集計タイプを使用できません。

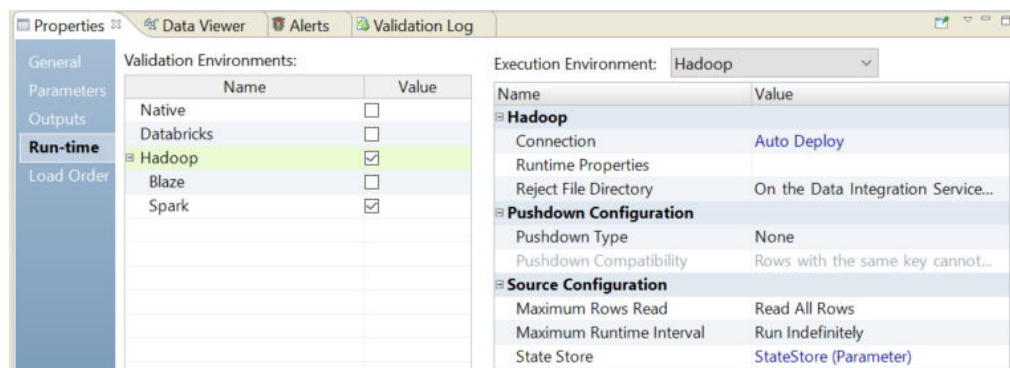
マッピング出力の詳細については、『*Developer ワークフローガイド*』の「マッピングタスク」の章を参照してください。

Blaze エンジンで実行するよう設定されたマッピングをマッピングタスクが実行する場合、マッピングタスクの出力をワークフロー変数に割り当てないでください。Blaze エンジンで実行されるマッピングでは、ターゲット、ソース、エラー行の合計数が提供されません。マッピングタスクに Blaze エンジンで実行されるマッピングが含まれている場合、そのタスク出力にはゼロ (0) の値が含まれています。

非ネイティブ環境で実行するマッピングの設定

非ネイティブ環境で実行するためのマッピングを設定できます。マッピングを設定するには、検証環境と実行環境を選択する必要があります。

1. **【オブジェクトエクスプローラ】** ビューのプロジェクトまたはフォルダからマッピングを選択して、エディタで表示します。
2. **【プロパティ】** ビューで **【ランタイム】** タブを選択します。
3. **【Hadoop】** か **【Databricks】**、または両方を検証環境の値として選択します。
Hadoop 環境を選択すると、Blaze および Spark エンジンがデフォルトで選択されます。使用しないエンジンを無効にします。
4. 実行環境として **【Hadoop】** または **【Databricks】** を選択します。



5. **【接続】** を選択し、値のフィールドのドロップダウンを使用して、接続を参照するか、接続パラメータを作成します。
 - 接続を選択するには、**【参照】** をクリックして接続を選択します。
 - 接続パラメータを作成するには、**【パラメータの割り当て】** をクリックします。
6. 実行環境の残りのプロパティを設定します。
7. エディタ内の何もない領域を右クリックし、**【検証】** をクリックします。
Developer tool がマッピングを検証します。
8. **【検証ログ】** タブで、検証エラーを表示します。
9. **【データビューア】** ビューをクリックします。
10. **【実行プランの表示】** をクリックして、マッピングの実行プランを表示します。

マッピング実行プラン

データ統合サービスは、Blaze、Spark、または Databricks Spark エンジンでマッピングを実行するための実行プランを生成します。データ統合サービスは、マッピングロジックをランタイムエンジンが実行できるコードに変換します。プランは、マッピングの実行前は Developer tool で表示し、実行後は Administrator ツールで表示できます。

データ統合サービスは、次のエンジンで実行するマッピング実行プランを生成します。

Informatica Blaze エンジン

Blaze エンジンの実行プランは、マッピングをセグメントに単純化します。マッピングの開始、マッピングの実行、一時テーブルと一時ファイルのクリーンアップを実行するタスクと、複数のタスクレットやタスクのリカバリストラテジが含まれます。また、計算クラスタで主なマッピングを実行する前に、マッピングごとにグリッドタスクの前と後に実行される準備コマンドも含まれます。グリッドタスクの前のタスクには、HDFS へのデータのコピーなどが含まれます。グリッドタスクの後のタスクには、一時ファイルのクリーンアップや HDFS からのデータのコピーなどが含まれます。

Spark エンジン

Spark 実行プランは、マッピングロジックを実行するランタイム Scala コードを示します。変換エンジンは、マッピングをロジックの内部表現に変換します。内部表現は、Spark API にアクセスする Scala コードにレンダリングされます。実行プランの Scala コードを表示して、ロジックをデバッグできます。

Databricks Spark エンジン

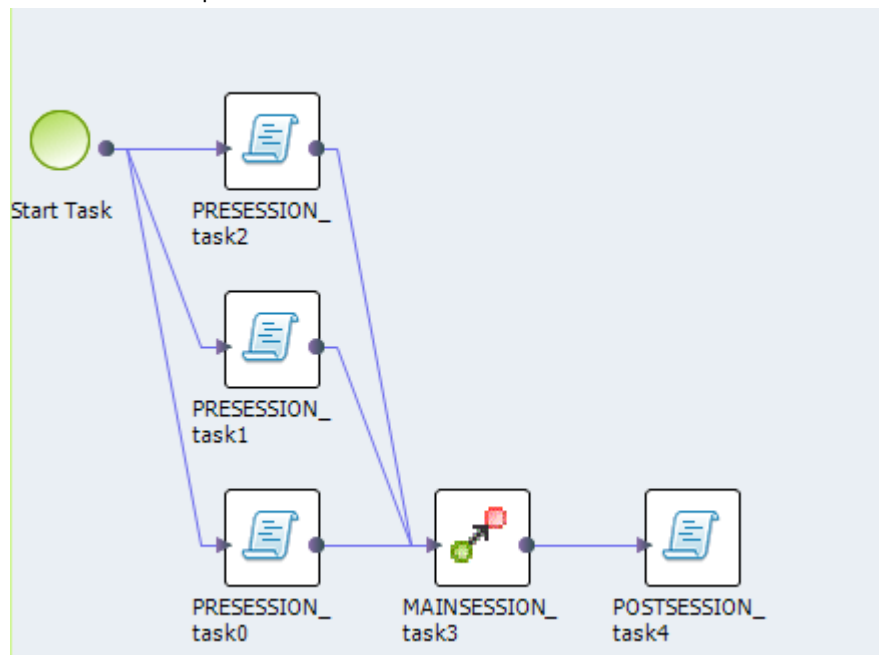
Databricks Spark 実行プランは、マッピングロジックを実行するランタイム Scala コードを示します。変換エンジンは、マッピングをロジックの内部表現に変換します。内部表現は、Spark API にアクセスする Scala コードにレンダリングされます。実行プランの Scala コードを表示して、ロジックをデバッグできます。

Blaze エンジンの実行プランの詳細

Blaze エンジンの実行プランの詳細は、Administrator ツールおよび Developer tool で表示できます。

Developer tool で、Blaze エンジンの実行プランがワークフローとして表示されます。ワークフロー内の各コンポーネントをクリックし、詳細を取得できます。

次の図に、Developer tool での Blaze 実行プランを示します。



Blaze エンジン実行プランワークフローには、次のコンポーネントが含まれます。

- 開始タスク。ワークフロー開始タスク。
- コマンドタスク。ローカルデータの前処理または後処理タスク。

- グリッドマッピング。Blaze エンジンがコンパイルしてノードのクラスタ全体に配布する Informatica のマッピング。
- グリッドタスク。Blaze エンジン実行者がグリッドマネージャに送信した並行処理ジョブ要求。
- グリッドセグメント。グリッドタスクに含まれるグリッドマッピングのセグメント。
- タスクレット。個別の DTM で実行されるグリッドセグメントのパーティション。

Administrator ツールで、Blaze エンジンの実行プランがスクリプトとして表示されます。

次の図に、Blaze 実行スクリプトを示します。

Test - XraKb1DXEeWg		Properties	Blaze Execution Plan	Summary
Script Id	Script			
MAINSESSION_task3	Execution scriptStep [MAINSESSION_task3], type [GridTaskStepImpl]. With "from" step(s): PRESESSION_task0, PRESESSION_task1, PRESESSION_task2. With "to" step(s): POSTSESSION_task4. Grid mapping task has totally [3] substeps: Execution step [submapping-2], type [SegmentStepImpl]. With no "from" step. With "to" step(s): submapping-3. Included instances: Read_IN_OUT[SourceTx], DETarget_Joiner_G1[TargetTx]. Execution step [submapping-1], type [SegmentStepImpl]. With no "from" step. With "to" step(s): submapping-3. Included instances: DETarget_Joiner_G0[TargetTx], Read_IN_OUT1[SourceTx]. Execution step [submapping-3], type [SegmentStepImpl]. With "from" step(s): submapping-1, submapping-2. With no "to" step. Included instances: Write_IN_OUT[TargetTx], DESource_Joiner_G1[SourceTx], Joiner[JoinerTx], DESource_Joiner_G0[SourceTx].			

Administrator ツールで、Blaze エンジンの実行プランには次の詳細があります。

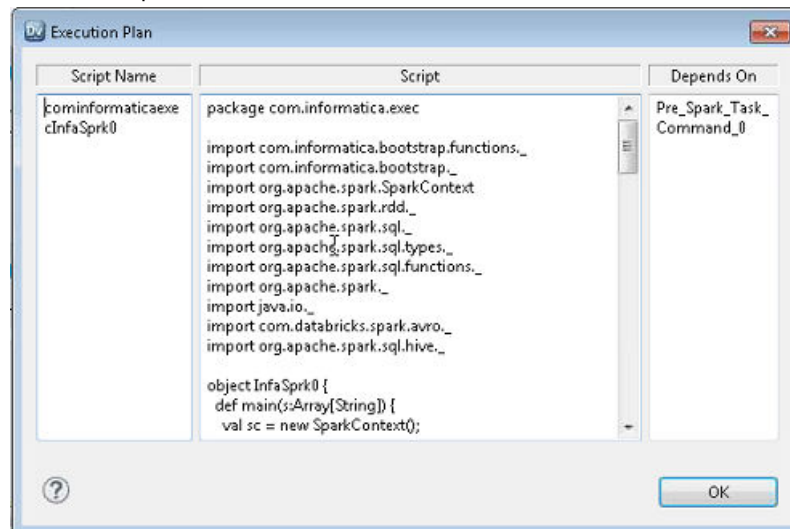
- スクリプト ID。Blaze エンジンスクリプトの固有の ID です。
- スクリプト。データ統合サービスがマッピングロジックに基づいて生成する Blaze エンジンスクリプトです。
- 依存対象。このスクリプトが依存するタスクです。タスクには、その他のスクリプトおよび開始タスクなどのデータ統合サービスタスクが含まれます。

Spark エンジンの実行プランの詳細

Spark エンジンの実行プランの詳細を、Administrator ツールまたは Developer tool から表示できます。

Spark エンジンの実行プランには、Databricks Spark エンジンの Scale コードが表示されます。

次の図は、Spark エンジンで実行するマッピングの実行プランを示します。



Spark エンジンの実行プランの詳細は次のとおりです。

- スクリプト ID。Spark エンジンスクリプトの一意の識別子。

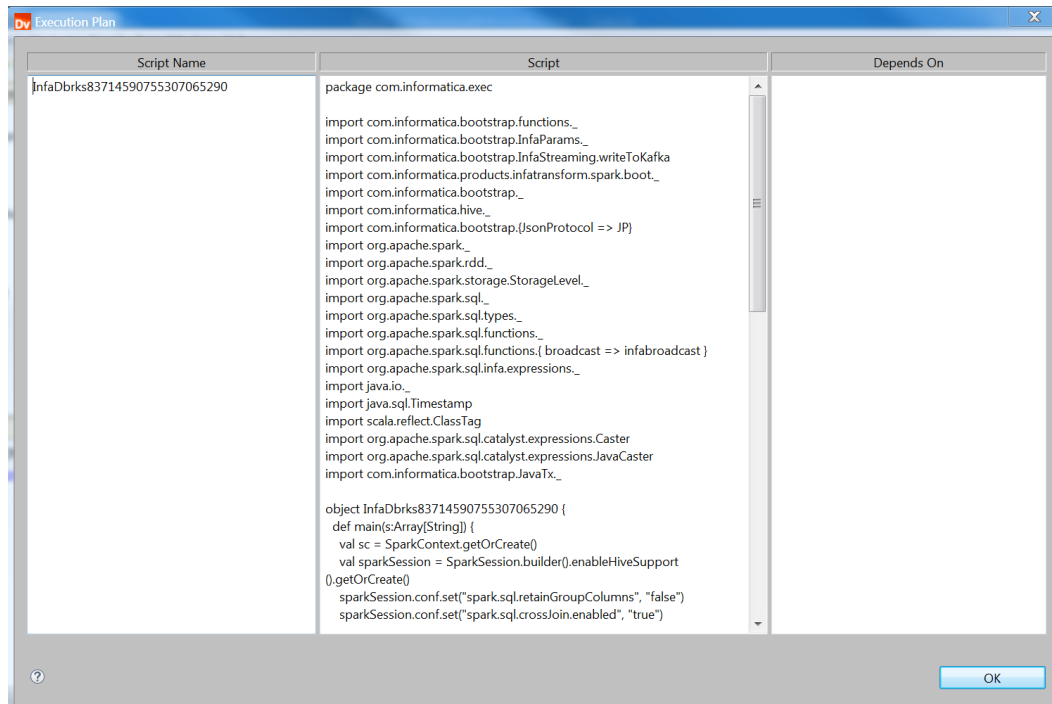
- スクリプト。データ統合サービスがマッピングロジックに基づいて生成したスカラコード。
- 依存対象。このスクリプトが依存するタスクです。タスクには、他のスクリプトやデータ統合サービスタスクが含まれます。

Databricks Spark エンジンの実行の詳細

Databricks Spark エンジンの実行プランの詳細を、Administrator ツールまたは Developer tool から表示できます。

Databricks Spark エンジンの実行プランには、Databricks Spark エンジンで実行される Scale コードが表示されます。

次の図は、Databricks Spark エンジンで実行されるマッピングの実行プランを示します。



Databricks Spark エンジンの実行プランの詳細は次のとおりです。

- スクリプト ID。Databricks Spark エンジンスクリプトの一意的識別子。
- スクリプト。データ統合サービスがマッピングロジックに基づいて生成したスカラコード。
- 依存対象。このスクリプトが依存するタスクです。タスクには、他のスクリプトやデータ統合サービスタスクが含まれます。

実行プランの表示

マッピングのエンジン実行プランを表示できます。Developer tool で実行プランを表示するために、マッピングを実行する必要はありません。

注: 実行プランは、Administrator ツールで表示することもできます。

1. Developer tool で実行プランを表示するには、マッピングの【データビューア】ビューを選択し、【実行プランの表示】をクリックします。
2. 【データビューア】ビューを選択します。

3. **【実行プランの表示】** を選択します。
【データビューア】 ビューに、実行プランの詳細が表示されます。

Hadoop 環境向けの最適化

Hadoop 環境を最適化して、パフォーマンスを向上させます。

Hadoop 環境の最適化は、次の方法で行えます。

可用性の高い Hadoop クラスタの設定

可用性の高い Hadoop クラスタで読み取りと書き込みを行うように、データ統合サービスと Developer tool を設定できます。可用性の高い Hadoop クラスタを設定する手順は、Hadoop ディストリビューションの種類によって異なります。ディストリビューションの設定手順の詳細については、『*Informatica Big Data Management 統合ガイド*』を参照してください。

一時的なステージングテーブルでのデータ圧縮

一時的なステージングテーブルでのデータ圧縮を有効にして、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。

Blaze エンジンでのマッピングの実行

可用性の高い Blaze エンジンで、マッピングを実行します。Blaze エンジンでは、デフォルトで、グリッドタスクとタスクレットの再開とリカバリが有効です。

並列ソートの実行

マッピングでソータトランスフォーメーションを使用すると、データ統合サービスは、Hadoop クラスタにマッピングロジックをプッシュするときに、デフォルトで並列ソートを有効にします。並列ソートを行うと、マッピングパフォーマンスが向上します。

ジョイナトランスフォーメーションのパーティション化

Blaze エンジンのマッピングでジョイナトランスフォーメーションを使用する場合、データ統合サービスは、マップ側の結合の最適化を適用して、マッピングパフォーマンスを改善できます。データ統合サービスは、マスタテーブルが詳細テーブルよりも小さい場合に、マップ側の結合の最適化を適用します。データ統合サービスは、マップ側の結合の最適化を適用するときに、データのシャッフルを行わずにデータをジョイナトランスフォーメーションに移動します。

Hive ターゲットでのパーティションの切り詰め

Hive ターゲットのパーティションを切り詰めて、パフォーマンスを向上させることができます。Hive ターゲットのパーティションを切り詰めるには、Hive ターゲットのパーティションの切り詰めとターゲットテーブルの切り詰めの両方を選択する必要があります。

Hadoop クラスタのリソースの割り当て

スケジューラを使用して、Hadoop クラスタにリソースを割り当てることができます。組織のニーズに応じて、キャパシティスケジューラまたはフェアスケジューラを使用できます。

Hadoop クラスタ上でリソースを共有するための YARN キューの設定

Hadoop クラスタ上のジョブを特定のキューにリダイレクトするように、YARN キューを設定できます。ジョブの割り当て先となったキューが、ジョブの実行に割り当てられるリソースを定義します。

Hadoop クラスタのノードへのラベル付け

Hadoop クラスタのノードにラベルを付けて、特定の特徴を持つパーティションのクラスタを分割できます。

Spark エンジンの Sqoop マッピングの最適化

データ統合サービスは、Spark エンジンで実行される Sqoop パススルーマッピングのパフォーマンスを最適化できます。

ビッグデータジョブリカバリの有効化

ビッグデータジョブリカバリを有効にして、データ統合サービスが Spark エンジンに処理をプッシュするマッピングジョブをリカバリできます。

Blaze エンジン高可用性

Blaze エンジンは、グリッドタスクおよびタスクレットにとって最上のリカバリ戦略を決定する高可能性エンジンです。

Blaze エンジンは、グリッドタスクのサイズに基づいて、次のリカバリ戦略を適用しようとします。

- 高可用性なし。Blaze エンジンはリカバリ戦略を適用しません。
- フル再起動。グリッドタスクを再起動します。

一時ステージングテーブルでのデータ圧縮の有効化

Hadoop 環境でマッピングを実行する際のパフォーマンスを最適化するために、一時ステージングテーブルでデータ圧縮を有効にできます。一時ステージングテーブルでデータ圧縮を有効にすると、マッピングパフォーマンスが改善されることがあります。

一時ステージングテーブルでデータ圧縮を有効にするには、次の手順を実行します。

1. 一時ステージングテーブルでの圧縮を有効にするために、Hadoop クラスタが使用するコーデッククラス名を使用するように Hadoop 接続を設定します。
2. 一時ステージングテーブルで圧縮を有効にするように Hadoop クラスタを設定します。

Hadoop は、次の圧縮コーデッククラス名に対して、次の圧縮ライブラリを提供します。

圧縮ライブラリ	コーデッククラス名	パフォーマンスの推奨事項
Zlib	org.apache.hadoop.io.compress.DefaultCodec	なし
Gzip	org.apache.hadoop.io.compress.GzipCodec	なし
Snappy	org.apache.hadoop.io.compress.SnappyCodec	最適なパフォーマンスで推奨。
Bz2	org.apache.hadoop.io.compress.BZip2Codec	非推奨。パフォーマンスが低下。
LZO	com.hadoop.compression.lzo.LzoCodec	なし

手順 1.Hadoop 接続でのデータ圧縮の有効化

Administrator ツールまたは Developer tool を使用して Hadoop 接続を設定して、一時ステージングテーブルでのデータ圧縮を有効にします。

1. Hadoop 接続のプロパティで、Hadoop クラスタでマッピングを実行するためのプロパティを編集します。
2. [一時テーブル圧縮コーデック] を選択します。
3. 事前定義されたコーデッククラス名を選択するか、カスタムコーデッククラス名を入力します。

- 事前定義されたコーデッククラス名を選択するには、リストから圧縮ライブラリを選択します。
- カスタムコーデッククラス名を入力するには、リストから [カスタム] を選択し、Hadoop クラスタのコーデッククラス名と一致するコーデッククラス名を入力します。

手順 2.Hadoop 環境でのデータ圧縮の有効化

一時ステージングテーブルで圧縮を有効にするには、Hadoop クラスタ上に圧縮コーデックをインストールする必要があります。

圧縮コーデックのインストール方法の詳細については、Apache Hadoop のドキュメントを参照してください。

1. 圧縮コーデッククラス名のネイティブライブラリが、クラスタ上の各ノードにインストールされていることを確認します。
2. 使用する圧縮コーデッククラス名を含めるには、io.compression.codecs プロパティを core-site.xml で更新します。このプロパティの値は、クラスタでサポートされるすべてのコーデッククラス名のカンマ区切りリストです。
3. 使用する圧縮コーデッククラス名の Hadoop ネイティブライブラリが、クラスタ上の各ノードにインストールされていることを確認します。
4. Hadoop クラスタの LD_LIBRARY_PATH 変数に、圧縮コーデックをインストールしたネイティブライブラリまたは Hadoop ネイティブライブラリの両方の場所が含まれていることを確認します。

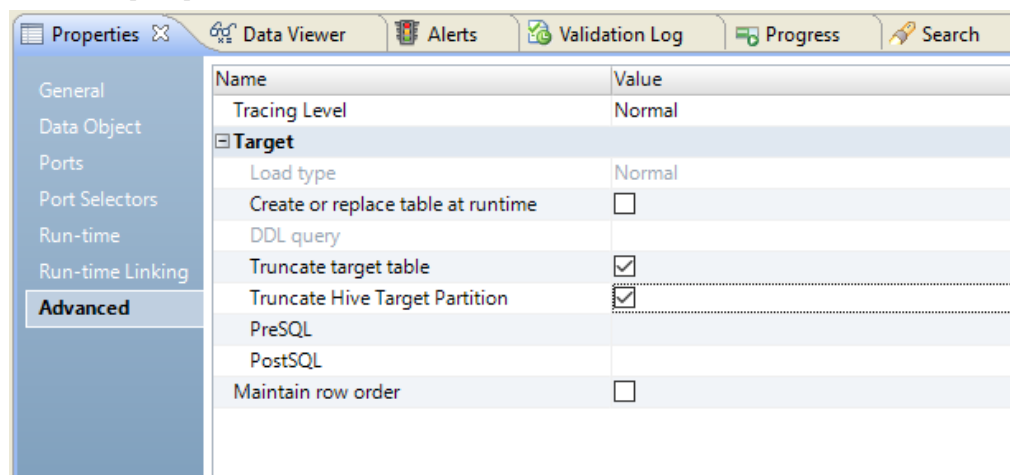
Hive ターゲットでのパーティションの切り詰め

Hive ターゲットでパーティションを切り詰めるには、Developer tool で Hive ターゲット用に作成した、カスタマイズ済みデータオブジェクトの書き込みプロパティを編集する必要があります。

マッピングを実行するために Blaze または Spark のランタイムエンジンを使用した場合、Hive ターゲットのパーティションを切り詰めることができます。

1. カスタマイズされたデータオブジェクトをエディタで開きます。
2. 書き込みプロパティを編集するには、[書き込み] ビューで [入力] トランスフォーメーションを選択し、[詳細] プロパティを選択します。

次の図は、[詳細] プロパティタブを示します。



3. [Hive ターゲットパーティションを切り詰めます] を選択します。
4. [ターゲットテーブルの切り詰め] を選択します。

スケジューラ、キュー、およびノードのラベル適用

YARN スケジューラ、YARN キュー、およびノードラベルを使用して、Hadoop 環境でのマッピング実行時のパフォーマンスを最適化できます。

YARN スケジューラは、組織のリソース共有ポリシーを順守しながら、Hadoop クラスタ上の YARN アプリケーションにリソースを割り当てます。フェアスケジューラまたはキャパシティスケジューラを使用するように YARN を設定できます。フェアスケジューラは、クラスタ上で経時的に実行されているすべてのジョブにわたって、リソースを共有します。キャパシティスケジューラを使用すると、複数の組織が大規模なクラスタを共有し、容量の割り当てに基づいてリソースを分散できます。キャパシティスケジューラは、組織ごとに特定の容量を保証し、利用されていない超過分の容量がある場合はそれを分散します。

YARN キューは、YARN スケジューラの構造を編成し、複数のテナントがクラスタを共有できるようにします。各キューの容量は、キューに送信されたアプリケーションで使用できるクラスタリソースの割合を指定します。Blaze、Spark、および Sqoop のジョブを特定の YARN キューにリダイレクトできます。

ノードラベルを使用すると、YARN キューがクラスタ内の特定のノードで実行されるようになります。ノードラベルを使用すると、特定の特性を持つノードでジョブを実行するなど、クラスタをサブクラスタにパーティション化することができます。例えば、他のノードと比較して、データをより速く処理するノードにラベルを付けられます。ラベルが適用されていないノードは、デフォルトパーティションに所属します。ノードラベルをキャパシティスケジューラのキューに関連付けることができます。

また、Blaze エンジンを設定するためにも、ノードラベルを使用できます。Blaze エンジンを設定するためのノードラベルを使用する場合、Blaze エンジンを実行する Hadoop クラスタ上でノードを指定できます。

スケジューラ設定とノードラベルの有効化

Hadoop 環境でスケジューラ設定とノードラベル適用を有効にするには、クラスタ設定で `yarn-site.xml` のプロパティを更新します。

以下のプロパティを設定します。

`yarn.resourcemanager.scheduler.class`

データ統合サービスがクラスタでのリソースの割り当てに使用する YARN スケジューラを定義します。

```
<property>
  <name>yarn.resourcemanager.scheduler.class</name>
  <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.[Scheduler Type].[Scheduler
Type]Scheduler</value>
</property>
```

以下に例を示します。

```
<property>
  <name>yarn.resourcemanager.scheduler.class</name>
  <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler</value>
</property>
```

`yarn.node-labels.enabled`

ノードのラベル適用を有効にします。

```
<property>
  <name>yarn.node-labels.enabled</name>
  <value>TRUE</value>
</property>
```

`yarn.node-labels.fs-store.root-dir`

ノードラベルを動的に更新する HDFS の場所を示します。

```
<property>
  <name>yarn.node-labels.fs-store.root-dir</name>
  <value>hdfs://[Node name]:[Port]/[Path to store]/[Node labels]</value>
</property>
```


YARN キューの定義

特定のキーにジョブをリダイレクトするように、複数の YARN キューを定義できます。Blaze、Spark、Sqoop のジョブをリダイレクトできます。

Blaze エンジンのジョブを特定のキューにリダイレクトするには、次の Blaze 設定プロパティを Hadoop 接続で設定します。

プロパティ	説明
YARN キュー名	クラスタ上の利用可能なリソースを指定する Blaze エンジンが使用する YARN スケジューラのキュー名。

Spark エンジンのジョブを特定のキューにリダイレクトするには、次の Spark 設定プロパティを Hadoop 接続で設定します。

プロパティ	説明
YARN キュー名	クラスタで使用可能なリソースを指定する Spark エンジンによって使用される YARN スケジューラキュー名。この名前は、大文字と小文字が区別されます。

Sqoop ジョブを特定のキューにリダイレクトするには、次の JDBC 接続プロパティを設定します。

プロパティ	説明
Sqoop 引数	MapReduce ジョブを特定の YARN キューにダイレクトするための Sqoop 接続レベル引数。次の形式を使用します。 -Dmapred.job.queue.name=<YARN queue name> ジョブを特定のキューにダイレクトしない場合、Spark エンジンはデフォルトのキューを使用します。

ノードラベルを使用するための Blaze エンジンの設定

Hadoop 環境でノードラベル適用を有効にする場合、ノードラベルを使用して、Blaze コンポーネントを特定のクラスタノード上で YARN アプリケーションとして実行できます。

Blaze コンポーネントをラベル付けされたクラスタノード上で実行するには、Blaze エンジンを設定するときに、ノードラベルを指定します。

Blaze エンジンのノードラベルを指定するには、次の Hadoop 接続プロパティでノードラベルを一覧表示します。

プロパティ	説明
Blaze YARN ノードラベル	Blaze エンジンが実行される Hadoop クラスタ上のノードを決定するノードラベル。ノードラベルを指定しない場合、Blaze エンジンはデフォルトのパーティションのノードで実行されます。 Hadoop クラスタがノードラベルの論理演算子をサポートしている場合は、ノードラベルの一覧を指定できます。ノードラベルを一覧表示するには、演算子&& (AND)、 (OR)、および! (NOT) を使用します。

注: Blaze エンジンがノードラベルを使用すると、Blaze コンポーネントは、ラベル付けされたノード上で重複する可能性があります。1 つのノードに複数のラベルが含まれており、異なる Hadoop 接続でラベルを指定し

た場合、複数のグリッドマネージャ、オーケストレータ、ジョブ監視インスタンスが同じノード上で実行される可能性があります。

Big Data ジョブリカバリ

管理者は、ビッグデータジョブリカバリを有効にして、Spark エンジンで実行するように設定されたビッグデータジョブを、データ統合サービスノードが予期せず停止した場合にリカバリできます。

ジョブが完了する前にデータ統合サービスノードで障害が発生した場合、データ統合サービスはそのジョブを別のノードに送信し、ノード障害の発生時点からジョブタスクの処理を再開します。リカバリは、ノードの起動時に実行されます。

ビッグデータリカバリを使用するには、Spark エンジンで実行するジョブを設定し、infacmd クライアントからジョブを送信する必要があります。

管理者は、データ統合サービスのプロパティでビッグデータリカバリを設定します。ビッグデータジョブリカバリの詳細については、『*Big Data Management 管理者ガイド*』を参照してください。

Sqoop パススルーマッピング向けの Spark エンジン最適化

Sqoop ソースを含むパススルーマッピングを Spark エンジンで実行すると、データ統合サービスでは次のシナリオでマッピングパフォーマンスを最適化します。

- テキスト形式を使用する Hive ターゲットにデータを書き込みます。
- カスタム DDL クエリによって作成された Hive ターゲットにデータを書き込みます。
- カスタム DDL クエリによってパーティション化されているか、カスタム DDL クエリによってパーティション化およびバケット化されている既存の Hive ターゲットにデータを書き込みます。
- パーティション化およびバケット化されている既存の Hive ターゲットにデータを書き込みます。
- Flat、Avro、または Parquet 形式を使用する HDFS ターゲットにデータを書き込みます。

パフォーマンスの最適化を無効にする場合、JDBC 接続または Sqoop マッピングで `--infaoptimize` 引数を `false` に設定します。例えば、最適化した Sqoop マッピングの実行後に、データ型の問題が発生したことがわかった場合、パフォーマンスの最適化を無効にできます。

以下の構文を使用します。

```
--infaoptimize false
```

Sqoop Spark エンジンの最適化のためのルールとガイドライン

Spark エンジンで Sqoop マッピングを実行する際は、次のルールとガイドラインを考慮します。

- データ統合サービスは、次のシナリオではマッピングのパフォーマンスを最適化しません。
 - マッピング内のソースとターゲットとの間に未接続のポートがある。
 - マッピング内のソースとターゲットのデータ型が一致しない。
 - パーティション化またはバケット化されている既存の Hive ターゲットテーブルにデータを書き込みます。
 - Parquet 形式の HDFS 複合ファイルターゲットにデータを書き込むために、WASB を使用する Azure HDInsight クラスタ上でマッピングを実行する。
- Hive ターゲットにデータを書き込むように Hive 固有の Sqoop 引数を設定した場合、Sqoop はその引数を無視します。
- デフォルトの区切り文字と異なる区切り文字を Hive ターゲットテーブルに設定した場合、Sqoop はその区切り文字を無視します。

非ネイティブ環境でのマッピングのトラブルシューティング

非ネイティブ環境でのマッピングのトラブルシューティングのヒントを考慮します。

Hadoop 環境

別のクラスタ上の Hive ソースまたは Hive ターゲットでマッピングを実行すると、データ統合サービスで次のエラーが発生して、Hadoop へのマッピングのプッシュに失敗する: [exec0_query_6]が原因で、クエリの実行に失敗しました。エラーコード[10]、エラーメッセージ: [FAILED: セマンティック分析のエラー: 行 1:181 テーブルが見つかりません customer_eur]、および SQL 状態[42000]]。

Hadoop 環境でマッピングを実行する場合は、Hive ソース/ターゲットに対して選択した Hive 接続とマッピングが同じ Hive メタストア上にある必要があります。

SQL オーバーライドとマッピングを並列実行すると、マッピングがハングする。

クラスタが異なるエンジン間で共有されているため、使用可能なリソースが不十分です。

キャパシティスケジューラを使用するように YARN を設定し、Blaze、および Spark で異なる YARN スケジューラキューを使用します。

パーティション化された Hive テーブルを作成するためにマッピングを設定した場合に、宛先テーブルがパーティション化されているためパーティションカラムを指定する必要があることを示すエラーが表示されて、マッピングが失敗する。

この問題は、Hive パーティション化テーブルを作成するために設計されたクエリに対する、Informatica の内部要件が原因で発生します。詳細と回避策については、[Knowledge Base article 516266](#) を参照してください。

Databricks 環境

マッピングが次のエラーで失敗する: SEVERE: ID [1857]での実行が状態[INTERNAL_ERROR]、エラーメッセージ[ライブラリインストールが 1800 秒後にタイムアウトしました。ライブラリはまだインストールされていません: jar: "dbfs:/tmp/DATABRICKS/sess6250142538173973565/staticCode.jar"]

この状況は、並列ジョブの実行時に発生することがあります。Databricks は、ジョブを処理するためのリソースがない場合、ジョブを最大 1,800 秒 (30 分) キューイングします。30 分以内にリソースが使用できない場合は、ジョブは失敗します。タイムアウトを回避するには次のアクションを検討してください。

- Databricks クラスタのプリエンブション環境変数を設定して、各ジョブに割り当てられるリソースの量をコントロールします。プリエンブションの詳細については、『*Big Data Management Integration ガイド*』を参照してください。
- エフェメラルクラスタを作成するクラスタワークフローを実行します。ワークフローは、クラスタを作成しジョブを実行してから、そのクラスタを削除するように設定できます。エフェメラルクラスタの詳細については、[第 6 章、「クラスタワークフロー」 \(ページ 112\)](#)を参照してください。

重要: Informatica は Databricks を統合し、標準並列クラスタをサポートします。標準並列クラスタの最大キュー時間は 30 分で、タイムアウトに達するとジョブが失敗します。最大キュー時間は延長できません。プリエンブションのしきい値を設定すると、より多くのジョブを並列実行できますが、割り当てられるリソースの割合が小さくなり、ジョブの実行に時間がかかる可能性があります。また、プリエンブションの環境を設定しても、すべてのジョブの実行が保証されるわけではありません。プリエンブションの設定に加えて、クラスタワークフローを実行して、クラスタを作成しジョブを実行してから、そのクラスタを削除するエフェメラルクラスタを作成することができます。Databricks の並列の詳細については、Azure Databricks に問い合わせてください。

ネイティブ環境でのマッピング

ネイティブ環境でマッピングを実行できます。ネイティブ環境では、データ統合サービスが Developer tool からマッピングを実行します。スタンドアロンのマッピングや、ワークフローに含まれるマッピングを実行できます。

ネイティブ環境では、大規模な構造化されていないファイルや半構造化ファイル、Hive、またはソーシャルメディアの Web サイトのデータを読み取って処理できます。マッピングには、次のオブジェクトを含めることができます。

- Hive ソース
- ローカルシステムまたは HDFS のフラットファイルソースまたはターゲット
- ローカルシステムまたは HDFS の複合ファイルソース
- 構造化されていないファイル形式や半構造化ファイル形式を処理するための、データプロセッサトランスフォーマー
- ソーシャルメディアソース

データプロセッサのマッピング

データプロセッサトランスフォーマーは、マッピングで非構造化ファイル形式および半構造化ファイル形式を処理します。このトランスフォーマーは、ソースデータを MapReduce アプリケーションが処理できるフラット CSV レコードに変換します。

データプロセッサトランスフォーマーをメッセージング形式、リレーショナルデータ、HTML ページ、XML、JSON、AVRO、Parquet、Cobol、Microsoft Excel、Microsoft Word、PDF ドキュメントを処理するように設定できます。ACORD、HIPAA、HL7、EDI-X12、EDIFACT、SWIFT などの構造形式に変換するように設定することもできます。

例えば、あるアプリケーションが毎秒数 100 個のデータファイルを作成し、それらのファイルを 1 つのディレクトリに書き込みます。そのディレクトリからファイルを抽出するマッピングを作成し、それらをデータプロセッサトランスフォーマーに渡し、データをターゲットに書き込みます。

HDFS マッピング

HDFS への読み取りまたは書き込みのための HDFS マッピングを作成します。

固定長および区切り形式のファイルに対して読み取りおよび書き込みできます。圧縮ファイルに対して読み取りおよび書き込みできます。テキストファイルとバイナリファイル形式（HDFS からのシーケンスファイルなど）を読み取れます。ファイルの圧縮形式を指定できます。複雑なファイルデータオブジェクトのバイナリストリーム出力を、ファイルを解析するためのデータプロセッサトランスフォーマーへの入力として使用できます。

HDFS マッピング内に、次のオブジェクトを定義できます。

- HDFS からのデータを読み取るためのソースとしてのフラットファイルデータオブジェクトまたは複合データオブジェクト操作。
- トランスフォーマー。
- HDFS または任意のターゲットに書き込むためのターゲットとしてのフラットファイルデータオブジェクト。

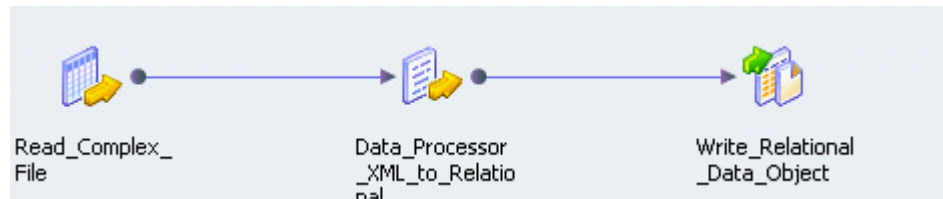
マッピングを検証して実行します。マッピングをデプロイし、そのマッピングを実行するか、ワークフロー内のマッピングタスクに追加できます。

HDFS データ抽出マッピングの例

あなたの組織は、顧客 ID、品目コード、品目の数量などの注文書の詳細を分析する必要があります。注文書の詳細は、半構造化圧縮 XML ファイルとして HDFS に保存されています。この階層データには、注文書の親階層レベルと、顧客の連絡先詳細の子階層レベルが含まれます。HDFS 内のファイルからすべての注文レコードを読み取るマッピングを作成します。このマッピングは、階層データをリレーショナルデータに変換し、そのデータをリレーショナルターゲットに書き込む必要があります。

ビジネス分析のために抽出データを使用できます。

次の図に、マッピングの例を示します。



次のオブジェクトを HDFS マッピング内で使用できます。

HDFS 入力

入力オブジェクト Read_Complex_File は、HDFS 内に保存された圧縮 XML ファイルを表す読み取りトランスフォーメーションです。

データプロセッサトランスフォーメーション

データプロセッサトランスフォーメーション Data_Processor_XML_to_Relational は、この XML ファイルを解析し、リレーショナル出力を提供します。

リレーショナル出力

出力オブジェクト Write_Relational_Data_Object は、Oracle データベース内のテーブルを表す書き込みトランスフォーメーションです。

マッピングを実行すると、データ統合サービスはバイナリストリーム内のファイルを読み取り、それをデータプロセッサトランスフォーメーションに渡します。データプロセッサトランスフォーメーションは、指定されたファイルを解析し、リレーショナル出力を提供します。この出力は、リレーショナルターゲットに書き込まれます。

ネイティブのランタイム環境または Hadoop ランタイム環境で実行するためのマッピングを設定できます。

以下の作業を実行して、マッピングの設定を行います。

1. Hadoop クラスタからのファイルを読み取るための HDFS 接続を作成します。
2. 複合ファイルデータオブジェクト読み取り操作を作成します。次のパラメータを指定します。
 - データオブジェクト内のリソースとしてのファイル。
 - ファイル圧縮形式。
 - HDFS ファイルの場所。
3. 必要に応じて、マッパーがファイルの読み取りに使用する入力形式を指定できます。
4. 複合ファイルデータオブジェクト読み取り操作をマッピングにドラッグアンドドロップします。
5. データプロセッサトランスフォーメーションを作成します。データプロセッサトランスフォーメーションで次のプロパティを設定します。
 - バッファ入力とバイナリデータ型に設定した入力ポート。
 - リレーショナル出力内に含めるカラム数に応じたリレーショナル出力ポート。ポートのサイズを指定します。XML 階層を表す XML スキーマ参照を使用します。目的の正規化された出力を指定します。例え

ば、PurchaseOrderNumber_Key を、注文書出力グループを顧客の詳細グループ関連付ける生成キーとして指定できます。

- ストリーマオブジェクトを作成し、ストリーマをスタートアップコンポーネントとして指定します。
6. Oracle データベースへのリレーショナル接続を作成します。
 7. リレーショナルデータオブジェクトをインポートします。
 8. リレーショナルデータオブジェクトの書き込みトランスフォーメーションを作成し、そのトランスフォーメーションをマッピングに追加します。

Hive マッピング

マッピング環境に基づいて、Hive との間でデータを読み書きできます。

ネイティブ環境では、Hive からデータを読み取れます。Hive からデータを読み取るには、次の手順を完了します。

1. Hive 接続を作成します。
2. Hive にソースまたはターゲットとしてアクセスするように、Hive 接続モードを設定します。
3. Hive 接続を使用して、Hive から読み取るデータオブジェクトを作成します。
4. マッピングにデータオブジェクトを追加し、マッピングをネイティブ環境で実行するように設定します。

Hadoop 環境では、Hive にデータを書き込めます。Hive にデータを書き込むには、次の手順を完了します。

1. Hive 接続を作成します。
2. Hive にソースまたはターゲットとしてアクセスするように、Hive 接続モードを設定します。
3. Hive 接続を使用して、Hive に書き込むデータオブジェクトを作成します。
4. マッピングにデータオブジェクトを追加し、マッピングを Hadoop 環境で実行するように設定します。

次のタイプのオブジェクトを Hive マッピングで定義できます。

- Hive からデータを読み取るための読み取りトランスフォーメーション
- トランスフォーメーション
- ターゲットまたは SQL データサービス Hadoop クラスタでマッピングを実行する場合、Hive にデータを書き込めます。

マッピングを検証して実行します。マッピングをデプロイし、そのマッピングを実行するか、ワークフロー内のマッピングタスクに追加できます。

Hive マッピングの例

あなたが所属する組織 HypoMarket Corporation は、顧客データを分析する必要があります。すべての顧客レコードを読み取るマッピングを作成します。SQL データサービスを作成して、エンドユーザーが仮想データベースに対してクエリを実行できるようにします。

Hive マッピングで次のオブジェクトを使用できます。

Hive 入力。

入力ファイルは、顧客名と連絡先の詳細が含まれる Hive テーブルです。

リレーショナルデータオブジェクトを作成します。Hive 接続を設定して、顧客データが含まれるテーブルをデータオブジェクトのリソースとして指定します。データオブジェクトをデータオブジェクトの読み取りとしてマッピングにドラッグします。

SQL データサービスの出力

SQL データサービスを Developer tool で作成します。エンドユーザーが使用できるようにするには、データサービスをアプリケーションに含めて、アプリケーションをデータ統合サービスにデプロイします。アプリケーションの実行中に、接続文字列を指定して、サードパーティ製クライアントツールから SQL データサービスに接続します。

クライアントツールから SQL クエリを実行して、顧客データにアクセスできます。

ソーシャルメディアマッピング

Facebook や LinkedIn などのソースからソーシャルメディアデータを読み取るためのマッピングを作成します。

ソーシャルメディアデータを抽出し、それらをネイティブ環境のターゲットにのみロードできます。このデータを解析するか、データマイニングや分析のためにデータを使用するかを選択できます。

Hadoop のデータを処理または分析するには、まず、データをリレーショナルターゲットかフラットファイルターゲットに移動し、次に Hadoop クラスタでマッピングを実行する必要があります。

Developer tool で次の Informatica アダプタを使用できます。

- PowerExchange for DataSift
- PowerExchange for Facebook
- PowerExchange for LinkedIn
- PowerExchange for Twitter
- PowerExchange for Web Content-Kapow Katalyst

詳細については、該当する PowerExchange アダプタのマニュアルを参照してください。

Twitter マッピングの例

あなたが所属する組織 HypoMarket Corporation は、2012 年 2 月のリリース以降、HypoBasket という製品について好意的なメンションを行ったすべてのツイートを確認する必要があります。

HypoBasket という単語が含まれるツイートを識別し、そのレコードをテーブルに書き込むマッピングを作成します。

Twitter マッピングで次のオブジェクトを使用できます。

Twitter の入力

マッピングソースは、リソースの検索を含む Twitter データオブジェクトです。

物理データオブジェクトを作成して、データオブジェクトをマッピングに追加します。検索リソースを物理データオブジェクトに追加します。次のクエリを使用して、クエリパラメータを変更します。

```
QUERY=HypoBasket:)&since:2012-02-01
```

ソータートランスフォーメーション

必要に応じて、タイムスタンプに基づいてデータをソートします。

マッピングにソータートランスフォーメーションを追加します。昇順の方向のソートキーとして、タイムスタンプを指定します。

マッピング出力

リレーショナルデータオブジェクトをターゲットとしてマッピングに追加します。

マッピングの実行後、データ統合サービスはターゲットテーブルに抽出したツイートを書き込みます。テキスト分析や評価分析のツールを使用して、ツイートを分析できます。

ネイティブ環境の最適化

ネイティブ環境を最適化して、パフォーマンスを向上させることができます。パフォーマンスを向上させるには、データ統合サービスを、グリッド上で実行し、データの処理に複数パーティションを使用するように設定できます。ネットワーク、ハードウェア、またはサービスに一時的な障害が発生した場合にも、ドメインが確実に実行を継続できるようにするために、高可用性を有効にすることもできます。

プロファイル、セッション、およびワークフローをグリッドで実行して、処理の帯域幅を増やすことができます。グリッドは、プロファイル、セッション、およびワークフローを実行するノードのグループに割り当てられるエイリアスです。グリッドを有効にすると、データ統合サービスがグリッドの使用可能な各ノードでサービスプロセスを実行し、パフォーマンスと拡張性が向上します。

また、パフォーマンスを向上させるために、パーティション化を有効にしたマッピングも実行できます。パーティション化されたセッションやパーティション化されたマッピングを実行すると、データ統合サービスは、各パーティションの抽出、トランスフォーメーション、およびロードを並行して実行します。

ドメインに対して高可用性を設定できます。高可用性によりドメイン内の単一の障害ポイントが除去され、障害の際にサービス中断が最小限になります。

グリッドでのビッグデータの処理

グリッドで統合サービスを実行して、処理の帯域幅を増やすことができます。グリッドを有効にすると、統合サービスはグリッドの使用可能な各ノードでサービスプロセスを実行し、パフォーマンスと拡張性が向上します。

ビッグデータの大量のデータを処理するために、追加の帯域幅が必要になる場合があります。例えば、モデルリポジトリプロファイルを極めて大きいデータセットで実行する場合、データ統合サービスグリッドは、プロファイルを複数のマッピングに分割し、これらのマッピングをグリッド内の異なるノードで同時に実行します。

グリッドでマッピングを実行すると、データ統合サービスは、グリッド内のノードで複数の DTM プロセスにマッピングを配布します。グリッドでプロファイルを実行すると、データ統合サービスは、プロファイルを複数のマッピングに分割し、マッピングをグリッド内のノードで複数の DTM プロセスに配布します。グリッドを最適化することで、パフォーマンスを向上し、データ統合サービスのスケーラビリティを高めることができます。

グリッドを最適化するには、グリッドに注記を追加して、データ統合サービスの処理の帯域幅を広げます。

データ統合サービスグリッドの詳細については、『Informatica Administrator ガイド』を参照してください。

パーティションでのビッグデータの処理

パフォーマンスを向上させるために、パーティション化を設定したモデルリポジトリマッピングを実行できます。パーティション化を設定したマッピングを実行すると、データ統合サービスは、各パーティションの抽出、トランスフォーメーション、およびロードを並行して実行します。

大きいデータセットを処理するマッピングは、処理に時間がかかり、データスループットが低下する可能性があります。パーティション化を設定すると、データ統合サービスはセッションまたはマッピングを処理するために追加のスレッドを使用するため、パフォーマンスを向上させることができます。

マッピングが実行されるノードに複数の CPU がある場合、データ統合サービスがマッピングを実行するとき、並行処理を最大化できます。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。

必要に応じて、開発者は、マッピングの並行処理の最大値を Developer tool で変更できます。デフォルトでは、各マッピングの最大並行処理が自動的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、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最小値が使用されます。

詳細については、『*Informatica のアプリケーションサービスガイド*』と『*Informatica Developer マッピングガイド*』を参照してください。

パーティションの最適化

モデルリポジトリマッピングのパーティションを最適化して、パフォーマンスを向上させることができます。パーティションの追加、パフォーマンスが最良のパーティションタイプの選択、使用する CPU の追加、パーティション化のソースまたはターゲットデータベースの最適化などを行えます。

パーティションを最適化するには、次のタスクを実行します。

パーティション数を増やす。

データ統合サービスまたはマッピングの最大並行処理の値を増やす場合は、モデルリポジトリマッピングを設定するときに、パーティションの数を増やします。

パーティションの数を増やすことにより、データ統合サービスはソースへの複数の接続を作成し、ソースデータのパーティションを同時に処理することができます。パーティションの数を増やすとスレッドの数も増加します。これにより、データ統合サービスノードへのロードも増加します。データ統合サービスノードに十分な CPU バンド幅がある場合は、複数のデータ行を同時に処理すると、パフォーマンスを向上させることができます。

注: 単一ノードのデータ統合サービスを使用しており、大量のデータを処理するセッションまたはマッピングで大量のパーティションをデータ統合サービスが使用する場合は、システムがオーバーロードする可能性があります。

複数の CPU を使用する。

対称型マルチプロセッシング（SMP）プラットフォーム上では、複数の CPU を使用してデータのパーティションを並列に処理できます。

パーティション化のためにソースデータベースを最適化する。

パーティション化のために、ソースデータベースを最適化できます。データベースの調整、並行クエリの有効化、異なるテーブルスペースへのデータの分割、ソート済みデータのグループ化などを行えます。

パーティション化のためにターゲットデータベースを最適化する。

パーティション化のために、ターゲットデータベースを最適化できます。データベースへの並行挿入の有効化、異なるテーブルスペースへのデータの分割、データベースに許可されたセッションの最大数の増加などを行えます。

高可用性

高可用性により Informatica ドメイン内の単一の障害ポイントが除去され、障害の際にサービス中断が最小限になります。ドメインの高可用性を設定すると、ドメインは一時的なネットワーク、ハードウェア、またはサービス障害の場合にも実行を継続できます。高可用性は、ドメイン、アプリケーションサービス、およびアプリケーションクライアントに対して設定できます。

Informatica ドメインでは、以下の高可用性コンポーネントにより、サービスの高可用性を実現しています。

- レジリエンス。Informatica ドメインは、レジリエンスタイムアウトの期限切れまで、または障害から回復するまでの間、一時的な接続障害を許容できます。
- 再開およびフェイルオーバー。各プロセスは、使用不能状態になった後、同じノード上またはバックアップノード上で再開できます。
- リカバリ。サービスが中断されても操作を完了できます。サービスプロセスが再開またはフェイルオーバーした後に、サービスの状態を復元して操作をリカバリします。

可用性の高い Informatica 環境を計画する場合、内部の Informatica コンポーネントと Informatica の外部にあるシステムとの違いを検討します。内部コンポーネントには、サービスマネージャ、アプリケーションサー

ビス、コマンドラインプログラムが含まれています。外部システムには、ネットワーク、ハードウェア、データベース管理システム、FTP サーバー、メッセージキュー、および共有ストレージが含まれています。

Informatica 環境の高可用性機能は、ユーザーのライセンスに基づいて提供されます。

第 3 章

ソース

この章では、以下の項目について説明します。

- [ソースの概要, 59 ページ](#)
- [PowerExchange アダプタソース, 59 ページ](#)
- [Databricks のソース, 60 ページ](#)
- [ファイルソース Hadoop 上, 62 ページ](#)
- [Hadoop のリレーショナルソース, 66 ページ](#)
- [Hive ソース Hadoop 上, 67 ページ](#)
- [Hadoop の Sqoop ソース, 70 ページ](#)

ソースの概要

ネイティブ環境または非ネイティブ環境のソースが含まれる非ネイティブ環境に、マッピングをプッシュできます。非ネイティブ環境でソースを参照する場合、一部のソースに制限事項があります。

非ネイティブ環境のソースにアクセスするには、PowerExchange アダプタが必要です。

Hadoop 環境でのマッピング実行時に、HDFS ソースはリモートクラスタ上に配置できません。

PowerExchange アダプタソース

PowerExchange ソースは、アダプタに基づいて、Hadoop または Databricks 環境で実行できます。

次の表に、PowerExchange ソースと、ソースをサポートする非ネイティブエンジンを示します。

PowerExchange アダプタ	サポートされるエンジン
Amazon Redshift	- Blaze - Spark
Amazon S3	- Blaze - Spark
Google Analytics	- Spark

PowerExchange アダプタ	サポートされるエンジン
Google BigQuery	- Spark
Google Cloud Spanner	- Spark
HBase	- Blaze - Spark
HDFS	- Blaze - Spark
Hive	- Blaze - Spark
MapR-DB	- Blaze - Spark
Microsoft Azure Blob Storage	- Spark - Databricks Spark
Microsoft Azure Cosmos DB	- Spark - Databricks Spark
Microsoft Azure Data Lake Store	- Spark - Databricks Spark
Microsoft Azure SQL Data Warehouse	- Spark - Databricks Spark
Snowflake	- Spark
Teradata Parallel Transporter API	- Blaze - Spark

Databricks のソース

Databricks 環境で実行するマッピングには、ファイルおよびデータベースソースを含めることができます。次の表に、格納のタイプとソースを示します。

格納のタイプ	ソース
ファイル	- Microsoft Azure Blob Storage - Microsoft Azure Data Lake Store
データベース	- Microsoft Azure Cosmos DB - Microsoft Azure SQL Data Warehouse

ADLS の複合ファイルソース

ADLS データオブジェクトからデータを読み取るには、PowerExchange for Microsoft Azure Data Lake Store 接続を使用します。

次の表に、Databricks 環境の ADLS ストレージ内でマッピングが処理できる複合ファイルを示します。

ファイルタイプ	形式
Avro	- Flat - 階層
JSON	- Flat - 階層
Parquet	- Flat - 階層
注: 階層ファイルの読み取り操作が、プロジェクトカラムに対して複合データ型として有効にされている必要があります。	

Azure Blob の複合ファイルソース

Blob Storage データオブジェクトからデータを読み取るには、PowerExchange for Microsoft Azure Blob Storage 接続を使用します。圧縮されたバイナリファイルを読み取れます。

次の表に、Databricks 環境の Azure Blob Storage 内でマッピングが処理できる複合ファイルを示します。

ファイルタイプ	形式
Avro	- Flat - 階層
JSON	- Flat - 階層
Parquet	- Flat - 階層
注: 階層ファイルの読み取り操作が、プロジェクトカラムに対して複合データ型として有効にされている必要があります。	

Databricks ソースのルールとガイドライン

ファイルソース

ファイルソースについては、次の一般的なルールおよびガイドラインを考慮します。

- フラットファイルソースは、Microsoft Azure Blob Storage または Microsoft Azure Data Lake Store に置く必要があります。
- 行区切り文字は/n にする必要があります。
- ファイルを固定長にすることはできません。
- 複数のカラム区切り文字はサポートされません。

- 複数行のデータを読み取るには、テキスト修飾子を一重引用符または二重引用符に設定し、データをその引用符付きの修飾子で囲みます。
- 空の値のみが NULL 値として扱われます。

NULL 処理

NULL 処理については、次のルールおよびガイドラインを考慮します。

NULL に変換される予期しない値

Databricks Spark エンジンでは、フィールドに予期しない値が含まれている場合、次のシナリオに基づいて、同じレコードのすべてのフィールドに NULL 値を生成します。

- string データを numeric カラムに渡すなど、型不一致が発生した。
- bigint データ型、int データ型などで、データが範囲を超えている。

NULL 行をフィルタするには、フィルタトランスフォーメーションの使用を考慮します。

NULL に変換される日付/時刻値

Databricks Spark エンジンでは、日付/時刻値を読み取る際に、Developer tool のランタイム設定の「マッピング」プロパティで設定されている形式を使用します。ソースから読み取った日付の形式が Developer tool に設定されている形式と一致しない場合は、Databricks Spark エンジンでは日付値を NULL に変換します。Developer tool のデフォルト値は MM/DD/YYYY HH24:MI:SS です。

double と decimal の変換

Databricks Spark エンジンでは、Azure ソースから読み取ったときに、double と decimal データ型を科学的表記に変換します。データを double または decimal に変換してターゲットに書き込むときに、15 を超える精度を削除し、15 桁の精度を維持します。

ファイルソース Hadoop 上

Hadoop 環境で実行されるマッピングでは、複合ファイルとフラットファイルを処理できます。

大量のデータを読み取るには、複合ファイルソースを接続し、同じ形式とプロパティのファイルのディレクトリからデータを読み取ることができます。圧縮されたバイナリファイルを読み取れます。

Hadoop 環境では、次のストレージタイプから複合ファイルを読み取れます。

- Amazon Simple Storage Service (Amazon S3)
- Hadoop 分散ファイルシステム (HDFS)
- MapR File System (MapR-FS)
- Microsoft Azure Blob Storage (Azure Blob Storage)
- Microsoft Azure Data Lake Store (ADLS)

Amazon S3 の複合ファイルソース

Amazon S3 データオブジェクトからデータを読み取るには、PowerExchange for Amazon S3 接続を使用します。

次の表に、Hadoop 環境の Amazon S3 ストレージ内でマッピングが処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ^{1 2}	- Blaze - Spark
JSON	- Flat - 階層 ^{1 2}	- Blaze - Spark
ORC	- Flat	- Spark
Parquet	- Flat - 階層 ^{1 2}	- Blaze - Spark
¹ Blaze エンジンで実行するには、複合ファイルデータオブジェクトがデータプロセッサトランスフォーマーに接続されている必要があります。 ² Spark エンジンで実行するには、複合ファイル読み取り操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

ADLS の複合ファイルソース

ADLS データオブジェクトからデータを読み取るには、PowerExchange for HDFS または PowerExchange for Microsoft Azure Azure Data Lake Store 接続を使用できます。PowerExchange for Microsoft Azure Data Lake Store 接続を使用する場合は、ADLS から圧縮されたバイナリファイルを読み取ることはできません。

次の表に、Hadoop 環境の ADLS ストレージ内で PowerExchange for Microsoft Azure Data Lake Store 接続が処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ¹	Spark
JSON	- Flat - 階層 ¹	Spark
Parquet	- Flat - 階層 ¹	Spark
¹ Spark エンジンで実行するには、複合ファイル読み取り操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

次の表に、Hadoop 環境の ADLS ストレージ内で PowerExchange for HDFS 接続が処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ^{1 2}	- Blaze - Spark
JSON	- Flat ¹ - 階層 ^{1 2}	- Blaze - Spark

ファイルタイプ	サポートされている形式	サポートされるエンジン
ORC	- Flat	- Spark
Parquet	- Flat - 階層 ^{1 2}	- Blaze - Spark
¹ Blaze エンジンで実行するには、複合ファイルデータオブジェクトがデータプロセッサトランスフォーメーションに接続されている必要があります。 ² Spark エンジンで実行するには、複合ファイル読み取り操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

Azure Blob の複合ファイルソース

Azure Blob データオブジェクトからデータを読み取るには、PowerExchange for HDFS または PowerExchange for Microsoft Azure Blob Storage 接続を使用できます。

次の表に、Hadoop 環境の Azure Blob Storage 内でマッピングが処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ¹	Spark
JSON	- Flat - 階層 ¹	Spark
Parquet	- Flat - 階層 ¹	Spark
¹ Spark エンジンで実行するには、複合ファイル読み取り操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

次の表に、Hadoop 環境の Azure Blob Storage 内で PowerExchange for HDFS 接続が処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ^{1 2}	- Blaze - Spark
JSON	- Flat ¹ - 階層 ^{1 2}	- Blaze - Spark
ORC	- Flat	- Spark
Parquet	- Flat - 階層 ^{1 2}	- Blaze - Spark
¹ Blaze エンジンで実行するには、複合ファイルデータオブジェクトがデータプロセッサトランスフォーメーションに接続されている必要があります。 ² Spark エンジンで実行するには、複合ファイル読み取り操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

MapR-FS の複合ファイルソース

MapR-FS データオブジェクトからデータを読み取るには、PowerExchange for HDFS 接続を使用します。

次の表に、Hadoop 環境の MapR-FS ストレージ内でマッピングが処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ^{1 2}	- Blaze - Spark
JSON	- Flat ¹ - 階層 ^{1 2}	- Blaze - Spark
ORC	- Flat	- Spark
Parquet	- Flat - 階層 ^{1 2}	- Blaze - Spark
¹ Blaze エンジンで実行するには、複合ファイルデータオブジェクトがデータプロセッサトランスフォーマーに接続されている必要があります。 ² Spark エンジンで実行するには、複合ファイル読み取り操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

HDFS の複合ファイルソース

HDFS データオブジェクトからデータを読み取るには、PowerExchange for HDFS 接続を使用します。

次の表に、Hadoop 環境の HDFS ストレージ内でマッピングが処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ^{1 2}	- Blaze - Spark
JSON	- Flat ¹ - 階層 ^{1 2}	- Blaze - Spark
ORC	- Flat	- Spark
Parquet	- Flat - 階層 ^{1 2}	- Blaze - Spark
¹ Blaze エンジンで実行するには、複合ファイルデータオブジェクトがデータプロセッサトランスフォーマーに接続されている必要があります。 ² Spark エンジンで実行するには、複合ファイル読み取り操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

フラットファイルソース Hadoop 上

Hadoop 環境で実行されるマッピングでは、ネイティブ環境からフラットファイルソースを読み取れます。

フラットファイルソースを読み取るマッピングを設定するときは、次の制限事項を考慮します。

- 間接のソースタイプは使用できません。
- フラットファイルソースの行サイズは、190 MB を超えることはできません。

- 実行時にコマンドを使用してフラットファイルデータを生成または変換し、その出力をフラットファイル読み取りに送信することはできません。

ソースファイル名の生成

フラットファイルデータオブジェクトのソースファイル名を生成できます。ファイル名カラムの内容は、異なる実行モード間で一貫性が保たれます。

要求されたファイルタイプの特定のエンジンに処理をプッシュすると、ファイル名カラムは、次の形式に基づいてパスを返します。

ランタイムエンジン	ソースファイルの場所	返されたパス
Blaze	ローカルシステム	<staged path><local file path> 以下に例を示します。 hdfs://host name:port/hive/warehouse/home/devbld/Desktop/ff.txt
Spark	HDFS	hdfs://<host name>:<port>/<file name path> 以下に例を示します。 hdfs://host name:port/hive/warehouse/ff.txt
Spark	ローカルシステム	<local file path> 以下に例を示します。 /home/devbld/Desktop/ff.txt

ファイル名カラムは、高可用性クラスタに対するコンテンツを hdfs://<host name>/<file name path> の形式で返します。

例: hdfs://irldv:5008/hive/warehouse/ff.txt

Hadoop のリレーショナルソース

Blaze エンジンでは、リレーショナルソースを含むマッピングを実行できます。

データ統合サービスは、マッピング前の SQL コマンドまたはマッピング後の SQL コマンドをリレーショナルソースに対して実行しません。Hadoop 環境のリレーショナルソースに、PreSQL プロパティまたは PostgreSQL プロパティを使用してマッピングを検証および実行することはできません。

データ統合サービスは、次のリレーショナルソースからの読み取りに複数のパーティションを使用できます。

- IBM DB2
- Oracle

Hive ソース Hadoop 上

Hadoop 環境で実行する Informatica のマッピングに Hive ソースを含めることができます。

Hadoop 環境で実行するマッピングに Hive ソースを設定する場合は、次の制限事項を考慮します。

- Hive ソース定義に Unicode 文字が含まれると、マッピングの実行に失敗します。
- Hive ソースは、リモートクラスタに置くことはできません。リモートクラスタは、マッピングで Hadoop 接続が参照するマシンのリモートにあるクラスタです。
- サードパーティの Hive JDBC ドライバは、10 進データ型に適切な精度と位取りの値を返しません。その結果、10 進データ型の Hive テーブルを Developer tool にインポートすると、10 進データ型の精度は 38 に、位取りは 0 に設定されます。Hive のバージョンに基づいて、次の設定ルールとガイドラインを考慮します。
 - Hive 0.11: Developer tool で、10 進データ型の精度と位取りのデフォルト値を受け入れます。
 - Hive 0.12: Developer tool で、10 進データ型の精度と位取りのデフォルト値を受け入れます。
 - Cloudera CDH 5.0 を使用する Hive 0.12: Developer tool で、10 進データ型のソースカラムに対する精度フィールドと位取りフィールドを設定できます。
 - Hive 0.13 以上: Developer tool で、10 進データ型のソースカラムに対する精度フィールドと位取りフィールドを設定できます。
 - Hive 0.14 以上: Hive データベースの 10 進データ型で使用される精度と位取りも Developer tool に表示されます。

Spark エンジンで実行されるマッピングに、パーティション化された Hive ソーステーブルとバケット化されたソースを使用できます。

PreSQL コマンドおよび PostSQL コマンド

Hive ソース向けに SQL コマンドを作成できます。挿入、更新、および削除などの SQL 文を Hive ソースで実行するために、SQL コマンドを実行できます。

PreSQL は、マッピングがソースから読み取る前に、Hive ソースに対して実行される SQL コマンドです。

PostSQL は、マッピングがターゲットに書き込んだ後で、Hive ソースに対して実行される SQL コマンドです。

PreSQL と PostSQL は Spark エンジンで使用できます。データ統合サービスは、Hive ソースに対する PreSQL コマンドや PostSQL コマンドを検証しません。

注: Hive コマンドラインユーティリティで次のクエリを実行して、SQL を手動で検証できます。

```
CREATE VIEW <table name> (<port list>) AS <SQL>
```

ここで、

- <table name>は選択した名前
- <port list>は、ソースに含まれるポートのカンマ区切りリスト
- <SQL>は検証するクエリ

マッピング前の SQL コマンド

PreSQL は、マッピングがソースから読み取る前に、Hive ソースに対して実行される SQL コマンドです。

例えば、マッピング内で Hive ソースを使用しているとします。Hive ソースに格納されたデータが定期的に変更されるため、最新のレコードから確実に読み取りが実行されるようにするために、マッピングでソースからの読み取りが実行される前に、Hive ソースのデータを更新する必要があります。Hive ソースを更新するために、PreSQL コマンドを設定できます。

マッピング後の SQL コマンド

PostSQL は、マッピングがターゲットに書き込んだ後で、Hive ソースに対して実行される SQL コマンドです。

例えば、マッピング内で Hive ソースを使用しているとします。ターゲットへのマッピングの書き込み後、Hive ソースに格納されているストレージレコードを削除する場合があります。Hive ソースからデータが早期に削除されることがないようにするため、ターゲットへのデータの書き込みがマッピングで行われた後にだけコマンドを実行します。マッピングによるターゲットへの書き込み後に、Hive ソーステーブルのレコードを削除するために、PostSQL コマンドを設定できます。

マッピング前またはマッピング後の SQL コマンドのルールとガイドライン

PreSQL コマンドと PostSQL コマンドを Hive ソースに対して実行する場合は、次の制限事項を考慮します。

- SQL オーバーライドを Hive ソースに作成する際、キーワードまたは特殊文字をバッククォート (`) 文字で囲む必要があります。
- Hadoop 環境の Hive ソースでマッピングを実行する場合は、マッピング前の SQL コマンドに含まれるローカルパスへの参照は、データ統合サービスノードに相対的になります。ネイティブ環境の Hive ソースでマッピングを実行する場合は、マッピング前の SQL コマンドに含まれるローカルパスへの参照は、Hive サーバーノードに相対的になります。

Blaze エンジン上の Hive ソースのルールとガイドライン

Blaze エンジンで実行する Informatica のマッピングに Hive ソースを含めることができます。

Blaze エンジンで実行するマッピングに Hive ソースを設定する場合は、次のルールとガイドラインを考慮します。

- Blaze マッピング向けの Hive ソースには、TEXT、Sequence、Avro、RCfile、ORC、および Parquet のストレージ形式が含まれます。
- Blaze エンジンで実行するマッピングはバケット化できます。
- Hive ソースと Hive ACID テーブル。
- Hive ACID テーブルはバケット化されている必要があります。
- Blaze エンジンは、ロックが有効化された Hive テーブルをサポートします。
- Hive ソースでは、Hive テーブル名、カラム名、およびスキーマ名に引用符付きの識別子を含めることができます。
- Blaze マッピング用の Hive ソースに含まれる TEXT ストレージ形式は、カラム区切り文字として ASCII を、行セパレータとして改行文字をサポートできます。ASCII 文字の 16 進値は使用できません。例えば、3B ではなくセミコロン (;) を使用します。
- Blaze マッピング用の Hive ソースに SQL オーバーライドを定義できます。
- Blaze エンジンは、Hive ソースとして RCFile から読み取ることができます。RCFile テーブルから読み取るためには、SerDe 句を使用してテーブルを作成する必要があります。
- Blaze エンジンは、圧縮された Hive テーブルから読み取りを行えます。圧縮された Hive テーブルから読み取るには、TBLPROPERTIES 句を設定する必要があります。

Hive テーブルとしての RCFile

Blaze エンジンは、Hive テーブルとして RCFile への読み書きを実行できます。ただし、Blaze エンジンがサポートするのは ColumnarSerDe SerDe のみです。Hortonworks では、RCFile のデフォルトの SerDe は LazyBinaryColumnarSerDe です。RCFile テーブルの読み書きを行うには、SerDe を `org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe` として指定することで、このテーブルを作成する必要があります。

以下に例を示します。

```
CREATE TABLE TEST_RCFFile
(id int, name string)
ROW FORMAT SERDE
'org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe' STORED AS RCFILE;
```

デフォルトの RCFile SerDe を Ambari または Cloudera マネージャから設定することもできます。
hive.default.rcfile.serde プロパティを org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe に設定します。

圧縮された Hive テーブル

Blaze エンジンでは、圧縮された Hive テーブルへの読み書きを実行できます。ただし、圧縮された Hive テーブルからの読み取りと、圧縮形式の Hive テーブルへの書き込みのためには、TBLPROPERTIES 句を次のように設定する必要があります。

- テーブルの作成時に、テーブルプロパティを次のように設定します。
TBLPROPERTIES ('property_name'='property_value')
- テーブルがすでにある場合、次のテーブルプロパティを設定するようにテーブルを変更します。
ALTER TABLE table_name SET TBLPROPERTIES ('property_name' = 'property_value');

プロパティの名前と値は大文字と小文字が区別されません。ファイル形式に応じて、テーブルプロパティは異なる値を取得します。

次の表に、ファイル形式別のプロパティの名前と値の一覧を示します。

ファイル形式	テーブルプロパティの名前	テーブルプロパティの値
Avro	avro.compression	BZIP2、deflate、Snappy
ORC	orc.compress	Snappy、ZLIB
Parquet	parquet.compression	GZIP、Snappy
RCFile	rcfile.compression	Snappy、ZLIB
シーケンス	sequencefile.compression	BZIP2、GZIP、LZ4、Snappy
テキスト	text.compression	BZIP2、GZIP、LZ4、Snappy

注: Blaze エンジンでは、ORC 形式として格納された Hive ターゲットへの書き込み時に、デフォルトの ZLIB 圧縮形式でデータを書き込みません。圧縮形式でデータを書き込むには、このテーブルを変更し、TBLPROPERTIES 句が ORC ファイル形式に ZLIB または Snappy を使用するように、この句を設定します。

次のテキストに、テーブルを作成し、テーブルを変更するためのコマンド例を示します。

- テーブルの作成は次のとおりです。
create table CBO_3T_JOINS_CUSTOMER_HIVE_SEQ_GZIP
(C_CUSTKEY DECIMAL(38,0), C_NAME STRING,C_ADDRESS STRING,
C_PHONE STRING,C_ACCTBAL DECIMAL(10,2),
C_MKTSEGMENT VARCHAR(10),C_COMMENT VARCHAR(117))
partitioned by (C_NATIONKEY DECIMAL(38,0))
TBLPROPERTIES ('sequencefile.compression'='gzip')
stored as SEQUENCEFILE;
- テーブルの変更は次のとおりです。
ALTER TABLE table_name
SET TBLPROPERTIES ('avro.compression'='BZIP2');

Hadoop の Sqoop ソース

Sqoop ソースは、Hadoop 環境のマッピングで有効です。

JDBC に準拠したデータベースを、Hadoop 環境で実行される Informatica マッピングの Sqoop ソースとして含めることができます。

例えば、次のソースを Sqoop マッピングに含めることができます。

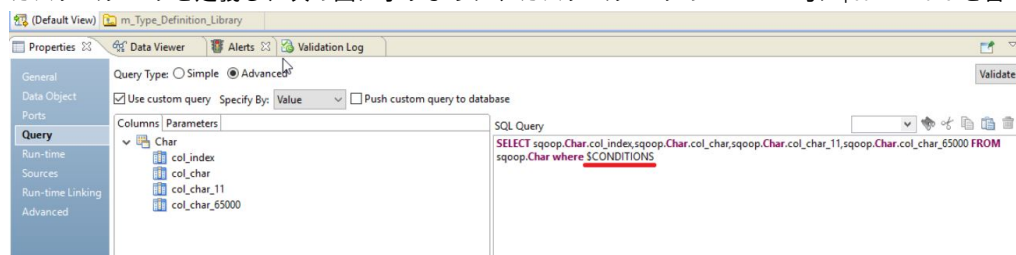
- Aurora
- Greenplum
- IBM DB2
- IBM DB2 for z/OS
- Microsoft SQL Server
- Netezza
- Oracle
- Teradata
- Vertica

Sqoop を使用した Vertica ソースからのデータ読み取り

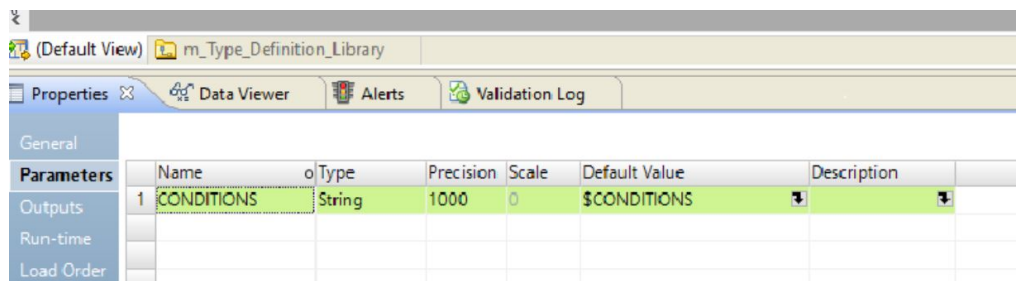
Sqoop 接続を使用して Vertica データを読み取るには、Sqoop にカスタムクエリが必要です。

WHERE 句に \$CONDITIONS パラメータを指定したカスタムクエリを作成します。次に、CONDITIONS という名前のマッピングパラメータを定義し、その値を \$CONDITIONS に設定します。実行時に Sqoop が \$CONDITIONS を一意の条件式に置き換えます。

1. Sqoop データオブジェクトを開き、【プロパティ】ビューの【クエリ】タブをクリックします。
2. カスタムクエリを定義し、次の図に示すように、カスタムクエリの WHERE 句に \$CONDITIONS を含めます。



3. Sqoop マッピングを開き、【プロパティ】ビューの【パラメータ】タブをクリックします。
4. 次の図に示すように、CONDITIONS という名前のマッピングパラメータを定義し、その値を \$CONDITIONS に設定します。



Sqoop ソースのルールとガイドライン

マッピングに Sqoop ソースを設定する場合は、次のルールとガイドラインを考慮します。

- マッピングにソート条件を指定した場合、データ統合サービスはソート基準条件を無視します。
- Sqoop ソースのカラムはソートできません。
- Sqoop ソースから個別の行を読み取ることはできません。
- Oracle ソースから Sqoop を使用してデータを読み取り、Blaze または Spark エンジンでマッピングを実行した場合、Sqoop は所有者名の大文字と小文字を区別します。
- Sqoop では、JDBC 接続の【ユーザー名】と【パスワード】の各フィールドで設定した値が使用されます。JDBC 接続またはマッピングで--username 引数または--password 引数を設定すると、Sqoop は引数を無視します。データベースにアクセスするためのパスワードファイルを作成しても、Sqoop ではパスワードファイルが無視されます。

Sqoop クエリのルールとガイドライン

マッピングに Sqoop クエリを設定する場合は、次のルールとガイドラインを考慮します。

- マッピングのデフォルトクエリを詳細クエリにオーバーライドするには、CONDITIONS という名前のマッピングパラメータを定義してその値を\$CONDITIONS に設定する必要があります。次に、\$CONDITIONS をカスタムクエリの WHERE 句に含める必要があります。カスタムクエリの設定の詳細については、Informatica ナレッジベースの記事「[KB 564550](#)」を参照してください。
- カスタムクエリを定義する場合、カスタムクエリのメタデータがソースオブジェクトのメタデータと一致することを確認する必要があります。一致しないと、Sqoop がターゲットに空白の値を書き込む可能性があります。
- OraOop を有効にし、Oracle ソースのデータを Sqoop を通じて読み取るように詳細クエリを設定すると、マッピングは Spark エンジンで失敗します。

第 4 章

ターゲット

この章では、以下の項目について説明します。

- [ターゲットの概要, 72 ページ](#)
- [PowerExchange アダプタターゲット, 72 ページ](#)
- [Databricks のターゲット, 73 ページ](#)
- [Hadoop でのファイルターゲット, 75 ページ](#)
- [Hadoop でのメッセージターゲット, 79 ページ](#)
- [Hadoop のリレーショナルターゲット, 80 ページ](#)
- [Hadoop での Hive ターゲット, 80 ページ](#)
- [Hadoop の Sqoop ターゲット, 84 ページ](#)

ターゲットの概要

ネイティブ環境または非ネイティブ環境のターゲットが含まれる非ネイティブ環境に、マッピングをプッシュできます。非ネイティブ環境でターゲットを参照する場合、一部のターゲットに制限事項があります。

非ネイティブ環境のターゲットにアクセスするには、PowerExchange アダプタが必要になることがあります。

PowerExchange アダプタターゲット

PowerExchange ターゲットは、アダプタに基づいて、Hadoop または Databricks 環境で実行できます。

次の表に、PowerExchange ターゲットと、ターゲットをサポートする非ネイティブエンジンを示します。

PowerExchange アダプタ	サポートされるエンジン
Amazon Redshift	- Blaze - Spark
Amazon S3	- Blaze - Spark
Google Analytics	- Spark

PowerExchange アダプタ	サポートされるエンジン
Google BigQuery	- Spark
Google Cloud Spanner	- Spark
HBase	- Blaze - Spark
HDFS	- Blaze - Spark
Hive	- Blaze - Spark
MapR-DB	- Blaze - Spark
Microsoft Azure Blob Storage	- Spark - Databricks Spark
Microsoft Azure Cosmos DB	- Spark - Databricks Spark
Microsoft Azure Data Lake Store	- Blaze - Spark - Databricks Spark
Microsoft Azure SQL Data Warehouse	- Spark - Databricks Spark
Snowflake	- Spark
Teradata Parallel Transporter API	- Blaze - Spark

Databricks のターゲット

Databricks 環境で実行するマッピングには、ファイルおよびデータベースターゲットを含めることができます。

次の表に、格納のタイプとターゲットを示します。

格納のタイプ	ターゲット
ファイル	- Microsoft Azure Blob Storage - Microsoft Azure Data Lake Store
データベース	- Microsoft Azure Cosmos DB - Microsoft Azure SQL Data Warehouse

ADLS の複合ファイルターゲット

ADLS データオブジェクトにデータを書き込むには、PowerExchange for Microsoft Azure Data Lake Store 接続を使用します。

次の表に、Databricks 環境の ADLS ストレージ内でマッピングが処理できる複合ファイルを示します。

ファイルタイプ	形式
Avro	- Flat - 階層
JSON	- Flat - 階層
Parquet	- Flat - 階層
注: 階層ファイルの書き込み操作が、プロジェクトカラムに対して複合データ型として有効にされている必要があります。	

Azure Blob の複合ファイルターゲット

Blob Storage データオブジェクトにデータを書き込むには、PowerExchange for Microsoft Azure Blob Storage 接続を使用します。圧縮されたバイナリファイルを書き込むことができます。

次の表に、Databricks 環境の Azure Blob Storage 内でマッピングが処理できる複合ファイルを示します。

ファイルタイプ	形式
Avro	- Flat - 階層
JSON	- Flat - 階層
Parquet	- Flat - 階層
注: 階層ファイルの書き込み操作が、プロジェクトカラムに対して複合データ型として有効にされている必要があります。	

Databricks ターゲットのルールとガイドライン

ファイルターゲット

ファイルターゲットについては、次の一般的なルールおよびガイドラインを考慮します。

- ファイルターゲットは、Microsoft Azure Blob Storage または Microsoft Azure Data Lake Store に置く必要があります。
- 行区切り文字は/n にする必要があります。
- ファイルを固定長にすることはできません。
- 複数のカラム区切り文字はサポートされません。

- 空の値のみが NULL 値として扱われます。

NULL 処理

NULL 処理については、次のルールおよびガイドラインを考慮します。

Null 値変換

Databricks Spark エンジンでは、ターゲットに書き込むときに、NULL 値を空の文字列 (" ") に変換します。例: 12, AB,"",23p09udj。

次の Databricks ターゲットに基づいて、NULL 処理動作を考慮します。

- ファイルターゲット。Databricks Spark エンジンでは、すべての NULL 値を空の文字列として書き込みます。
- Azure SQL データウェアハウス Databricks Spark エンジンでは、空の文字列を文字列カラムに書き込むことができますが、空の文字列を文字列以外のカラムに書き込もうとしたときに、マッピングが型不一致で失敗します。

Databricks Spark エンジンが空の文字列を NULL 値に戻してターゲットに書き込めるようにするには、Databricks Spark 接続の詳細プロパティ `infaspark.flatfile.writer.nullValue=true` を設定します。

NULL に変換される予期しない値

Databricks Spark エンジンでは、フィールドに予期しない値が含まれている場合、次のシナリオに基づいて、同じレコードのすべてのフィールドに NULL 値を生成します。

- string データを numeric カラムに渡すなど、型不一致が発生した。
- bigint データ型、int データ型などで、データが範囲を超えている。

NULL 行をフィルタするには、フィルタトランスフォーメーションの使用を考慮します。

NULL に変換される日付/時刻値

Databricks Spark エンジンでは、日付/時刻値を書き込むときに、Developer tool のランタイム設定の「マッピング」プロパティで設定されている形式を使用します。ターゲットに渡した日付の形式が Developer tool に設定されている形式と一致しない場合は、Databricks Spark エンジンでは NULL 値を書き込みます。Developer tool のデフォルト値は MM/DD/YYYY HH24:MI:SS です。

double と decimal の変換

Databricks Spark エンジンでは、Azure ソースから読み取ったときに、double と decimal データ型を科学的表記に変換します。データを double または decimal に変換してターゲットに書き込むときに、15 を超える精度を削除し、15 桁の精度を維持します。

Hadoop でのファイルターゲット

Hadoop 環境で実行されるマッピングでは、複合ファイルとフラットファイルを処理できます。

大量のデータを書き込むには、複合ファイルターゲットを接続し、同じ形式とプロパティのファイルのディレクトリにデータを書き込むことができます。圧縮されたバイナリファイルを読み取れます。

Hadoop 環境では、次のストレージタイプの複合ファイルに書き込むことができます。

- Amazon Simple Storage Service (Amazon S3)
- Hadoop 分散ファイルシステム (HDFS)
- MapR File System (MapR-FS)

- Microsoft Azure Blob Storage (Azure Blob Storage)
- Microsoft Azure Data Lake Store (ADLS)

Amazon S3 の複合ファイルターゲット

Amazon S3 データオブジェクトにデータを書き込むには、PowerExchange for Amazon S3 接続を使用します。

次の表に、Hadoop 環境の Amazon S3 ストレージ内でマッピングが処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ^{1 2}	- Blaze - Spark
JSON	- Flat - 階層 ^{1 2}	- Blaze - Spark
ORC	- Flat	- Spark
Parquet	- Flat - 階層 ^{1 2}	- Blaze - Spark
¹ Blaze エンジンで実行するには、複合ファイルデータオブジェクトがデータプロセッサトランスフォーマーに接続されている必要があります。 ² Spark エンジンで実行するには、複合ファイル書き込み操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

ADLS の複合ファイルターゲット

ADLS データオブジェクトにデータを書き込むには、PowerExchange for HDFS または PowerExchange for Microsoft Azure Azure Data Lake Store 接続を使用できます。PowerExchange for Microsoft Azure Data Lake Store 接続を使用する場合は、ADLS に圧縮されたバイナリファイルを書き込むことはできません。

次の表に、Hadoop 環境の ADLS ストレージ内で PowerExchange for Microsoft Azure Data Lake Store 接続が処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ¹	Spark
JSON	- Flat - 階層 ¹	Spark
Parquet	- Flat - 階層 ¹	Spark
¹ Spark エンジンで実行するには、複合ファイル書き込み操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

次の表に、Hadoop 環境の ADLS ストレージ内で PowerExchange for HDFS 接続が処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ^{1 2}	- Blaze - Spark
JSON	- Flat ¹ - 階層 ^{1 2}	- Blaze - Spark
ORC	- Flat	- Spark
Parquet	- Flat - 階層 ^{1 2}	- Blaze - Spark
¹ Blaze エンジンで実行するには、複合ファイルデータオブジェクトがデータプロセッサトランスフォーマーに接続されている必要があります。 ² Spark エンジンで実行するには、複合ファイル書き込み操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

Azure Blob の複合ファイルターゲット

Azure Blob データオブジェクトにデータを書き込むには、PowerExchange for HDFS または PowerExchange for Microsoft Azure Blob Storage 接続を使用できます。

次の表に、Hadoop 環境の Azure Blob Storage 内でマッピングが処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ¹	Spark
JSON	- Flat - 階層 ¹	Spark
Parquet	- Flat - 階層 ¹	Spark
¹ Spark エンジンで実行するには、複合ファイル書き込み操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

次の表に、Hadoop 環境の Azure Blob Storage 内で PowerExchange for HDFS 接続が処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ^{1 2}	- Blaze - Spark
JSON	- Flat ¹ - 階層 ^{1 2}	- Blaze - Spark
ORC	- Flat	- Spark

ファイルタイプ	サポートされている形式	サポートされるエンジン
Parquet	- Flat - 階層 ^{1 2}	- Blaze - Spark
¹ Blaze エンジンで実行するには、複合ファイルデータオブジェクトがデータプロセッサトランスフォーマーに接続されている必要があります。 ² Spark エンジンで実行するには、複合ファイル書き込み操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

MapR-FS の複合ファイルターゲット

MapR-FS データオブジェクトからデータを読み取るには、PowerExchange for HDFS 接続を使用します。
 次の表に、Hadoop 環境の MapR-FS ストレージ内でマッピングが処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ^{1 2}	- Blaze - Spark
JSON	- Flat ¹ - 階層 ^{1 2}	- Blaze - Spark
ORC	- Flat	- Spark
Parquet	- Flat - 階層 ^{1 2}	- Blaze - Spark
¹ Blaze エンジンで実行するには、複合ファイルデータオブジェクトがデータプロセッサトランスフォーマーに接続されている必要があります。 ² Spark エンジンで実行するには、複合ファイル読み取り操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

HDFS の複合ファイルターゲット

HDFS データオブジェクトにデータを書き込むには、PowerExchange for HDFS 接続を使用します。
 次の表に、Hadoop 環境の HDFS ストレージ内でマッピングが処理できる複合ファイルを示します。

ファイルタイプ	サポートされている形式	サポートされるエンジン
Avro	- Flat - 階層 ^{1 2}	- Blaze - Spark
JSON	- Flat ¹ - 階層 ^{1 2}	- Blaze - Spark
ORC	- Flat	- Spark

ファイルタイプ	サポートされている形式	サポートされるエンジン
Parquet	- Flat - 階層 ^{1 2}	- Blaze - Spark
¹ Blaze エンジンで実行するには、複合ファイルデータオブジェクトがデータプロセッサトランスフォーマーに接続されている必要があります。 ² Spark エンジンで実行するには、複合ファイル書き込み操作がプロジェクトカラムに対して複合データ型として有効にされている必要があります。		

フラットファイルターゲット Hadoop 上

Hadoop 環境のフラットファイルターゲットについては、次のルールおよびガイドラインを考慮します。

- Hadoop 環境で実行されるマッピングは、ネイティブ環境のフラットファイルターゲットに書き込むことができます。
- データ統合サービスは、ターゲットファイルを切り詰め、データの書き込み前にファイルを拒否します。フラットファイルターゲットの使用時に、データをそのターゲットファイルに追加し、ファイルを拒否することはできません。
- データ統合サービスは、フラットファイルターゲットのファイル出力に書き込みます。マッピング内にフラットファイルターゲットがあるときは、データをコマンドに書き込むことはできません。

HDFS フラットファイルターゲットについては、次のルールおよびガイドラインを考慮します。

- データ統合サービスは、ターゲットファイルを切り詰め、データの書き込み前にファイルを拒否します。出力データを HDFS ターゲットファイルと許可ファイルに追加するには、HDFS ターゲットが存在する場合にデータを追加するように選択します。
拒否ファイルディレクトリがデータ統合サービスマシンにある場合は、データは拒否ファイルのみに追加されます。ディレクトリが Hadoop 環境にある場合は、拒否された行は上書きされます。
- HDFS ターゲットが存在する場合にデータを追加するように選択した場合は、データ統合サービスにより、マッピング実行 ID がターゲットファイルおよび拒否ファイルの名前に追加されます。
- マッピング内で HDFS フラットファイルターゲットを使用するときに、出力ファイルディレクトリとファイル名を含むフルパスを指定する必要があります。データ統合サービスは、マッピングを Hadoop 環境で実行すると、出力ディレクトリ内に複数の出力ファイルを生成する可能性があります。
- HDFS ターゲットは、リモートクラスタに置くことはできません。リモートクラスタは、マッピングで Hadoop 接続が参照するマシンのリモートにあるクラスタです。

Hadoop でのメッセージターゲット

Kafka メッセージターゲットに書き込むことができます。

Apache Kafka は Publish-Subscribe メッセージシステムです。Kafka は、ブローカと呼ばれる 1 つまたは複数のサーバーで構成されるクラスタとして動作します。Kafka ブローカは、データをメッセージの形式で処理し、メッセージをトピックにパブリッシュし、メッセージをトピックからサブスクライブしてから、メッセージを Kafka ターゲットに書き込みます。

Kafka データオブジェクト書き込み操作をターゲットとして使用できます。マッピングに高可用性を設定する場合は、Kafka クラスタの可用性が高くなっていることを確認してください。

Kafka クラスタ、Kafka ブローカ、パーティションの詳細については、
<https://kafka.apache.org/documentation> を参照してください。

テクニカルレビュー: Big Data Management での Kafka ターゲットへのバッチデータ書き込みサポートは、テクニカルレビューで利用できます。テクニカルレビュー機能はサポートされていますが、保証対象外で本番環境には対応していません。非本番環境でのみ使用することをお勧めします。

Hadoop のリレーショナルターゲット

Blaze エンジンでは、リレーショナルターゲットを含むマッピングを実行できます。

データ統合サービスは、マッピング前の SQL コマンドまたはマッピング後の SQL コマンドを、Hadoop 環境のリレーショナルターゲットに対して実行しません。Hadoop 環境のリレーショナルターゲットに、PreSQL プロパティまたは PostSQL プロパティを使用してマッピングを検証および実行することはできません。

データ統合サービスは、次のリレーショナルターゲットへの書き込みに複数のパーティションを使用できます。

- IBM DB2
- Oracle

Hadoop での Hive ターゲット

Hadoop 環境で実行中のマッピングは、Hive ターゲットに書き込むことができます。

Hive ターゲットは、内部テーブルまたは外部テーブルにできます。内部の Hive テーブルは Hive が管理し、管理対象テーブルとしても知られています。外部の Hive テーブルは、HDFS、Amazon S3、Microsoft Azure Blob Storage などの外部ソースが管理します。

Hadoop 環境で実行するマッピングに Hive ターゲットを設定する場合は、次の制限事項を考慮します。

- Hive ターゲット定義で Unicode 文字が使用されていると、マッピングの実行に失敗します。
- Hive ターゲットは、リモートクラスタに置くことはできません。リモートクラスタは、マッピングで Hadoop 接続が参照するマシンのリモートにあるクラスタです。
- パーティション化された Hive テーブルに動的ターゲットを設定すると、パーティションに使用される値はテーブルの最終カラムになります。テーブルに動的パーティションカラムがある場合、テーブルの最終カラムは動的パーティションカラムになります。パーティションに別のカラムを使用するには、テーブルの最終カラムに目的のカラムを移動します。テーブルに複数のパーティションカラムがある場合、動的パーティションの値は、アップストリームトランスフォーメーションの最終カラムから選択されます。必要に応じてカラムの順序を変更するには、式トランスフォーメーションを使用できます。

マッピングが Hive テーブルを作成または置き換える場合、マッピングが作成するテーブルのタイプは、マッピングの実行に使用するランタイムエンジンによって決まります。

次の表は、ランタイムエンジンごとのテーブルタイプを示します。

ランタイムエンジン	作成されるテーブルのタイプ
Blaze	MANAGED_TABLE
Spark	EXTERNAL_TABLE

バケット化およびパーティション化された内部または外部の Hive テーブルを切り詰めるように、マッピングを設計できます。

Spark エンジンまたは Blaze エンジンで実行するマッピングで、実行時に Hive テーブルを作成または置き換えるカスタムの DDL クエリを作成できます。ただし、Blaze エンジンの場合は、バッククォート (`) 文字を DDL クエリに使用できません。バッククォート文字は、クエリに特殊文字やキーワードを含める際に HiveQL で必要になります。

Spark エンジンは、バケット化された Hive ターゲットに書き込めます。Hive テーブルをバケット化またはパーティション化することで、データのシャッフルとソートが減り、パフォーマンスを向上させることができます。

PreSQL コマンドおよび PostSQL コマンド

Hive ターゲット向けに SQL コマンドを作成できます。挿入、更新、および削除などの SQL 文を Hive ターゲットで実行するために、SQL コマンドを実行できます。

PreSQL は、マッピングがソースから読み取る前に、Hive ターゲットに対して実行される SQL コマンドです。PostSQL は、マッピングがターゲットに書き込んだ後で、Hive ターゲットに対して実行される SQL コマンドです。

PreSQL と PostSQL は Spark エンジンで使用できます。データ統合サービスは、Hive ターゲットに対する PreSQL コマンドや PostSQL コマンドを検証しません。

マッピング前の SQL コマンド

PreSQL は、マッピングがソースから読み取る前に、Hive ターゲットに対して実行される SQL コマンドです。

例えば、マッピング内で Hive ターゲットを使用しているとします。Hive ターゲット内に格納されたデータには前日の古いレコードが含まれており、新しいレコードをターゲットに書き込むマッピングを実行する前に、Hive ターゲットの古いレコードを削除する必要があります。Hive ターゲットの古いレコードを削除するために、PreSQL コマンドを設定できます。

マッピング後の SQL コマンド

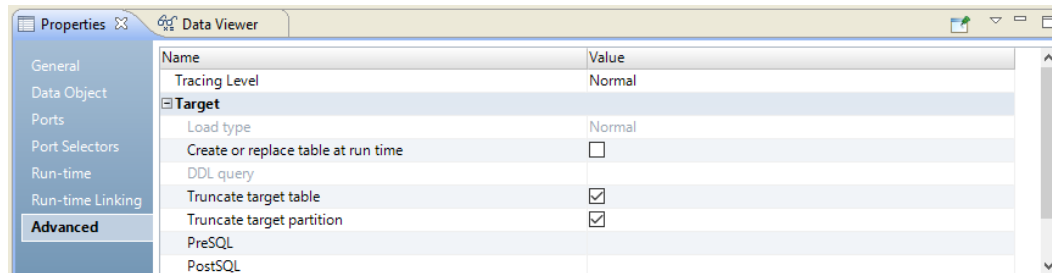
PostSQL は、マッピングがターゲットに書き込んだ後で、Hive ソースに対して実行される SQL コマンドです。

例えば、テスト環境で実行されるマッピングで、Hive ターゲットを使用しているとします。ターゲットへの書き込みをマッピングで実行した後、次のマッピングの実行で同じ Hive ターゲットを再びテストできるようにするために、Hive ターゲット内に格納されたレコードを削除する場合があります。Hive ターゲットのレコードを削除するために、PostSQL コマンドを設定できます。

Hive ターゲットの切り詰め

Hive ターゲットテーブルを切り詰めて、テーブルの内容を削除します。Hadoop 環境の内部および外部の Hive テーブルを切り詰めることができます。

Hive テーブルプロパティの Hive テーブルを切り詰めます。次の図は Hive テーブルプロパティを示します。



Hive テーブル全体を切り詰めるには、ターゲットテーブルを切り詰めるためのオプションを選択します。

トランスフォーメーションが入力データを受信する Hive テーブルでパーティションのみを切り詰めるには、ターゲットテーブルの切り詰めと、Hive ターゲットテーブルでのパーティションの切り詰めるを選択する必要があります。

注: 入力データがすべてのパーティションに影響を与えテーブル全体が切り詰められる原因になる可能性があるため、[ターゲットテーブルの切り詰め] オプションおよび [ターゲットパーティションの切り詰め] オプションの両方を選択する必要があります。

Hadoop 環境で実行するマッピングの Hive テーブルを切り詰める場合は、次の制限事項を考慮します。

- データ統合サービスでは、データが挿入される Hive ターゲットのパーティションを切り詰めることができます。
- Hive バージョン 0.7 の Hive テーブルにデータを上書きするためには、ターゲットテーブルを切り詰める必要があります。データ統合サービスは、Hive ターゲットにデータを書き込む際は、書き込み、更新オーバーライド、削除、挿入、およびアップデートストラテジのプロパティを無視します。

アップデートストラテジトランスフォーメーションを使用した Hive ターゲットの更新

Spark エンジンで実行されるマッピングの場合、アップデートストラテジタスクを実行するための Hive MERGE 文を使用できます。クエリで INSERT、UPDATE または DELETE の代わりに MERGE 文を使用すると、処理がより効率的になります。

Hive MERGE を使用するには、アップデートストラテジトランスフォーメーションの詳細プロパティで、当該オプションに **true** を選択します。

マッピングは Hive MERGE オプションを無視し、データ統合サービスは INSERT、UPDATE および DELETE を使用して、次のシナリオで操作を実行します。

- マッピングは Blaze エンジンで実行されます。
- Hadoop ディストリビューションで Hive MERGE が制限されています。

マッピングのログには、制限事項が結果に影響したかどうかを含む、操作の結果が含まれます。

パーティション化カラムまたはバケット化カラムに更新が影響した場合、カラムへの更新は省略されます。

注: Developer tool とデータ統合サービスは、この制限に対して検証を行いません。アップデートストラテジの式がこれらの制限事項に違反した場合、マッピングが予期しない結果になることがあります。

Blaze エンジン上の Hive ターゲットのルールとガイドライン

Blaze エンジンで実行する Informatica のマッピングに Hive ターゲットを含めることができます。

Blaze エンジンで実行するマッピングに Hive ターゲットを設定する場合は、次のルールとガイドラインを考慮します。

- Blaze エンジンで実行するマッピングには、ターゲットとしてパーティション化およびバケット化された Hive テーブルを設定できます。ただし、バケット化されたテーブルにデータを付加した場合、Blaze エンジンはバケット化されたターゲットのデータを上書きします。
- Blaze エンジンで実行されるマッピングは、ソート済みターゲットへの読み取りと書き込みを行えます。
- Blaze エンジンは、ロックが有効化された Hive テーブルをサポートします。
- Blaze エンジンは、Hive ターゲットテーブルを作成または置き換えることができます。
- Blaze エンジンで実行されるマッピングは、Hive ACID テーブルへの書き込みを行えます。Hive ACID テーブルに書き込むには、Hive ターゲットに接続されたアップデートストラテジ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必要があります。アップデートストラテジの式は、挿入のために各行にフラグを設定する必要があります。
- Blaze エンジンは、圧縮された Hive テーブルへの書き込みを行えます。圧縮された形式の Hive テーブルに書き込むには、TBLPROPERTIES 句を設定する必要があります。
- Blaze エンジン上の Hive ターゲットに書き込むマッピングの場合、アップデートまたはアップデート削除のストラテジプロパティを使用できません。

Hive テーブルとしての RCFile

Blaze エンジンは、Hive テーブルとして RCFile への読み書きを実行できます。ただし、Blaze エンジンがサポートするのは ColumnarSerDe SerDe のみです。Hortonworks では、RCFile のデフォルトの SerDe は LazyBinaryColumnarSerDe です。RCFile テーブルの読み書きを行うには、SerDe を `org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe` として指定することで、このテーブルを作成する必要があります。

以下に例を示します。

```
CREATE TABLE TEST_RCFILE
(id int, name string)
ROW FORMAT SERDE
'org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe' STORED AS RCFILE;
```

デフォルトの RCFile SerDe を Ambari または Cloudera マネージャから設定することもできます。

hive.default.rcfile.serde プロパティを `org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe` に設定します。

圧縮された Hive テーブル

Blaze エンジンは、圧縮された Hive テーブルへの読み書きを実行できます。ただし、圧縮された Hive テーブルからの読み取りと、圧縮形式の Hive テーブルへの書き込みのためには、TBLPROPERTIES 句を次のように設定する必要があります。

- テーブルの作成時に、テーブルプロパティを次のように設定します。
`TBLPROPERTIES ('property_name'='property_value')`
- テーブルがすでにある場合、次のテーブルプロパティを設定するようにテーブルを変更します。
`ALTER TABLE table_name SET TBLPROPERTIES ('property_name' = 'property_value');`

プロパティの名前と値は大文字と小文字が区別されません。ファイル形式に応じて、テーブルプロパティは異なる値を取得します。

次の表に、ファイル形式別のプロパティの名前と値の一覧を示します。

ファイル形式	テーブルプロパティの名前	テーブルプロパティの値
Avro	avro.compression	BZIP2、deflate、Snappy
ORC	orc.compress	Snappy、ZLIB
Parquet	parquet.compression	GZIP、Snappy
RCFile	rcfile.compression	Snappy、ZLIB
シーケンス	sequencefile.compression	BZIP2、GZIP、LZ4、Snappy
テキスト	text.compression	BZIP2、GZIP、LZ4、Snappy

注: Blaze エンジンでは、ORC 形式として格納された Hive ターゲットへの書き込み時に、デフォルトの ZLIB 圧縮形式でデータを書き込みません。圧縮形式でデータを書き込むには、このテーブルを変更し、TBLPROPERTIES 句が ORC ファイル形式に ZLIB または Snappy を使用するよう、この句を設定します。

次のテキストに、テーブルを作成し、テーブルを変更するためのコマンド例を示します。

- テーブルの作成は次のとおりです。

```
create table CBO_3T_JOINS_CUSTOMER_HIVE_SEQ_GZIP
(C_CUSTKEY DECIMAL(38,0), C_NAME STRING, C_ADDRESS STRING,
C_PHONE STRING, C_ACCTBAL DECIMAL(10,2),
C_MKTSEGMENT VARCHAR(10), C_COMMENT VARCHAR(117))
partitioned by (C_NATIONKEY DECIMAL(38,0))
TBLPROPERTIES ('sequencefile.compression'='gzip')
stored as SEQUENCEFILE;
```

- テーブルの変更は次のとおりです。

```
ALTER TABLE table_name
SET TBLPROPERTIES ('avro.compression'='BZIP2');
```

Hadoop の Sqoop ターゲット

Sqoop ターゲットは、Hadoop 環境のマッピングで有効です。

JDBC に準拠したデータベースを、Hadoop 環境で実行される Informatica マッピングの Sqoop ターゲットとして含めることができます。

例えば、次のターゲットを Sqoop マッピングに含めることができます。

- Aurora
- Greenplum
- IBM DB2
- IBM DB2 for z/OS
- Microsoft SQL Server
- Netezza
- Oracle
- Teradata

- Vertica

データを挿入できます。ターゲットのデータは更新または削除することはできません。更新の引数を設定した場合、Sqoop はそれらを無視します。

Sqoop ターゲットのルールとガイドライン

マッピングに Sqoop ターゲットを設定する場合は、次のルールとガイドラインを考慮します。

- カラム名またはテーブル名に特殊文字が含まれる場合、Sqoop のエクスポートプロセスは失敗します。
- Sqoop ターゲットに **【行順序を保持】** プロパティを設定した場合、データ統合サービスはこのプロパティを無視します。
- マッピングに Sqoop ソース、アグリゲータトランスフォーメーション、およびフラットファイルターゲットが含まれている場合、ターゲットの **【行順序を保持】** プロパティを無効化する必要があります。無効化していないと、マッピングは失敗します。
- Blaze エンジンで Sqoop マッピングを実行するときに、マッピングからターゲットポートを削除していないことを確認します。削除されていると、マッピングは失敗します。
- 非 NULL として定義されている Microsoft SQL Server カラムに NULL データをエクスポートすると、データ統合サービスは、不良ファイルを拒否したり、NULL データを書き込んだりする代わりに、Blaze エンジン上での Sqoop マッピングに失敗します。
- Oracle ターゲットに Sqoop を通じてデータを書き込み、Blaze または Spark エンジンでマッピングを実行した場合、Sqoop は所有者名の大文字と小文字を区別します。
- Sqoop では、JDBC 接続の **【ユーザー名】** と **【パスワード】** の各フィールドで設定した値が使用されます。JDBC 接続またはマッピングで `--username` 引数または `--password` 引数を設定すると、Sqoop は引数を無視します。データベースにアクセスするためのパスワードファイルを作成しても、Sqoop ではパスワードファイルが無視されます。
- Sqoop を使用して Vertica ターゲットにデータを書き込むときには、`--batch` 引数が必要です。

第 5 章

トランスフォーメーション

この章では、非ネイティブ環境でのトランスフォーメーションのサポートに関して説明します。

トランスフォーメーションの概要

ネイティブ環境と非ネイティブ環境との間で違いがあるため、非ネイティブ環境で特定のトランスフォーメーションのみが有効な場合や、制限付きで有効な場合があります。一部の関数、式、データ型、および変数のフィールドは、非ネイティブ環境で無効になります。

トランスフォーメーションやトランスフォーメーションの動作が、非ネイティブ環境で有効になるか、制限付きで有効になるかに影響する可能性がある、次の処理の違いを考慮します。

- 計算クラスタは、分散処理を使用し、異なるノードでデータを処理します。各ノードが、他のノードで処理されているデータにアクセスすることはありません。その結果、ランタイムエンジンではデータが生成された順序を判断できない場合があります。
- 非ネイティブ環境ではランタイムエンジンごとにマッピングロジックの処理を変えることができます。

次の表は、各トランスフォーメーションと、非ネイティブ環境の各エンジンに対するサポートを一覧表示しています。

トランスフォーメーション	サポートされるエンジン
この表に一覧表示されていないトランスフォーメーションは非ネイティブ環境ではサポートされません。	
アドレスバリデータ	- Blaze - Spark*
アグリゲータ	- Blaze - Spark - Databricks Spark
大文字小文字変換プログラム	- Blaze - Spark*
分類子	- Blaze - Spark*
比較	- Blaze - Spark*

トランスフォーメーション	サポートされるエンジン
統合	- Blaze - Spark*
データマスキング	- Blaze - Spark
データプロセッサ	- Blaze
ディシジョン	- Blaze - Spark*
式	- Blaze - Spark - Databricks Spark
フィルタ	- Blaze - Spark - Databricks Spark
Java	- Blaze - Spark
ジョイナ	- Blaze - Spark - Databricks Spark
キージェネレータ	- Blaze - Spark*
ラベラ	- Blaze - Spark*
ルックアップ	- Blaze - Spark - Databricks Spark
一致	- Blaze - Spark*
マージ	- Blaze - Spark*
ノーマライザ	- Blaze - Spark - Databricks Spark
パーサー	- Blaze - Spark*
Python	- Spark
ランク	- Blaze - Spark - Databricks Spark

トランスフォーメーション	サポートされるエンジン
ルーター	- Blaze - Spark - Databricks Spark
シーケンスジェネレータ	- Blaze - Spark*
ソーター	- Blaze - Spark - Databricks Spark
標準化	- Blaze - Spark*
共有体	- Blaze - Spark - Databricks Spark
アップデートストラテジ	- Blaze - Spark*
加重平均	- Blaze - Spark*
*Spark エンジン上の Big Data Streaming ではサポート対象外です。Big Data Streaming のトランスフォーメーションの詳細については、Informatica Big Data Streaming のユーザーガイドを参照してください。	

非ネイティブ環境でのアドレスバリデータトランスフォーメーション

非ネイティブ環境でのアドレスバリデータトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。サポートしません。

Blaze エンジンでのアドレスバリデータトランスフォーメーション

アドレスバリデータトランスフォーメーションのサポートには、次の制限があります。

- アドレスバリデータトランスフォーメーションでは、認証レポートは生成できません。
- アドレスバリデータトランスフォーメーションは、対話モードまたは提案リストモードで実行されるように構成されていると、検証に失敗します。

Spark エンジンでのアドレスバリデータトランスフォーメーション

アドレスバリデータトランスフォーメーションのサポートには、次の制限があります。

- アドレスバリデータトランスフォーメーションでは、認証レポートは生成できません。
- アドレスバリデータトランスフォーメーションは、対話モードまたは提案リストモードで実行されるように構成されていると、検証に失敗します。

非ネイティブ環境でのアグリゲータトランスフォーメーション

非ネイティブ環境でのアグリゲータトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。制限付きのサポート。

Blaze エンジンでのアグリゲータトランスフォーメーション

Blaze エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションに、ステートフル変数ポートが含まれる。
- 式のトランスフォーメーションに、サポートされていない関数が含まれる。

集計関数

アグリゲータトランスフォーメーションの式でポートを使用し、集計関数内でポートを使用しないと、ランタイムエンジンはポート内の任意の行を使用して式を処理することがあります。

ランタイムエンジンは、ポートの最後の行以外の行を使用することがあります。処理が分散されているために、ランタイムエンジンがポートの最後の行を特定できないことがあるからです。

データキャッシュの最適化

アグリゲータトランスフォーメーションのデータキャッシュは、可変長を使用して、アグリゲータトランスフォーメーションを介して渡されるバイナリデータ型と文字列データ型を格納するように最適化されます。この最適化は、8 MB までのレコードに対して有効化されます。レコードサイズが 8 MB を超える場合、可変長の最適化は無効になります。

アグリゲータトランスフォーメーションを介して渡されるデータのデータキャッシュ内の格納に可変長が使用される場合、アグリゲータトランスフォーメーションは、ランタイムマッピング内のアグリゲータトランスフォーメーションの前に挿入されるソートされた入力とパススルーのソータートランスフォーメーションを使用するように最適化されます。

ソータートランスフォーメーションを表示するには、最適化されたマッピングを表示するか、Blaze 検証環境内の実行プランを表示します。

データキャッシュの最適化中は、アグリゲータトランスフォーメーション用のそのデータキャッシュとインデックスキャッシュは「自動」に設定されます。ソータートランスフォーメーションのソーターキャッシュは、アグリゲータトランスフォーメーションのデータキャッシュと同じサイズに設定されます。ソーターキャッシュを設定するには、アグリゲータトランスフォーメーションのデータキャッシュのサイズを設定する必要があります。

Spark エンジンでのアグリゲータトランスフォーメーション

Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションに、ステートフル変数ポートが含まれる。
- 式のトランスフォーメーションに、サポートされていない関数が含まれる。

集計関数

アグリゲータトランスフォーメーションの式でポートを使用し、集計関数内でポートを使用しないと、ランタイムエンジンはポート内の任意の行を使用して式を処理することがあります。

ランタイムエンジンは、ポートの最後の行以外の行を使用することがあります。処理が分散されているために、ランタイムエンジンがポートの最後の行を特定できないことがあるからです。

データキャッシュの最適化

トランスフォーメーションのデータキャッシュを、可変長を使用してデータを格納するように最適化することはできません。

Databricks Spark エンジンでのアグリゲータトランスフォーメーション

Databricks Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションに、ステートフル変数ポートが含まれる。
- 式のトランスフォーメーションに、サポートされていない関数が含まれる。

集計関数

アグリゲータトランスフォーメーションの式でポートを使用し、集計関数内でポートを使用しないと、ランタイムエンジンはポート内の任意の行を使用して式を処理することがあります。

ランタイムエンジンは、ポートの最後の行以外の行を使用することがあります。処理が分散されているために、ランタイムエンジンがポートの最後の行を特定できないことがあるからです。

データキャッシュの最適化

トランスフォーメーションのデータキャッシュを、可変長を使用してデータを格納するように最適化することはできません。

非ネイティブ環境での大文字小文字変換トランスフォーマー

非ネイティブ環境での大文字小文字変換トランスフォーマー処理は、そのトランスフォーマーを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限なくサポートされます。
- Databricks Spark エンジン。サポートしません。

非ネイティブ環境での分類子トランスフォーマー

非ネイティブ環境での分類子トランスフォーマー処理は、そのトランスフォーマーを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限なくサポートされます。
- Databricks Spark エンジン。サポートしません。

非ネイティブ環境での比較トランスフォーマー

非ネイティブ環境での比較トランスフォーマー処理は、そのトランスフォーマーを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限なくサポートされます。
- Databricks Spark エンジン。サポートしません。

非ネイティブ環境での統合トランスフォーメーション

非ネイティブ環境でのアドレスバリデータトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。サポートしません。

Blaze エンジンでの統合トランスフォーメーション

統合トランスフォーメーションのサポートには、次の制限があります。

- このトランスフォーメーションは、それぞれの環境でレコードを異なる順序で処理する場合があります。
- このトランスフォーメーションは、それぞれの環境で異なるレコードを存続レコードとして特定する場合があります。

Spark エンジンでの統合トランスフォーメーション

統合トランスフォーメーションのサポートには、次の制限があります。

- このトランスフォーメーションは、それぞれの環境でレコードを異なる順序で処理する場合があります。
- このトランスフォーメーションは、それぞれの環境で異なるレコードを存続レコードとして特定する場合があります。

非ネイティブ環境でのデータマスキングトランスフォーメーション

非ネイティブ環境でのデータマスキングトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。サポートしません。

Blaze エンジンでのデータマスキングトランスフォーメーション

データマスキングトランスフォーメーションのサポートには、次の制限があります。

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションが、再現可能な式のマスキングに対して設定されている。

- トランスフォーメーションが、再現可能な一意の置換に対して設定されている。

Blaze エンジンでは次のマスキング方法を使用できます。

クレジットカード
Email
式
IP アドレス
キー
電話番号
ランダム
SIN
SSN
トークン化
URL
ランダムな置換
再現可能な置換
ランダムな置換による依存
再現可能な置換による依存

Spark エンジンでのデータマスキングトランスフォーメーション

データマスキングトランスフォーメーションのサポートには、次の制限があります。

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションが、再現可能な式のマスキングに対して設定されている。
- トランスフォーメーションが、再現可能な一意の置換に対して設定されている。

Spark エンジンでは次のマスキング方法を使用できます。

クレジットカード
Email
式
IP アドレス
キー
電話番号
ランダム
SIN
SSN
トークン化
URL
ランダムな置換
再現可能な置換
ランダムな置換による依存

再現可能な置換による依存

データマスキングトランスフォーメーションのパフォーマンスを最適化するには、Hadoop 接続の次の Spark エンジン設定プロパティを設定します。

`spark.executor.cores`

各実行プログラムプロセスが Spark エンジンでタスクレットを実行するために使用するコアの数を示します。

`spark.executor.cores=1` に設定します。

`spark.executor.instances`

各実行プログラムプロセスが Spark エンジンでタスクレットを実行するために使用するインスタンスの数を示します。

`spark.executor.instances=1` に設定します。

`spark.executor.memory`

各実行プログラムプロセスが Spark エンジンでタスクレットを実行するために使用するメモリの量を示します。

`spark.executor.memory=3G` に設定します。

非ネイティブ環境でのデータプロセッサトランスフォーメーション

非ネイティブ環境でのデータプロセッサトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。サポートしません。
- Databricks Spark エンジン。サポートしません。

Blaze エンジンでのデータプロセッサトランスフォーメーション

データプロセッサトランスフォーメーションのサポートには、次の制限があります。

- マッピング検証は、トランスフォーメーションのデータプロセッサモードが **【入力マッピング】** または **【入力マッピングとサービス】** に設定されていると失敗します。

非ネイティブ環境でのディシジョントランスフォーメーション

非ネイティブ環境でのディシジョントランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限付きのサポート。ディシジョントランスフォーメーションプロパティはパーティション化可能に設定する必要があります。
- Databricks Spark エンジン。サポートしません。

非ネイティブ環境での式トランスフォーメーション

非ネイティブ環境での式トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。制限付きのサポート。

Blaze エンジンでの式トランスフォーメーション

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションに、ステートフル変数ポートが含まれる。
- 式のトランスフォーメーションに、サポートされていない関数が含まれる。

ユーザー定義関数を持つ式トランスフォーメーションは、関数に例外エラーがある行に対して NULL 値を返します。

Spark エンジンでの式トランスフォーメーション

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションに、ステートフル変数ポートが含まれる。
- 式のトランスフォーメーションに、サポートされていない関数が含まれる。

注: 式の結果が、ゼロによる除算や負の数値の SQRT などの、数値エラーになった場合は、NULL 値を返し、出力に行は表示されません。ネイティブ環境では、式は infinite または NaN 値を返します。

Databricks Spark エンジンでの式トランスフォーメーション

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションに、ステートフル変数ポートが含まれる。

- 式のトランスフォーメーションに、サポートされていない関数が含まれる。

注: 式の結果が、ゼロによる除算や負の数値の SQRT などの、数値エラーになった場合は、NULL 値を返し、出力には表示されません。ネイティブ環境では、式は infinite または NaN 値を返します。

非ネイティブ環境でのフィルタトランスフォーメーション

非ネイティブ環境でのフィルタトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限なくサポートされます。
- Databricks Spark エンジン。制限なくサポートされます。

Blaze エンジンでのフィルタトランスフォーメーション

Hive ソースのパーティション化されたカラムでのフィルタ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、Blaze エンジンはフィルタ条件を満たすデータを持つパーティションのみを読み取れます。フィルタを Hive ソースにプッシュするには、フィルタトランスフォーメーションがソースの後のマッピング内の次のトランスフォーメーションになるように設定します。

非ネイティブ環境での Java トランスフォーメーション

非ネイティブ環境での Java トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。サポートしません。

Blaze エンジンでの Java 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内で外部.jar ファイルを使用するには、次の手順を実行します。

1. 外部.jar ファイルを、データ統合サービスマシン内の<Informatic installation directory>/services/shared/jars にある Informatica のインストールディレクトリにコピーします。次に、データ統合サービスをリサイクルします。

2. Developer tool をホストするマシンが、Java トランスフォーメーションを含むマッピングを開発して実行する場所です。
 - a. 外部.jar ファイルを、ローカルマシンのディレクトリにコピーします。
 - b. Java トランスフォーメーションを、そのローカル.jar ファイルをポイントするインポート文を含めるように編集します。
 - c. Java トランスフォーメーションのクラスパスを更新します。
 - d. トランスフォーメーションをコンパイルします。

Spark エンジンでの Java トランスフォーメーション

階層データ型を処理するために、複合データを使用できます。

Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

一般的な制限

Java トランスフォーメーションのサポートには、Spark エンジンでは次の制限があります。

- トランスフォーメーションロジックを Hadoop にプッシュするときに、トランスフォーメーション内の Java コードは、出力を標準出力に書き込むことはできません。Java コードは、標準エラーに出力を書き込むことはできます。このエラーはログファイルに表示されます。
- 日時の値の場合、Spark エンジンで最大マイクロ秒の精度がサポートされます。日時の値にナノ秒が含まれている場合、後続の桁は切り詰められます。

パーティション化

Java トランスフォーメーションには、パーティション化と一緒に使用される場合に次の制限があります。

- パーティション化が可能なプロパティが、Java トランスフォーメーションで有効である必要があります。トランスフォーメーションは 1 つのパーティション内では実行できません。
- 次の制限がトランスフォーメーション範囲プロパティに適用されます。
 - トランスフォーメーション範囲の値トランザクションは有効ではありません。
 - パーティションキーの入力ポートを有効にした場合、トランスフォーメーション範囲は [すべての入力] に設定される必要があります。
 - ステートレスは、トランスフォーメーション範囲が行の場合に有効にされる必要があります。

マッピング検証

マッピング検証は、次の場合に失敗します。

- Java トランスフォーメーション内の式から未接続のルックアップトランスフォーメーションを参照した。
- 複合データ型のポートをパーティションまたはソートキーとして選択した。
- 日時でナノ秒処理を有効にし、Java トランスフォーメーションに、日時型の要素を持つ複合データ型のポートが含まれる。例えば、array<data/time>型のポートは、日時でナノ秒処理を有効にした場合に有効ではありません。
- 高精度を有効にしているときに、Java トランスフォーメーションに含まれている式が小数位ポートまたは decimal データ型の要素を持つ複合ポートを使用し、ポートが演算子で使用されている場合。
例えば、トランスフォーメーションに小数位ポート decimal_port が含まれ、式 decimal_port + 1 を使用する場合、検証エラーが発生します。

マッピングは、次の場合に失敗します。

- Java トランスフォーメーションに小数位ポートまたは decimal データ型の要素を持つ複合ポートが含まれているときに、Java トランスフォーメーションとマッピングが異なる精度モードを使用する場合。

マッピングで高精度が有効にされている場合でも、マッピングに decimal データ型の要素を持つ複合ポートが含まれているときや、マッピングがストリーミングマッピングであるときなど、マッピングがデータを低精度モードで処理することがあります。Java トランスフォーメーションとマッピングの両方で高精度が有効にされていても、マッピングが低精度モードでデータを処理した場合、マッピングは失敗します。

外部.jar ファイルの使用

Java トランスフォーメーション内で外部.jar ファイルを使用するには、次の手順を実行します。

1. 外部.jar ファイルを、データ統合サービスマシン内の次の場所にある Informatica のインストールディレクトリにコピーします。
<Informatic installation directory>/services/shared/jars
2. データ統合サービスをリサイクルします。
3. Developer tool をホストするマシンが、Java トランスフォーメーションを含むマッピングを開発して実行する場所です。
 - a. 外部.jar ファイルを、ローカルマシンのディレクトリにコピーします。
 - b. Java トランスフォーメーションを、そのローカル.jar ファイルをポイントするインポート文を含めるように編集します。
 - c. Java トランスフォーメーションのクラスパスを更新します。
 - d. トランスフォーメーションをコンパイルします。

非ネイティブ環境でのジョイナトランスフォーメーション

非ネイティブ環境でのジョイナトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。制限付きのサポート。

Blaze エンジンでのジョイナトランスフォーメーション

マッピング検証は、次の場合に失敗します。

- 不等な結合とマップ側の結合を含むトランスフォーメーションが無効。
- ジョイナトランスフォーメーションの式が、未接続のルックアップトランスフォーメーションを参照する。

ジョイナトランスフォーメーションが、詳細外部結合または完全な外部結合用に設定されると、マップ側の結合は無効になります。

Spark エンジンでのジョイナトランスフォーメーション

マッピング検証は、次の場合に失敗します。

- 大文字小文字の区別が無効。
- 結合条件がバイナリデータ型、またはバイナリ式が含まれる。

Databricks Spark エンジンでのジョイナトランスフォーメーション

マッピング検証は、次の場合に失敗します。

- 大文字小文字の区別が無効。
- 結合条件がバイナリデータ型、またはバイナリ式が含まれる。

非ネイティブ環境でのキージェネレータトランスフォーメーション

非ネイティブ環境でのキージェネレータトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限なくサポートされます。
- Databricks Spark エンジン。サポートしません。

非ネイティブ環境でのラベラトランスフォーメーション

非ネイティブ環境でのラベラトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限なくサポートされます。
- Databricks Spark エンジン。サポートしません。

非ネイティブ環境でのルックアップトランスフォーメーション

非ネイティブ環境でのルックアップトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。制限付きのサポート。

Blaze エンジンでのルックアップトランスフォーメーション

Blaze エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証は、次の場合に失敗します。

- キャッシュが共有、名前付き、動的、キャッシュを使用しないように設定された。キャッシュは静的キャッシュである必要があります。

マッピング内でルックアップトランスフォーメーションとして Sqoop を使用するデータオブジェクトを追加する場合、データ統合サービスは、Sqoop を介してマッピングを実行しません。JDBC を介してマッピングを実行します。

Spark エンジンでのルックアップトランスフォーメーション

Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証

マッピング検証は、次の場合に失敗します。

- 大文字小文字の区別が無効。
- ルックアップトランスフォーメーション内のルックアップ条件に、バイナリデータ型が含まれる。
- キャッシュが共有、名前付き、動的、キャッシュを使用しないように設定された。キャッシュは静的キャッシュである必要があります。

マッピングは、次の場合に失敗します。

- トランスフォーメーションが未接続であり、ジョイナトランスフォーメーションまたは Java トランスフォーメーションと一緒に使用された。

複数一致

複数一致で最初、最後、または任意の値を返すように選択すると、ルックアップトランスフォーメーションは任意の値を返します。

トランスフォーメーションが複数一致でエラーを報告するように設定した場合、Spark エンジンは重複した行を削除し、それらの行をログに含めません。

Databricks Spark エンジンでのルックアップトランスフォーメーション

Databricks Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証

マッピング検証は、次の場合に失敗します。

- 大文字小文字の区別が無効。
- ルックアップトランスフォーメーション内のルックアップ条件に、バイナリデータ型が含まれる。
- キャッシュが共有、名前付き、動的、キャッシュを使用しないように設定された。キャッシュは静的キャッシュである必要があります。
- ルックアップソースは Microsoft Azure SQL Data Warehouse ではありません。

マッピングは、次の場合に失敗します。

- トランスフォーメーションが未接続であり、ジョイナトランスフォーメーションと一緒に使用された。

複数一致

複数一致で最初、最後、または任意の値を返すように選択すると、ルックアップトランスフォーメーションは任意の値を返します。

トランスフォーメーションが複数一致でエラーを報告するように設定した場合、Spark エンジンは重複した行を削除し、それらの行をログに含めません。

非ネイティブ環境での一致トランスフォーメーション

非ネイティブ環境での一致トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。サポートしません。

Blaze エンジンでの一致トランスフォーメーション

マッピング検証は、一致トランスフォーメーションが ID インデックスデータをデータベーステーブルに書き込むように設定されていると失敗します。

一致トランスフォーメーションでは、クラスタ ID 値はネイティブ環境と非ネイティブ環境で異なって生成されます。非ネイティブ環境では、トランスフォーメーションはグループ ID 値をクラスタ ID に追加します。

Spark エンジンでの一致トランスフォーメーション

マッピング検証は、一致トランスフォーメーションが ID インデックスデータをデータベーステーブルに書き込むように設定されていると失敗します。

一致トランスフォーメーションでは、クラスタ ID 値はネイティブ環境と非ネイティブ環境で異なって生成されます。非ネイティブ環境では、トランスフォーメーションはグループ ID 値をクラスタ ID に追加します。

非ネイティブ環境でのマージトランスフォーメーション

非ネイティブ環境でのマージトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限なくサポートされます。
- Databricks Spark エンジン。サポートしません。

非ネイティブ環境でのノーマライザトランスフォーメーション

非ネイティブ環境でのノーマライザトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限なくサポートされます。
- Databricks Spark エンジン。制限なくサポートされます。

非ネイティブ環境でのパーサトランスフォーメーション

非ネイティブ環境でのパーサトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限なくサポートされます。
- Databricks Spark エンジン。サポートしません。

非ネイティブ環境での Python トランスフォーメーション

非ネイティブ環境での Python トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。サポートしません。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。サポートしません。

Spark エンジンでの Python トランスフォーメーション

マッピング検証は、ユーザー定義のデフォルト値が出力ポートに割り当てられた場合に失敗します。

次のような場合に、マッピングが失敗します。

- 出力ポートに Python コードで値が割り当てられていない。
- 対応する入力ポートと出力ポート内のデータ型が同じでなく、Python コードで入力ポートのデータ型が出力ポートのデータ型に変換されない。
- Python トランスフォーメーションに小数位ポートが含まれ、マッピングで高精度が有効にされている。

注: データ統合サービスは Python コードを検証できません。

非ネイティブ環境での ランク トランスフォーメーション

非ネイティブ環境での ランク トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。制限付きのサポート。

Blaze エンジンでの ランク トランスフォーメーション

Blaze エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

ランク トランスフォーメーションのデータキャッシュは、可変長を使用して、ランク トランスフォーメーションを介して渡されるバイナリデータ型と文字列データ型を格納するように最適化されます。この最適化は、8 MB までのレコードに対して有効化されます。レコードサイズが 8 MB を超える場合、可変長の最適化は無効になります。

可変長が、ランク トランスフォーメーションを介して渡されるデータのデータキャッシュ内の格納に使用される場合、ランク トランスフォーメーションは、ランタイムマッピング内にランク トランスフォーメーション前

に挿入されるソートされた入力とパススルーのソータートランスフォーメーションを使用するように最適化されます。

ソータートランスフォーメーションを表示するには、最適化されたマッピングを表示するか、Blaze 検証環境内の実行プランを表示します。

データキャッシュの最適化中は、ランクトランスフォーメーション用のそのデータキャッシュとインデックスキャッシュは「自動」に設定されます。ソータートランスフォーメーションのソーターキャッシュは、ランクトランスフォーメーションのデータキャッシュと同じサイズに設定されます。ソーターキャッシュを設定するには、ランクトランスフォーメーションのデータキャッシュのサイズを設定する必要があります。

Spark エンジンでのランクトランスフォーメーション

Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証

マッピング検証は、次の場合に失敗します。

- 大文字小文字の区別が無効。
- ランクポートのデータ型がバイナリ型。

データキャッシュの最適化

トランスフォーメーションのデータキャッシュを、可変長を使用してデータを格納するように最適化することはできません。

Databricks Spark エンジンでのランクトランスフォーメーション

Databricks Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証

マッピング検証は、次の場合に失敗します。

- 大文字小文字の区別が無効。
- ランクポートのデータ型がバイナリ型。

データキャッシュの最適化

トランスフォーメーションのデータキャッシュを、可変長を使用してデータを格納するように最適化することはできません。

非ネイティブ環境でのルータートランスフォーメーション

非ネイティブ環境でのルータートランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限なくサポートされます。

- Databricks Spark エンジン。制限なくサポートされます。

シーケンスジェネレータトランスフォーメーション 非ネイティブ環境で

非ネイティブ環境でのシーケンスジェネレータトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。サポートしません。

Blaze エンジンでのシーケンスジェネレータトランスフォーメーション

シーケンスジェネレータトランスフォーメーションを使用したマッピングでは、次の条件が当てはまる場合に大量のリソースが消費されます。

- トランスフォーメーションの【行順序を保持】プロパティを *true* に設定した。
- マッピングが単一パーティションで実行されている。

Spark エンジンでのシーケンスジェネレータトランスフォーメーション

シーケンスジェネレータトランスフォーメーションは、出力データの行順序を保持しません。トランスフォーメーションの【行順序を保持】プロパティを有効にした場合、データ統合サービスはこのプロパティを無視します。

非ネイティブ環境でのソータートランスフォーメーション

非ネイティブ環境でのソータートランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。制限付きのサポート。

Blaze エンジンでのソータートランスフォーメーション

Blaze エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証

マッピング検証は、次の場合に失敗します。

- ターゲットデータが行の順序を維持するように設定されており、ソータートランスフォーメーションがフラットファイルターゲットと直接接続されていない。

並列ソート

データ統合サービスでは、次の制限付きで、並列ソートが有効になります。

- マッピング内のソータートランスフォーメーションとターゲットの間に別のトランスフォーメーションが含まれていない。
- ソータートランスフォーメーションとターゲットの間で、ソートキーのデータ型が変わらない。
- ソータートランスフォーメーションの各ソートキーは、ターゲットのカラムにリンクされている必要がある。

グローバルソート

Blaze エンジンは、次の条件が満たされたときにグローバルソートを実行できます。

- ソータートランスフォーメーションがフラットファイルターゲットと直接接続されている。
- ターゲットが行の順序を維持するように設定されている。
- ソートキーがバイナリデータ型ではない。

満たされない条件がある場合、Blaze エンジンはローカルソートを実行します。

データキャッシュの最適化

アグリゲータデータキャッシュまたはランクデータキャッシュを最適化するために、ソータートランスフォーメーションがアグリゲータトランスフォーメーションまたはランクトランスフォーメーションの前に挿入されている場合、ソーターキャッシュのサイズは、アグリゲータトランスフォーメーションまたはランクトランスフォーメーションのデータキャッシュと同じにする必要があります。ソーターキャッシュを設定するには、アグリゲータトランスフォーメーションまたはランクトランスフォーメーションのデータキャッシュのサイズを設定する必要があります。

Spark エンジンでのソータートランスフォーメーション

Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピングの検証

マッピング検証は、大文字小文字の区別が無効な場合に失敗します。

データ統合サービスは、次の場合に、警告をログに記録し、ソータートランスフォーメーションを無視します。

- ターゲットとソータートランスフォーメーションのソートキーの間に型の不一致がある。
- トランスフォーメーションに、ターゲットに接続されていないソートキーが含まれる。
- 書き込みトランスフォーメーションが行の順序を維持するように設定されていない。
- トランスフォーメーションが書き込みトランスフォーメーションからの直接のアップストリームではない。

NULL 値

データ統合サービスは、NULL 値を最高値として扱うようにトランスフォーメーションが設定されている場合でも、NULL 値を最低値として扱います。

データキャッシュの最適化

ソーターキャッシュを、可変長を使用してデータを格納するように最適化することはできません。

並列ソート

データ統合サービスでは、次の制限付きで、並列ソートが有効になります。

- マッピング内のソータートランスフォーメーションとターゲットの間に別のトランスフォーメーションが含まれていない。
- ソータートランスフォーメーションとターゲットの間で、ソートキーのデータ型が変わらない。
- ソータートランスフォーメーションの各ソートキーは、ターゲットのカラムにリンクされている必要がある。

Databricks Spark エンジンでのソータートランスフォーメーション

Databricks Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピングの検証

マッピング検証は、大文字小文字の区別が無効な場合に失敗します。

データ統合サービスは、次の場合に、警告をログに記録し、ソータートランスフォーメーションを無視します。

- ターゲットとソータートランスフォーメーションのソートキーの間に型の不一致がある。
- トランスフォーメーションに、ターゲットに接続されていないソートキーが含まれる。
- 書き込みトランスフォーメーションが行の順序を維持するように設定されていない。
- トランスフォーメーションが書き込みトランスフォーメーションからの直接のアップストリームではない。

NULL 値

データ統合サービスは、NULL 値を最高値として扱うようにトランスフォーメーションが設定されている場合でも、NULL 値を最低値として扱います。

データキャッシュの最適化

ソーターキャッシュを、可変長を使用してデータを格納するように最適化することはできません。

並列ソート

データ統合サービスでは、次の制限付きで、並列ソートが有効になります。

- マッピング内のソータートランスフォーメーションとターゲットの間に別のトランスフォーメーションが含まれていない。
- ソータートランスフォーメーションとターゲットの間で、ソートキーのデータ型が変わらない。
- ソータートランスフォーメーションの各ソートキーは、ターゲットのカラムにリンクされている必要がある。

非ネイティブ環境での標準化トランスフォーメーション

非ネイティブ環境での標準化トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限なくサポートされます。
- Databricks Spark エンジン。サポートしません。

非ネイティブ環境での共有体トランスフォーメーション

非ネイティブ環境での共有体トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限なくサポートされます。
- Databricks Spark エンジン。制限なくサポートされます。

非ネイティブ環境でのアップデートストラテジトランスフォーメーション

非ネイティブ環境でのアップデートストラテジトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。制限付きのサポート。
- Databricks Spark エンジン。サポートしません。

注: アップデートストラテジトランスフォーメーションは、Hive ACID をサポートする Hadoop ディストリビューションでのみサポートされます。

Blaze エンジンでのアップデートストラテジトランスフォーメーション

Hive ACID をサポートする Hadoop ディストリビューションで、アップデートストラテジトランスフォーメーションを使用できます。

Blaze エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

一般的な制限

アップデートストラテジトランスフォーメーションが、同じプライマリー値に対して複数の行の更新を受け取った場合、トランスフォーメーションは 1 つのランダムな行を選択してターゲットを更新します。

複数のアップデートストラテジトランスフォーメーションが、同じターゲットの異なるインスタンスに書き込む場合、ターゲットデータが予測できなくなる場合があります。

Blaze エンジンは、削除、更新、挿入の順番で操作を実行します。アップデートストラテジトランスフォーメーションが受け取った順番では行を処理しません。

Hive ターゲットは、更新を常に更新操作として実行します。Hive ターゲットは、[更新しない場合は挿入] または [挿入時に更新] をサポートしません。

マッピング検証とコンパイル検証

マッピング検証は、次の場合に失敗します。

- アップデートストラテジトランスフォーメーションが、複数のターゲットに接続されている。
- アップデートストラテジトランスフォーメーションが、ターゲットの直前に配置されていない。
- アップデートストラテジターゲットが Hive ターゲットではない。
- アップデートストラテジトランスフォーメーションのターゲットが外部の ACID テーブルである。
- ターゲットにプライマリーキーが含まれていない。
- 実行時にターゲットテーブルを切り詰めるための Hive ターゲットプロパティが有効になっている。
- 実行時にターゲットテーブルを作成または置き換えるための Hive ターゲットプロパティが有効になっている。

マッピングは、次の場合に失敗します。

- ターゲットが ORC パケット化されていない。
- 変更後の Hive ターゲットで、実際のテーブルより行が少なくなっている。

次の場合に、コンパイル検証エラーが発生し、マッピングの実行が停止されます。

- Hive バージョンが 0.14 より古い。
- ターゲットテーブルがトランザクションに対して有効化されていない。

Hive ターゲットテーブルの使用

アップデートストラテジトランスフォーメーションと一緒に Hive ターゲットテーブルを使用するには、Hive のデータ定義言語の句 TBLPROPERTIES ("transactional"="true") を使用して、Hive ターゲットテーブルを作成する必要があります。

Hive ターゲットと一緒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使用するには、Hadoop 接続に関連付けられた hive-site.xml 設定セットに次のプロパティが設定されていることを確認します。

```
hive.support.concurrency      true
hive.enforce.bucketing       true
hive.exec.dynamic.partition.mode nonstrict
hive.txn.manager              org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
hive.compactor.initiator.on   true
hive.compactor.worker.threads 1
```

Spark エンジンでのアップデートストラテジトランスフォーメーション

Hive ACID をサポートする Hadoop ディストリビューションで、アップデートストラテジトランスフォーメーションを使用できます。

アップデートストラテジトランスフォーメーションを使用して、マッピングの結果を JDBC 準拠のリレーショナルターゲットに書き込むこともできます。

Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

Hive ターゲットの一般的な制限

アップデートストラテジトランスフォーメーションは、ターゲットが Hive テーブルまたは JDBC 準拠のテーブルであるときは、拒否された行を次のトランスフォーメーションに転送しません。

アップデートストラテジトランスフォーメーションが、同じプライマリキー値に対して複数の行の更新を受け取った場合、トランスフォーメーションは 1 つのランダムな行を選択してターゲットを更新します。

複数のアップデートストラテジトランスフォーメーションが、同じターゲットの異なるインスタンスに書き込む場合、ターゲットデータが予測できなくなる場合があります。

Spark エンジンでマッピングを実行する場合、Hive Merge を使用するオプションを選択できます。このオプションには、次の制限事項があります。

- マッピングを Blaze または Hive で実行する場合、Hive MERGE オプションは無視される。データ統合サービスは、以前の実装でトランスフォーメーションを実行する。
- 削除または更新の単一行は、ターゲットの複数行とは一致できない。マッピングがこの制限に違反すると、マッピングはランタイムエラーで失敗する。
- アップデートストラテジトランスフォーメーションが、パーティション化カラムまたはバケット化カラムを更新するように設定されている場合、マッピングは Hive MERGE オプションを無視する。また、マッピングはカラムを更新しない。

注: Developer tool とデータ統合サービスは、これらの制限に対して検証を行いません。式またはマッピングがこれらの制限事項に違反している場合、マッピングが実行される可能性はありますが、期待どおりの結果にはなりません。

Hive ターゲットは、更新を常に更新操作として実行します。Hive ターゲットは、[更新しない場合は挿入] または [挿入時に更新] をサポートしません。

マッピング検証

マッピング検証は、次の場合に失敗します。

- アップデートストラテジトランスフォーメーションが、複数のターゲットに接続されている。
- アップデートストラテジトランスフォーメーションが、ターゲットの直前に配置されていない。
- アップデートストラテジトランスフォーメーションのターゲットが外部の ACID テーブルである。
- ターゲットに接続されたプライマリキーが含まれていない。
- 実行時の Hive またはリレーショナルターゲットテーブルの切り詰めを有効にするプロパティが選択されている。
- 実行時の Hive またはリレーショナルターゲットテーブルについて次のターゲットストラテジの 1 つが選択されている。
 - ターゲットテーブルの作成または置換
 - ApplyNewColumns

- ApplyNewSchema
- 失敗

ターゲットが Hive ターゲットのとき、マッピングは次の場合に失敗します。

- ターゲットテーブルがトランザクションに対して有効化されていない。
- ターゲットが ORC バケット化されていない。

Hive ターゲットテーブルの使用

アップデートストラテジトランスフォーメーションと一緒に Hive ターゲットテーブルを使用するには、Hive のデータ定義言語の句 TBLPROPERTIES ("transactional"="true")を使用して、Hive ターゲットテーブルを作成する必要があります。

Hive ターゲットと一緒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使用するには、Hadoop 接続に関連付けられた hive-site.xml 設定セットに次のプロパティが設定されていることを確認します。

```
hive.support.concurrency      true
hive.enforce.bucketing       true
hive.exec.dynamic.partition.mode nonstrict
hive.txn.manager              org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
hive.compactor.initiator.on   true
hive.compactor.worker.threads 1
```

JDBC ターゲットテーブルの使用

マッピングのアップデートストラテジトランスフォーメーションの JDBC 準拠のリレーショナルデータベースターゲットを設定して、Spark エンジンで実行することができます。

非ネイティブ環境での加重平均トランスフォーメーション

非ネイティブ環境での加重平均トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。制限なくサポートされます。
- Databricks Spark エンジン。サポートしません。

第 6 章

クラスタワークフロー

この章では、以下の項目について説明します。

- [クラスタワークフローの概要, 112 ページ](#)
- [クラスタワークフローのコンポーネント, 113 ページ](#)
- [クラスタワークフロープロセス, 114 ページ](#)
- [クラスタの作成タスクのプロパティ, 115 ページ](#)
- [マッピングタスクのプロパティ, 123 ページ](#)
- [クラスタの削除タスクの追加, 123 ページ](#)
- [ワークフローのデプロイと実行, 124 ページ](#)

クラスタワークフローの概要

ワークフローは、マッピングやその他のタスクをクラウドプラットフォームクラスタ上で実行するクラスタを作成するために実行できます。

Hadoop 環境では、Amazon EMR または Microsoft Azure HDInsight クラウドプラットフォームで実行するクラスタワークフローを作成します。Databricks 環境では、Databricks クラウドプラットフォームで実行するクラスタワークフローを作成します。

クラスタワークフローでは、データ統合サービスとクラウドプラットフォームとの間の通信を可能にする、クラウドプロビジョニング設定やクラスタ接続などのその他の要素が使用されます。

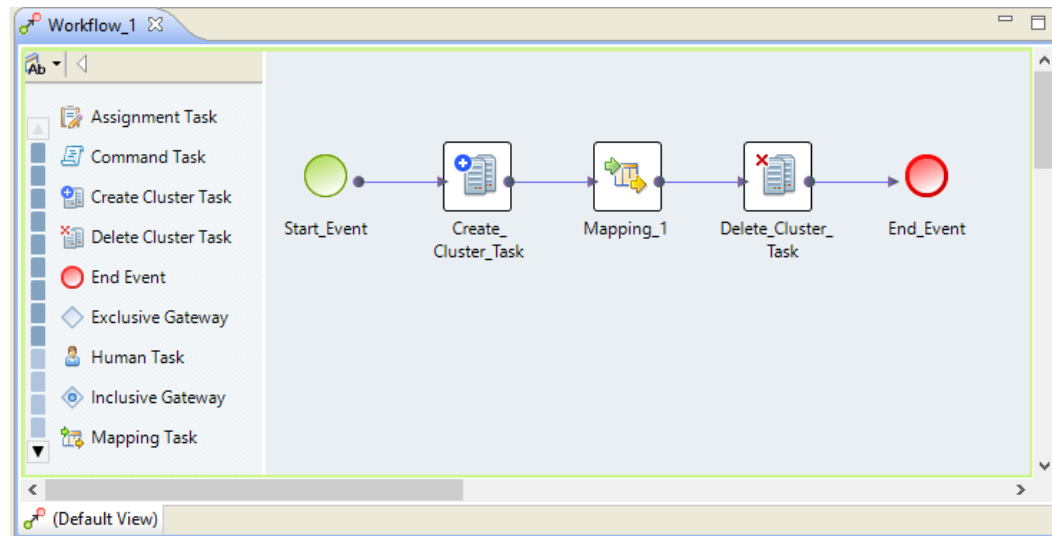
1 つのクラスタワークフローには 1 つのクラスタの作成タスクが含まれ、このタスクは、作成するクラスタに関する情報を使用して設定します。エフェメラルクラスタを作成する場合、クラスタの削除タスクを含められます。エフェメラルクラスタは、マッピングおよびその他のタスクを実行し、タスクが完了したら終了するクラウドプラットフォームクラスタです。クラウドプラットフォームリソースを保存するエフェメラルクラスタを作成します。

注: クラスタを Cloudera Altus 上で作成するには、クラスタワークフローを自動化するタスクを実行する、コマンドタスクを持つワークフローを作成します。Cloudera Altus 上でのクラスタの作成に関する詳細については、Informatica ネットワークで、Cloudera Altus クラスタ上で Informatica Big Data Management をエフェメラルクラスタと一緒に実装する方法に関する記事を参照してください。

クラスタワークフローのコンポーネント

エフェメラルクラスタを作成するクラスタワークフローには、クラスタの作成タスクと、少なくとも 1 つのマッピングタスク、クラスタの削除タスクが含まれます。

次の図に、クラスタワークフローの例を示します。



クラスタワークフローでは次のコンポーネントを使用します。

クラウドプロビジョニング設定

クラウドプロビジョニング設定は、クラスタ接続を介してクラスタの作成タスクに関連付けられます。

クラスタ接続

クラスタワークフローで使用するクラスタ接続は、クラウドプロビジョニング設定に関連付けられます。Hadoop または Databricks クラスタ接続を使用できます。ワークフローを実行すると、データ統合サービスにより一時的なクラスタ接続が作成されます。

クラスタの作成タスク

クラスタの作成タスクには、クラウドプラットフォームがマスタノードおよび作業ノードを持つクラスタを作成するために必要なすべての設定が含まれます。クラウドプロビジョニング設定への参照も含まれます。クラスタワークフローには、1 つのクラスタの作成タスクを含めます。

マッピングタスク

ビッグデータマッピングをマッピングタスクに追加します。クラスタワークフローには、複数のマッピングタスクを含めることができます。マッピングは、一部を既存のクラスタで実行し、一部をワークフローで作成するクラスタで実行することができます。マッピングとマッピングタスクは、タスクを実行する場所に基づいて設定します。

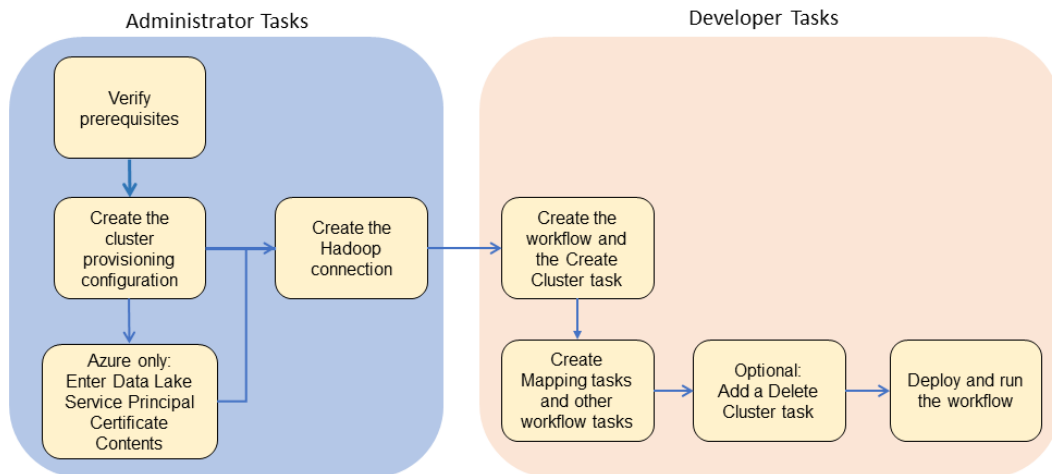
クラスタの削除タスク

クラスタの削除タスクでは、クラスタが終了され、そのクラスタと、ワークフローが作成したその他のすべてのリソースが削除されます。

クラスタワークフロープロセス

クラスタワークフローを作成するには、管理者と開発者のタスクが必要です。

次の図に、クラスタワークフローを作成、設定、実行するための処理を示します。



クラスタワークフロー開発プロセスには、次のユーザーによるタスクが必要です。

Administrator

1. **インストールを確認します。**ドメインは、オンプレミスにするか、クラウドプラットフォーム内のものにすることができます。
2. **クラスタプロビジョニング設定を作成します。**Administrator tool でクラウドプロビジョニング設定を作成します。クラウドプロビジョニング設定には、データ統合サービスがクラウドプラットフォーム上でリソースのアクセスおよび作成に必要なすべての情報が格納されます。
3. **クラスタ接続を作成します。**クラスタプロビジョニング設定に関連付ける専用のクラスタ接続を作成します。

管理者タスクの詳細については、『*Big Data Management Administrator Guide*』を参照してください。

ディベロッパー

ディベロッパーは、次のタスクを完了します。

1. ワークフローとクラスタの作成タスクを作成します。
2. マッピングタスクと、必要に応じてその他のタスクを作成します。
3. 処理が完了したときにクラスタとリソースを削除する場合は、クラスタの削除タスクを作成します。
4. このワークフローをデプロイして実行します。

クラスタの作成タスクのプロパティ

クラスタの作成タスクを設定するときは、すべてのクラスタタイプに共通のオプションを設定し、特定のクラスタの詳細オプションを設定します。

クラスタの作成タスクのための全般プロパティ

次の表に、クラスタの作成タスクの全般プロパティを示します。

プロパティ	説明
名前	タスク名。
説明	オプション説明。
接続名	ワークフローと一緒に使用するクラウドプロビジョニング設定の名前。
接続タイプ	次のオプションから選択します。 - Amazon EMR。Hadoop 環境で Amazon EMR クラスタを作成します。 - HDInsight。Hadoop 環境で Azure HDInsight クラスタを作成します。 - DatabricksDatabricks 環境で Databricks クラスタを作成します。

クラスタの作成タスクのための入力プロパティ

タスクの入力をワークフローパラメータおよび変数に割り当てることができます。

次の表に、クラスタの作成タスクの入力プロパティを示します。

プロパティ	説明
クラスタの作成タスクの入力	タスクの入力に使用するパラメータまたは変数。

クラスタの作成タスクのための出力プロパティ

変数をクラスタの作成タスクの出力に割り当てることができます。

次の表に、クラスタの作成タスクの出力プロパティを示します。

プロパティ	説明
開始時刻	タスクの実行が開始された日時。
終了時刻	タスクの実行が終了した日時。
成功	タスクが正常に実行されたかを示します。
クラスタ識別子	クラスタの作成タスクのクラスタ識別子プロパティは、マッピングタスクのクラスタ識別子プロパティよりも優先されます。 デフォルトは AutoDeployCluster です。

クラスタの作成タスクのための詳細プロパティ

クラスタの作成タスクのための詳細プロパティは、作成するクラスタタイプによって異なります。

Microsoft Azure の詳細プロパティの詳細については、「[Azure HDInsight の詳細プロパティ](#)」 (ページ 116) を参照してください。

Amazon Web Services の詳細プロパティの詳細については、[「Amazon EMR の詳細プロパティ」](#) (ページ 117) を参照してください。

Blaze エンジンでマッピングを実行するための詳細プロパティの詳細については、[「Blaze エンジンの詳細プロパティ」](#) (ページ 120) を参照してください。

Databricks の詳細プロパティの詳細については、[「Databricks の詳細プロパティ」](#) (ページ 121) を参照してください。

Azure HDInsight の詳細プロパティ

Azure HDInsight クラスタの詳細プロパティを設定します。

次の表に、Microsoft Azure HDInsight クラスタの詳細プロパティを示します。

プロパティ	説明
クラスタ名	作成するクラスタの名前。
Azure クラスタタイプ	作成されるクラスタのタイプ。 ドロップダウンリストでいずれかのオプションを選択します。 デフォルトは Hadoop です。
HDInsight バージョン	クラスタ上で実行する HDInsight バージョン。バージョンを指定する HDInsight バージョンタグ文字列を入力します。 デフォルトはサポートされている最新バージョンです。
Azure クラスタの場所	ドロップダウンリストを使用して、クラスタを作成する場所を選択します。
ヘッドノードの VM サイズ	作成するヘッドノードインスタンスサイズ。 デフォルトは Standard_D12_v2 です。
作業ノードインスタンスの数	クラスタで作成する作業ノードインスタンスの数。 デフォルトは 2 です。
作業ノードの VM サイズ	作成する作業ノードインスタンスサイズ。 デフォルトは Standard_D13_v2 です。
デフォルトストレージタイプ	クラスタに使用するプライマリストレージタイプ。 次のいずれかのオプションを選択します。 - Azure Data Lake Store - Azure BLOB ストレージアカウント デフォルトは BLOB ストレージです。
デフォルトのストレージコンテナまたはルートマウントパス	データのデフォルトのコンテナ。次のいずれかのパスを入力します。 - ADLS ストレージの場合、そのストレージへのパスを入力します。例えば、storage-name または storage-name/folder-name と入力できます。 - BLOB ストレージの場合、そのコンテナへのパスを入力します。形式: /path/

プロパティ	説明
ログの場所	オプション。ワークフローイベントログを格納するディレクトリへのパス。 デフォルトは/app-logs です。
外部 Hive メタストアのアタッチ	このオプションを選択すると、クラウドプロビジョニング設定で外部 Hive メタストアが設定されている場合、ワークフローで外部 Hive メタストアがクラスタに接続されます。
ブートストラップの JSON 文字列	クラスタの作成中に実行される JSON 文。この文は、クラスタの詳細を設定するために使用できます。例えば、クラスタ上で Hadoop プロパティを設定し、タグをクラスタリソースに追加する、またはスクリプトアクションを実行できます。 プロパティを入力するには、次のいずれかの方法を選択します。 - JSON 文を入力します。次の形式を使用します。 <pre>{ "core-site" : { "<sample_property_key1>": "<sample_property_val1>", "<sample_property_key2>": "<sample_property_val2>" }, "tags": { "<tag_key>": "<tag_val>" }, "scriptActions": [{ "name": "setenvironmentvariable", "uri": "scriptActionUri", "parameters": "headnode" }] }</pre> - JSON 文が格納されているファイルへのパスを入力します。形式: <pre>file://<path_to_bootstrap_file></pre>

Amazon EMR の詳細プロパティ

Amazon EMR クラスタの詳細プロパティを設定します。

全般オプション

次の表に、EMR クラスタに対して設定できる全般オプションを示します。

プロパティ	説明
クラスタ名	作成するクラスタの名前。
リリースバージョン	クラスタ上で実行する EMR バージョン。 バージョンを指定する AWS バージョンタグ文字列を入力します。例: emr-5.8.0 デフォルトはサポートされている最新バージョンです。
接続名	クラスタワークフローと一緒に使用するために設定した Hadoop 接続の名前。
S3 ログ URI	オプション。クラスタ作成のためのログの S3 の場所。形式: <pre>s3://<bucket name>/<folder name></pre> 場所を指定しない場合、クラスタログは保存されません。

マスタインスタンスグループのオプション

次の表に、EMR クラスタに対して設定できるマスタインスタンスグループのオプションを示します。

プロパティ	説明
マスタインスタンス タイプ	マスタノード EC2 インスタンスタイプ。 任意の使用できる EC2 インスタンスタイプを指定できます。 デフォルトは m4.4xlarge です。
マスタインスタンスの 最大スポット価格	マスタノードの最大スポット価格。このプロパティを設定すると、マスタインスタンスグループの購入オプションがオンデマンドではなくスポットに変更されます。

コアインスタンスグループのオプション

次の表に、EMR クラスタに対して設定できるコアインスタンスグループのオプションを示します。

プロパティ	説明
コアインスタンス タイプ	コアノード EC2 インスタンスタイプ。 任意の使用できる EC2 インスタンスタイプを指定できます。 デフォルトは m4.4xlarge です。
コアインスタンス 数	クラスタで作成するコア EC2 インスタンスの数。 デフォルトは 2 です。
コアインスタンス 最大スポット 価格	コアノードの最大スポット価格。このプロパティを設定すると、コアインスタンスグループの購入オプションがオンデマンドではなくスポットに変更されます。
コア自動スケー リングポリシー	オプション。コアインスタンスの自動スケーリングポリシー。ここにポリシー JSON 文を入力するか、JSON 文が格納されているファイルへのパスを入力します。 形式: file:\\<path_to_policy_config_file>

タスクインスタンスグループのオプション

次の表に、EMR クラスタに対して設定できるタスクインスタンスグループのオプションを示します。

プロパティ	説明
タスクインスタンス タイプ	タスクノード EC2 インスタンスタイプ。 任意の使用できる EC2 インスタンスタイプを指定できます。 デフォルトは m4.4xlarge です。
タスクインスタンス 数	クラスタで作成するタスク EC2 インスタンスの数。 デフォルトは 2 です。

プロパティ	説明
タスクインスタンスの最大スポット価格	タスクノードの最大スポット価格。このプロパティを設定すると、タスクインスタンスグループの購入オプションは、オンデマンドではなくスポットに変更されます。
タスク自動スケーリングポリシー	オプション。タスクインスタンスの自動スケーリングポリシー。ここにポリシー JSON 文を入力するか、JSON 文が格納されているファイルへのパスを入力します。 形式: file:\ <path_to_policy_config_file>< td=""></path_to_policy_config_file><>

追加オプション

次の表に、EMR クラスタに対して設定できる追加オプションを示します。

プロパティ	説明
アプリケーション	オプション。AWS によりインストールされるデフォルトアプリケーションに追加するアプリケーション。 AWS によって、EMR クラスタの作成時に特定のアプリケーションがインストールされます。さらに、追加のアプリケーションを指定できます。追加のアプリケーションをドロップダウンリストから選択します。 このフィールドは、AWS EMR クラスタ作成ウィザードのソフトウェア設定リストと同じです。
タグ	オプション。クラスタ EC2 インスタンスにプロパゲートするタグ。 EC2 インスタンスの特定を補助するタグ。 形式: TagName1=TagValue1,TagName2=TagValue2
ソフトウェア設定	オプション。クラスタにインストールされているアプリケーションに適用するカスタム設定。 このフィールドは、AWS クラスタ作成ウィザードのソフトウェア設定の編集フィールドと同じです。この方法を使用して、クラスタ上のソフトウェア設定を変更できます。 ここに設定 JSON 文を入力するか、JSON 文が格納されているファイルへのパスを入力します。 形式: file:\ <path_to_custom_config_file>< td=""></path_to_custom_config_file><>
手順	オプション。クラスタ作成後に実行するコマンド。例えば、Linux コマンド、または HDFS コマンドや Hive Hadoop コマンドを実行するために使用できます。 このフィールドは、AWS クラスタ作成ウィザードの手順の追加フィールドと同じです。 ここにコマンド文を入力するか、JSON 文が格納されているファイルへのパスを入力します。形式: file:\ <path_to_command_file>< td=""></path_to_command_file><>
ブートストラップアクション	オプション。EC2 インスタンスの実行後、アプリケーションのインストール前に実行するアクション。 ここに JSON 文を入力するか、JSON 文が格納されているファイルへのパスを入力します。形式: file:\ <path_to_policy_config_file>< td=""></path_to_policy_config_file><>
タスクリカバリストラテジ	次のオプションから選択します。 - タスクの再開 - タスクのスキップ デフォルトは、再起動タスクです。

Blaze エンジンの詳細プロパティ

クラウドプラットフォームクラスタでマッピングを実行するために Blaze エンジンを使用する場合、クラスタの作成タスクのソフトウェア設定プロパティで、クラスタ設定プロパティを設定する必要があります。

クラスタの作成タスクを設定し、クラスタ上の *-site.xml ファイルで設定プロパティを設定します。Hadoop クラスタはこれらの設定に基づいて実行されます。

テキストファイルを使用して、*-site.xml ファイル設定を指定し、このファイルへのパスをソフトウェア設定プロパティ内で指定します。

次のテキストに、ソフトウェア設定プロパティファイルの設定例を示します。

```
[
  {
    "Classification": "yarn-site",
    "Properties": {
      "yarn.scheduler.minimum-allocation-mb": "250",
      "yarn.scheduler.maximum-allocation-mb": "8192",
      "yarn.nodemanager.resource.memory-mb": "16000",
      "yarn.nodemanager.resource.cpu-vcores": "12"
    }
  },
  {
    "Classification": "core-site",
    "Properties": {
      "hadoop.proxyuser.<DIS/OSPUSER>.groups": "<group names>",
      "hadoop.proxyuser.<DIS/OSPUSER>.hosts": "*"
    }
  }
]
```

yarn-site

yarn.scheduler.minimum-allocation-mb

各コンテナに使用できる最小 RAM を示します。Blaze エンジンリソース割り当てに必要です。

yarn.scheduler.maximum-allocation-mb

各コンテナに使用できる最大 RAM を示します。Blaze エンジンリソース割り当てに必要です。

yarn.nodemanager.resource.memory-mb

各コンテナに使用できる最大 RAM を示します。Blaze エンジンに使用できるリソースメモリを増やすには、クラスタ上の最大メモリを設定します。

yarn.nodemanager.resource.cpu-vcores

各コンテナの仮想コアを示します。Blaze エンジンリソース割り当てに必要です。

core-site

hadoop.proxyuser.<proxy user>.groups

プロキシユーザーアカウントが偽装できるグループを定義します。セキュアなクラスタ上では、<proxy user>は、クラスタキータブファイルに一致するサービスプリンシパル名です。セキュアでないクラスタ上では、<proxy user>は、Informatica デーモンを実行するシステムユーザーです。

カンマ区切りの偽装ユーザーのグループ名を設定します。セキュリティが低くてよい場合、ワイルドカードの「*」を使用し、すべてのグループからの偽装を許可します。

hadoop.proxyuser.<user>.hosts

偽装接続を許可するホストマシンを定義します。セキュアなクラスタ上では、<proxy user>は、クラスタキータブファイルに一致するサービスプリンシパル名です。セキュアでないクラスタ上では、<proxy user>は、Informatica デーモンを実行するシステムユーザーです。

プロパティを「*」に設定し、すべてのホストからの偽装を許可します。これは、Blaze マッピングをクラウドプラットフォームクラスタ上で実行するために必要です。

Hive メタストアデータベースの詳細プロパティ

クラウドプラットフォームのリレーショナルデータベースをクラスタの Hive メタストアデータベースとして使用する場合、クラスタの作成タスクのソフトウェア設定プロパティで、クラスタ設定プロパティを設定する必要があります。

クラスタの作成タスクを設定し、クラスタ上の hive-site.xml 設定ファイルで設定プロパティを設定します。テキストファイルを使用して、hive-site 設定を指定し、このファイルへのパスをソフトウェア設定プロパティ内で指定します。

ソフトウェア設定プロパティファイルで以下のプロパティを作成します。

`javax.jdo.option.ConnectionURL`

データストアの JDBC 接続文字列。

`javax.jdo.option.ConnectionDriverName`

データストアの JDBC ドライバクラス名。クラウドプラットフォームと互換性がある JDBC ドライバを指定します。

`javax.jdo.option.ConnectionUserName`

データベースへの接続に使用するユーザー名。

`javax.jdo.option.ConnectionPassword`

上記データベースユーザアカウントのパスワード。

次のテキストに、ソフトウェア設定プロパティファイルの設定例を示します。

```
{
  "Classification": "hive-site",
  "Properties": {
    "javax.jdo.option.ConnectionURL": "jdbc:mysql://<RDS_HOST>:<PORT>\\/<USER_SCHEMA>?createDatabaseIfNotExist=true",
    "javax.jdo.option.ConnectionDriverName": "<JDBC driver name>",
    "javax.jdo.option.ConnectionUserName": "<USER>",
    "javax.jdo.option.ConnectionPassword": "<USER>"
  }
}
```

以下に例を示します。

```
{
  "Classification": "hive-site",
  "Properties": {
    "javax.jdo.option.ConnectionURL": "jdbc:mysql://<host name>:<port number>\\/<hive>",
    "createDatabaseIfNotExist": true,
    "javax.jdo.option.ConnectionDriverName": "org.mariadb.jdbc.Driver",
    "javax.jdo.option.ConnectionUserName": "hive",
    "javax.jdo.option.ConnectionPassword": "hive"
  }
}
```

Databricks の詳細プロパティ

Databricks のクラスタの作成タスクの詳細プロパティの全般オプションと詳細オプションを設定します。

全般オプション

次の表に、Databricks クラスタに対して設定できる全般オプションを示します。

プロパティ	説明
クラスタ名	作成するクラスタの名前。
Databricks Runtime Version	クラスタ上で実行する Databricks バージョン。 デフォルトは、最新サポートバージョンです。
ドライバの種類	ドライバノードで使用するノードのタイプ。 デフォルトは、ワーカーの種類 ID です。
ワーカーの種類	作業ノードで使用するノードのタイプ。
ワーカー数	クラスタで作成する作業ノードの数。クラスタの自動スケーリングを設定している場合は、このプロパティは無視されます。 デフォルトは 1 です。
自動スケーリング	負荷に基づいて、作業ノードの数を自動的にスケーリングします。
最小ワーカー数	クラスタの自動スケーリングを設定しているときに使用する、作業ノードの最小数。 デフォルトは 0 です。
最大ワーカー数	クラスタの自動スケーリングを設定しているときに使用する、作業ノードの最大数。 デフォルトは 1 です。

詳細オプション

環境変数や自動終了などの詳細オプションを設定します。

次の表に、Databricks クラスタに対して設定できる詳細オプションを示します。

プロパティ	説明
自動終了	クラスタの自動終了を有効にします。
自動終了時間	指定された分数、非アクティブだった後にクラスタを終了します。10～10,000 の範囲で値を入力してください。これを設定しない場合、または 0 に設定した場合、クラスタは自動終了しません。
クラスタログ設定	長期ストレージ用のログを記録する場所。設定した場合、Databricks Spark エンジン は 5 分間隔でログを記録します。 DBFS へのパスを指定します。
初期化スクリプト	初期化スクリプトログを保存する場所。複数の場所を入力できます。スクリプトは、設定した順序で順番に実行されます。次の形式を使用します。 <DBFS path>/script1, script2
クラスタタグ	トレースの目的でリソースに割り当てることができるラベル。"key1" = <value1>, "key2" = <value2> の形式でキーと値のペアを入力します。キーと値のペアを含むファイルへのパスを指定することもできます。

プロパティ	説明
Spark 設定	Databricks Spark エンジンのパフォーマンス設定。"<key1>" = <value1> , "<key2>" = <value2>の形式でキーと値のペアを入力します。キーと値のペアを含むファイルへのパスを指定することもできます。
環境変数	Databricks Spark エンジンに対して設定できる環境変数。"<key1>" = <value1> , "<key2>" = <value2>の形式でキーと値のペアを入力します。

マッピングタスクのプロパティ

ワークフローがマッピングタスクを実行する場所を指定するように、マッピングとマッピングタスクのプロパティを設定します。

クラスタワークフローにマッピングタスクを含めるときは、マッピングを実行するクラスタを指定する必要があります。

次の表に、[マッピング] タスクの [詳細] プロパティで設定できる [クラスタ識別子] プロパティ値を示します。

クラスタ識別子プロパティ	説明
自動デプロイ	そのワークフローが作成したクラスタ上でマッピングを実行します。データ統合サービスが、クラスタのプロビジョニング設定に関係する接続に基づいて一時的なクラスタ接続を生成します。マッピングの接続プロパティも [自動デプロイ] に設定されている必要があります。
空白	マッピングと関連付けられたクラスタ接続内に設定されたクラスタで、マッピングを実行します。マッピングの接続プロパティが、クラスタ設定に関連付けられたクラスタ接続に設定されている必要があります。クラスタのプロビジョニング設定に関連付けられている場合は、検証は失敗します。
タスク入力に割り当てる	クラスタの作成タスクとは別のソースからの入力を受け入れます。このオプションは、パラメータを使用するように入力プロパティを設定する場合に使用します。

注: マッピングタスクおよびクラスタの作成タスクに含まれているクラスタ識別子が異なる値の場合、クラスタの作成タスクの値が優先されます。

クラスタの削除タスクの追加

エフェメラルクラスタを作成するには、クラスタの削除タスクを追加します。

クラスタの削除タスクでは、クラスタが終了し、そのクラスタと、クラスタワークフローが作成したその他のリソースが削除されます。

クラスタの削除タスクを追加しない場合、ワークフローが作成したクラスタは、ワークフローの終了時に実行されたままになります。クラスタはいつでも削除できます。

1. クラスタの削除タスクをワークフローエディタにドラッグします。
2. [全般] プロパティで、必要に応じてクラスタの削除タスクの名前を変更します。
3. ワークフロー内の最後のタスクをクラスタの削除タスクと連結し、クラスタの削除タスクをワークフローの End_Event に連結します。

クラウドクラスタの削除のために、`infacmd ccps deleteCluster` を使用することもできます。

ワークフローのデプロイと実行

クラスタワークフローの完了後、ワークフローをデプロイおよび実行できます。

AWS および Azure クラスタワークフローは Web コンソールで監視できます。ログイン場所を設定する場合、クラスタの作成タスクのプロパティで設定した場所でログを表示します。

Databricks クラスタワークフローは Databricks ワークスペースで監視できます。

データ統合サービスジョブは Administrator ツールでも監視できます。

注: ワークフローでのタスクの実行開始後、クラスタをプロビジョニングするタスクには数分間かかる場合があります。

Azure HDInsight クラスタワークフロージョブの監視

マッピングログ URL に、Administrator ツールの **【監視】** タブからアクセスし、Azure HDInsight クラスタで実行されるワークフロージョブを監視できます。ログの場所は、各マッピングが使用するランタイムエンジンによって異なります。

Blaze または Spark エンジンで実行されるマッピングの監視 URL にアクセスするには、**【監視】** タブでワークフローとマッピングを展開します。[グリッドタスク] を選択し、下のペインで [URL の監視] プロパティの値を表示します。このパスは、ログを見つけるために使用します。

第 7 章

プロファイル

この章では、以下の項目について説明します。

- [プロファイルの概要, 125 ページ](#)
- [ネイティブ環境, 125 ページ](#)
- [Hadoop 環境, 126 ページ](#)
- [Informatica Developer での単一のデータオブジェクトプロファイルの作成, 127 ページ](#)
- [Informatica Developer でのエンタープライズ検出プロファイルの作成, 128 ページ](#)
- [Informatica Analyst でのカラムプロファイルの作成, 129 ページ](#)
- [Informatica Analyst でのエンタープライズ検出プロファイルの作成, 130 ページ](#)
- [Informatica Analyst でのスコアカードの作成, 131 ページ](#)
- [プロファイルの監視, 132 ページ](#)
- [プロファイリング機能のサポート, 133 ページ](#)
- [トラブルシューティング, 134 ページ](#)

プロファイルの概要

カラムプロファイル、エンタープライズ検出プロファイル、およびスコアカードを、ネイティブのランタイム環境または Hadoop ランタイム環境で作成および実行できます。

プロファイルまたはスコアカードを作成または編集するときに、ランタイム環境を選択できます。ランタイム環境の選択後、Developer tool または Analyst ツールによってプロファイル定義にランタイム環境が設定されます。プロファイルをすばやく処理するには、Hadoop ランタイム環境を選択します。

ネイティブ環境

Informatica Developer では、単一オブジェクトのプロファイル、複数オブジェクトのプロファイル、エンタープライズ検出プロファイルをネイティブ環境で実行できます。Informatica Analyst では、カラムプロファイル、エンタープライズ検出プロファイル、スコアカードをネイティブ環境で実行できます。

ネイティブのランタイム環境でプロファイルを実行すると、Analyst ツールまたは Developer tool がプロファイルジョブをプロファイリングサービスモジュールにサブミットします。次に、プロファイリングサービスモジュールが、プロファイルジョブを一連のマッピングに分割します。データ統合サービスは、データ統合サー

ビスが実行する同じマシン上でこれらのマッピングを実行し、プロファイル結果をプロファイリングウェアハウスに書き込みます。デフォルトでは、すべてのプロファイルがネイティブのランタイム環境で実行されます。

ネイティブソースを使用し、ネイティブ環境でプロファイルを作成して実行できます。ネイティブのデータソースは、フラットファイル、リレーショナルソース、メインフレームソースなど、Hadoop 以外のソースのことです。また、ネイティブ環境で Hive または HDFS データソースを使用してマッピング仕様または論理データソースに対してプロファイルを実行することもできます。

Hadoop 環境

プロファイルとスコアカードは、Hadoop 環境内で Blaze エンジンによって実行できます。Hadoop オプションを選択し、Hadoop ランタイム環境でプロファイルを実行できます。Hadoop オプションを選択し、Hadoop 接続を選択すると、データ統合サービスがプロファイルロジックを Hadoop クラスタの Blaze エンジンにプッシュして、プロファイルを実行します。

Hadoop 環境でプロファイルを実行すると、Analyst ツールまたは Developer tool がプロファイルジョブをプロファイリングサービスモジュールにサブミットします。次に、プロファイリングサービスモジュールが、プロファイルジョブを一連のマッピングに分割します。データ統合サービスは、これらのマッピングを Hadoop 接続で Hadoop 環境にプッシュします。Blaze エンジンはマッピングを処理し、データ統合サービスはプロファイル結果をプロファイリングウェアハウスに書き込みます。

Developer tool では、単一のオブジェクトプロファイルと複数のオブジェクトプロファイル、およびエンタープライズ検出プロファイルを Blaze エンジンで実行できます。Analyst ツールでは、カラムプロファイル、エンタープライズ検出プロファイル、スコアカードを Blaze エンジンで実行できます。

Sqoop データソースのカラムプロファイル

Sqoop を使用するデータオブジェクトでカラムプロファイルを実行できます。カラムプロファイルを実行する Hadoop ランタイム環境を選択できます。

論理データオブジェクトまたはカスタマイズデータオブジェクトのカラムプロファイルを実行すると、num-mappers 引数を設定して並行処理を実現しパフォーマンスを最適化することができます。また、Sqoop で作業単位を分割する際の基準のカラムを指定するように、split-by 引数も設定しなければなりません。

以下の構文を使用します。

```
--split-by <column_name>
```

プライマリキーの範囲の上限と下限の間で値の均等分布がない場合、データの均等分布がある別のカラムを指定して作業単位を分割するように、split-by 引数を設定できます。

split-by カラムを定義しない場合、Sqoop では次の条件に基づいて作業単位が分割されます。

- データオブジェクトに単一プライマリキーが含まれる場合、Sqoop はプライマリキーを split-by カラムとして使用します。
- データオブジェクトにコンポジットプライマリキーが含まれる場合、Sqoop のデフォルトは、split-by 引数を使用せずにコンポジットプライマリキーを処理する Sqoop の動作になります。詳細については、Sqoop のマニュアルを参照してください。
- データオブジェクトに同一のカラムを持つ 2 つのテーブルが含まれる場合、split-by カラムをテーブル修飾名で定義する必要があります。例えば、テーブル名が CUSTOMER であり、カラム名が FULL_NAME の場合、split-by カラムを次のように定義します。

```
--split-by CUSTOMER.FULL_NAME
```
- データオブジェクトにプライマリキーがない場合、m 引数の値と num-mappers 引数は、デフォルトで 1 になります。

Cloudera Connector Powered by Teradata または Hortonworks Connector for Teradata を使用し、Teradata テーブルにプライマリキーがない場合、split-by 引数は必須です。

Informatica Developer での単一のデータオブジェクトプロファイルの作成

データオブジェクトの 1 つ以上のカラムに対して単一のデータオブジェクトプロファイルを作成し、プロファイルオブジェクトをモデルリポジトリに格納することができます。

1. **【オブジェクトエクスプローラ】** ビューで、プロファイリングするデータオブジェクトを選択します。
 2. **【ファイル】 > 【新規】 > 【プロファイル】** をクリックしてプロファイルウィザードを開きます。
 3. **【プロファイル】** を選択して **【次へ】** をクリックします。
 4. プロファイルの名前を入力し、プロジェクトの場所を確認します。必要に応じて、新しい場所を参照します。
 5. 必要に応じて、プロファイルの説明を入力します。
 6. 選択したデータオブジェクトの名前が **【データオブジェクト】** セクション内に表示されることを確認します。
 7. **【次へ】** をクリックします。
 8. 実行するプロファイルの操作を P 設定します。以下の操作を設定できます。
 - カラムプロファイリング
 - プライマリキー検出
 - 機能依存性検出
 - データドメイン検出
- 注:** プロファイルの操作を有効にするには、その操作に対して **【"プロファイルの実行"アクションの一部として有効化】** を選択します。カラムプロファイリングはデフォルトで有効になっています。
9. プロファイルのオプションを確認します。

すべてのプロファイルタイプのカラム選択を編集できます。カラムプロファイルのフィルタおよびサンプリングのオプションを確認します。プライマリキー、機能依存性、およびデータドメイン検出の推測オプションを確認できます。また、データドメイン検出のデータドメイン選択も確認できます。
 10. ドリルダウンのオプションを確認し、必要に応じて編集します。デフォルトでは、**【行のドリルダウンを有効にする】** オプションが選択されています。カラムプロファイルでは、ドリルダウンのオプションを編集できます。このオプションでは、ドリルダウン操作の読み取り元をデータソースにするか、またはステージングされたデータにするか、および以前のプロファイルの実行による結果データを格納するかも決定します。
 11. **【実行の設定】** セクションで、ランタイム環境を選択します。ランタイム環境として、**【ネイティブ】** または **【Hadoop】** を選択します。**【Hadoop】** オプションの選択後に、Hadoop 接続を選択できます。
 12. **【完了】** をクリックします。

Informatica Developer でのエンタープライズ検出プロファイルの作成

プロファイルは、複数の接続の複数のデータソースに対して作成できます。Developer ツールは、ソースごとに個別のプロファイルタスクを作成します。

1. **【オブジェクトエクスプローラ】** ビューで、プロファイルを実行する複数のデータオブジェクトを選択します。
2. **【ファイル】 > 【新規】 > 【プロファイル】** をクリックしてプロファイルウィザードを開きます。
3. **【エンタープライズ検出プロファイル】** を選択して、**【次へ】** をクリックします。
4. プロファイルの名前を入力し、プロジェクトの場所を確認します。必要に応じて、新しい場所を参照します。
5. 選択したデータオブジェクトの名前が **【データオブジェクト】** セクション内に表示されることを確認します。さらにデータオブジェクトを選択する必要がある場合は、**【選択】** をクリックして選択します。
6. **【次へ】** をクリックします。

【プロファイル定義へのリソースの追加】 ペインが表示されます。このペインから複数の外部リレーショナル接続とデータソースを選択できます。

7. **【選択】** をクリックして、**【リソースの選択】** ダイアログボックスを開きます。

【リソース】 ペインに、Informatica ドメインの下に、内部接続、外部接続、およびデータオブジェクトがすべて一覧表示されます。

8. **【OK】** をクリックしてダイアログボックスを閉じます。
9. **【次へ】** をクリックします。

10. 実行するプロファイルタイプを設定します。以下のプロファイルタイプを設定できます。

- データドメイン検出
- カラムプロファイル
- プライマリキープロファイル
- 外部キープロファイル

注: エンタープライズ検出プロファイルの一部として実行するプロファイルタイプに対して **【「エンタープライズ検出プロファイルの実行」アクションの一部として有効化】** を選択します。カラムプロファイリングはデフォルトで有効になっています。

11. プロファイルのオプションを確認します。

カラムプロファイルではサンプリングオプションを編集できます。また、データドメイン、プライマリキー、および外部キープロファイル用の推測オプションを編集できます。

12. **【プロファイルの作成】** を選択します。

Developer ツールによって、個々のデータソースのプロファイルが作成されます。

13. プロファイル設定を完了したら、**【終了時にエンタープライズ検出プロファイルを実行】** を選択してプロファイルを実行します。すべてのプロファイリング操作を有効にすると、選択したすべてのデータソースに対して、カラムプロファイル、データドメインプロファイル、およびプライマリキープロファイルが実行されます。その後、すべてのデータソースに対して外部キープロファイルが実行されます。

14. **【完了】** をクリックします。

エンタープライズ検出プロファイルを実行したら、結果を表示する前にモデルリポジトリサービスを更新する必要があります。モデルリポジトリで外部接続に対してメタデータのインポートが行われるため、この手順が必要となります。Developer ツールによってモデルリポジトリの変更が反映されるように、モデルリポジトリサービスを更新する必要があります。

Informatica Analyst でのカラムプロファイルの作成

カスタムプロファイルまたはデフォルトプロファイルを作成できます。カスタムプロファイルを作成する際には、カラム、サンプル行、ドリルダウンオプションを設定できます。デフォルトプロファイルを作成すると、カラムプロファイルとデータドメイン検出がすべてのデータドメインの全データセットに対して実行されます。

1. **【検出】** ワークスペースで、**【プロファイル】** をクリックするか、ヘッダ領域から **【新規】** > **【プロファイル】** を選択します。

注: 【ライブラリ】 ワークスペースでデータオブジェクトを右クリックして、プロファイルを作成できます。このプロファイルでは、データオブジェクトのプロパティからプロファイル名、場所名、およびデータオブジェクトが抽出されます。デフォルトのプロファイルを作成するか、設定をカスタマイズして、カスタムプロファイルを作成できます。

新規プロファイルウィザードが表示されます。

2. **【単一ソース】** オプションがデフォルトで選択されています。**【次へ】** をクリックします。
3. **【全般プロパティの指定】** 画面で、プロファイルの名前と説明（省略可能）を入力します。**【場所】** フィールドで、プロファイルの作成先のプロジェクトまたはフォルダを選択します。**【次へ】** をクリックします。
4. **【ソースの選択】** 画面で、**【選択】** をクリックしてデータオブジェクトを選択するか、**【新規】** をクリックしてデータオブジェクトをインポートします。**【次へ】** をクリックします。

- **【データオブジェクトの選択】** ダイアログボックスで、データオブジェクトを選択します。**【OK】** をクリックします。

【プロパティ】 ペインに、選択したデータオブジェクトのプロパティが表示されます。**【データプレビュー】** ペインに、データオブジェクトのカラムが表示されます。

- **【新しいデータオブジェクト】** ダイアログボックスで、接続、スキーマ、テーブル、またはビューを選択し、プロファイルの作成、場所の選択、データオブジェクトのインポート先フォルダの作成ができます。**【OK】** をクリックします。

5. **【ソースの選択】** 画面で、プロファイルを実行するカラムを選択します。**【名前】** を選択すると、すべてのカラムを選択できます。**【次へ】** をクリックします。

デフォルトでは、すべてのカラムが選択されています。Analyst ツールによって、名前、データ型、精度、スケール、NULL 可能、および各カラムがプライマリキーに含まれているかどうかなど、カラムのプロパティがリストされます。

6. **【設定の指定】** 画面で、カラムプロファイル、データドメイン検出、またはカラムプロファイルとデータドメイン検出のうちどれを実行するかを選択します。デフォルトでは、カラムプロファイルオプションが選択されています。

- **【カラムプロファイルを実行】** を選択して、カラムプロファイルを実行します。

- **【データドメイン検出を実行】** を選択して、データドメイン検出を実行します。**【データドメイン】** ペインで、検出するデータドメインを選択して、適合条件を選択し、**【データドメイン検出のカラム選択の編集】** ダイアログボックスでデータドメイン検出対象カラムを選択します。

- **【カラムプロファイルを実行】** と **【データドメイン検出を実行】** を選択して、カラムプロファイルとデータドメイン検出を実行します。**【データドメイン】** ペインでデータドメインオプションを選択します。

注: デフォルトでは、選択したカラムはカラムプロファイルとデータドメイン検出の対象となります。**【編集】** をクリックして、データドメイン検出のカラムを選択または選択解除します。

- データドメイン検出の実行対象として、**【データ】**、**【カラム】**、**【データとカラム】** を選択します。

- サンプルングオプションを選択します。**【次に対してプロファイルを実行】** ペインで、サンプルングオプションとして **【すべての行（完全分析）】**、**【サンプルング最初の】**、**【ランダムサンプルング】**、または **【ランダムサンプルング（自動）】** を選択できます。このオプションはカラムプロファイルとデータドメイン検出に適用されます。

- ドリルダウンオプションを選択します。[ドリルダウン] ペインでは、[ライブ] または [ステージングされた] ドリルダウンオプションを選択したり、[オフ] を選択してドリルダウンを無効にしたりできます。必要に応じて、[カラムの選択] をクリックして、ドリルダウンするカラムを選択します。承認済みのデータ型またはデータドメインを持つカラムの場合は、データ型およびデータドメインの推測を省略できます。
 - ランタイム環境として、[ネイティブ] または [Hadoop] オプションを選択します。[Hadoop] オプションを選択した場合は、[選択] をクリックして、[Hadoop 接続の選択] ダイアログボックスで Hadoop 接続を選択します。
7. [次へ] をクリックします。
[ルールとフィルタの指定] 画面が開きます。
 8. [ルールとフィルタの指定] 画面では、次のタスクを実行できます。
 - ルールを作成、編集、または削除します。既存のルールをプロファイルに適用できます。
 - フィルタを作成、編集、または削除します。

注: このプロファイルでスコアカードを作成した場合、このプロファイル用に作成したフィルタを再利用できます。
 9. [保存して完了] をクリックしてプロファイルを作成するか、[保存して実行] をクリックしてプロファイルを作成および実行します。

Informatica Analyst でのエンタープライズ検出プロファイルの作成

Informatica Analyst のエンタープライズ検出の一部分として、カラムプロファイルとデータドメイン検出を実行することができます。

1. [検出] ワークスペースで、[新規] > [プロファイル] を選択します。
[新規プロファイル] ウィザードが表示されます。
2. [エンタープライズ検出] を選択します。[次へ] をクリックします。
[全般プロパティの指定] タブが表示されます。
3. [全般プロパティの指定] タブで、エンタープライズ検出プロファイルの名前と説明（省略可能）を入力します。[場所] フィールドで、プロファイルの作成先のプロジェクトまたはフォルダを選択します。[次へ] をクリックします。
[データオブジェクトの選択] タブが表示されます。
4. [データオブジェクトの選択] タブで [選択] をクリックします。
[データオブジェクトの選択] ダイアログボックスが表示されます。
5. [データオブジェクトの選択] ダイアログボックスで、1 つ以上のデータオブジェクトを選択してプロファイルに追加します。[保存] をクリックします。
[データオブジェクト] ペインにデータオブジェクトが表示されます。
6. [次へ] をクリックします。
[リソースの選択] タブが表示されます。
7. [リソースの選択] タブで [選択] をクリックし、[リソースの選択] タブを開きます。
複数のリレーショナルデータソースからデータをインポートできます。

8. **【リソースの選択】** タブで、プロファイルに含める接続、スキーマ、テーブル、およびビューを選択します。**【保存】** をクリックします。

ダイアログボックスの左ペインには、Informatica ドメインの内部接続、外部接続、スキーマ、テーブル、ビューのすべてが一覧表示されます。

【リソース】 ペインにリソースが表示されます。

9. **【次へ】** をクリックします。

【設定の指定】 タブが表示されます。

10. **【設定の指定】** タブで、カラムプロファイルのオプションとデータドメイン検出のオプションを指定できます。**【保存して完了】** をクリックしてエンタープライズ検出プロファイルを保存するか、**【保存して実行】** をクリックしてプロファイルを実行します。

【設定の指定】 タブで次のタスクを実行できます。

- データドメイン検出の有効化。**【選択】** をクリックして、**【データドメインの選択】** ダイアログボックスから検出するデータドメインを選択します。**【データドメイン検出のデータドメイン】** ペインに、選択したデータドメインが表示されます。
- データ、カラム名、またはデータとカラム名の両方を対象に、データドメイン検出を実行します。
- データソースのすべての行を選択するか、ドメイン検出を実行する行の最大数を選択します。
- データドメイン検出の最小一致率を選択するか、適合する行の最小数を指定します。
- カラムプロファイル設定を有効にし、カラムプロファイルのデータソースのすべての行または最初の数行を選択します。カラムプロファイルでデータ型が承認されているカラムについてのデータ型の推測を除外できます。
- ランタイム環境として、**【ネイティブ】** または **【Hadoop】** を選択します。

【サマリ】 タブと **【プロファイル】** タブで、エンタープライズ検出結果を表示できます。

Informatica Analyst でのスコアカードの作成

スコアカードを作成し、プロファイルからスコアカードにカラムを追加します。スコアカードにカラムを追加する前に、プロファイルを実行する必要があります。

1. **【ライブラリ】** ワークスペースで、プロファイルを含むプロジェクトまたはフォルダを選択します。

2. プロファイルをクリックして開きます。

プロファイル結果が **【検出】** ワークスペースのサマリビューに表示されます。

3. **【アクション】** > **【スコアカードに追加】** をクリックします。

【スコアカードに追加】 ウィザードが表示されます。

4. **【スコアカードに追加】** 画面で、新しいスコアカードを作成するように選択したり、既存のスコアカードを編集してカラムを定義済みのスコアカードに追加したりできます。デフォルトでは、**【新しいスコアカード】** オプションが選択されています。**【次へ】** をクリックします。

5. **【手順 2/8】** 画面で、スコアカードの名前を入力します。必要に応じて、スコアカードの説明を入力できます。スコアカードを保存するプロジェクトおよびフォルダを選択します。**【次へ】** をクリックします。

デフォルトでは、プロファイルで定義されているカラムとルールが選択されます。プロファイルに含まれていないカラムは追加できません。

6. **【手順 3/8】** 画面で、スコアカードにメトリックとして追加するカラムとルールを選択します。必要に応じて、左側のカラムヘッダのチェックボックスをクリックしてすべてのカラムを選択します。必要に応じて、**【カラム名】** を選択してカラム名をソートします。**【次へ】** をクリックします。

7. **【手順 4/8】** 画面で、メトリックにフィルタを追加できます。

プロファイルに対して作成したフィルタをメトリックに適用したり、新しいフィルタを作成したりできます。**【メトリックフィルタ】** ペインでメトリックを選択し、**【フィルタの管理】** アイコンをクリックして、**【フィルタの編集: カラム名】** ダイアログボックスを開きます。**【フィルタの編集: カラム名】** ダイアログボックスで、以下のいずれかのタスクを実行するように選択できます。

- プロファイル用に作成したフィルタを選択します。**【次へ】** をクリックします。
- 既存のフィルタを選択します。**【編集】** アイコンをクリックし、**【フィルタの編集】** ダイアログボックスでフィルタを編集します。**【次へ】** をクリックします。
- プラス (+) アイコンをクリックし、**【新規フィルタ】** ダイアログボックスでフィルタを作成します。**【次へ】** をクリックします。

必要に応じて、選択したフィルタをスコアカードのすべてのメトリックに適用することもできます。

フィルタが **【メトリックフィルタ】** ペインに表示されます。

8. **【手順 4/8】** 画面で **【次へ】** をクリックします。

9. **【手順 5/8】** 画面で、**【メトリック】** ペインの各メトリックを選択し、次のタスクを実行します。

- 有効な値を設定します。**【スコアで使用: 値】** ペインで、**【利用可能な値】** ペインの 1 つ以上の値を選択し、右矢印ボタンをクリックしてそれらの値を **【有効な値】** ペインに移動します。メトリックの有効な値の総数が **【利用可能な値】** ペインの上部に表示されます。
- メトリックのしきい値を設定します。**【メトリックのしきい値】** ペインで、**【正常】**、**【許容可能】**、および **【許容不可】** スコアに対してしきい値を設定します。
- 無効なデータのコストを設定します。メトリックのコストに定数値を割り当てるには、**【固定費】** を選択します。数値カラムを変動費としてメトリックに添付するには、**【変動費】** を選択し、**【カラムの選択】** をクリックして数値カラムを選択します。必要に応じて、**【コスト単位の変更】** をクリックしてコストの単位を変更します。メトリックに対して無効なデータのコストを設定しない場合は、**【なし】** を選択します。

10. **【次へ】** をクリックします。

11. **【手順 6/8】** 画面で、メトリックを追加できるメトリックグループを選択したり、新しいメトリックグループを作成したりできます。新しいメトリックグループを作成するには、グループアイコンをクリックします。**【次へ】** をクリックします。

12. **【手順 7/8】** 画面で、グループ内のメトリックのウェイトと、グループのしきい値を指定します。

13. **【手順 8/8】** 画面で、スコアカードを実行するランタイム環境として **【ネイティブ】** または **【Hadoop】** を選択します。**【Hadoop】** を選択した場合は、**【参照】** をクリックして Hadoop 接続を選択し、Blaze エンジンでプロファイルを実行します。

14. **【保存】** をクリックしてスコアカードを保存するか、**【保存して実行】** をクリックしてスコアカードを保存してから実行します。

スコアカードが **【スコアカード】** ワークスペースに表示されます。

プロファイルの監視

Administrator ツールでプロファイルを監視できます。

1. Administrator ツールで、**【モニタ】** タブをクリックします。
2. ナビゲータワークスペースで **【ジョブ】** を選択します。
3. プロファイリングジョブを選択します。

4. **【サマリ統計】** タブで、プロファイルの全般プロパティ、サマリ統計、およびプロファイルの詳細統計を表示できます。
5. **【実行統計】** タブをクリックして、プロファイルの実行統計を表示します。

プロファイリング機能のサポート

JDBC 接続またはネイティブ接続を介してプロファイリングウェアハウスに接続できます。プロファイリングウェアハウスの接続に基づいて、プロファイリングで特定の機能を実行できます。

以下の表に、Big Data Quality と Big Data Management で選択したプロファイリングウェアハウス接続のタイプに基づいて実行できる機能を示します。

機能	JDBC 接続	ネイティブ接続
単一のデータオブジェクトのカラムプロファイル	サポート	サポート
単一のデータオブジェクトのデータドメイン検出	サポート	サポート
プライマリキー検出を使用した単一のデータオブジェクトプロファイル	サポートされていません	サポートされていません
機能依存性検出を使用した単一のデータオブジェクトプロファイル	サポートされていません	サポートされていません
[サンプリング最初の<number>行] サンプリングを使用したプロファイル	サポート	サポート
[ランダムサンプリング<number>行] サンプリングを使用したプロファイル	サポート	サポート
スコアカードのメトリックと値のコスト	サポートされていません	サポート
プライマリキーおよび外部キー検出を使用したエンタープライズ検出プロファイル	サポートされていません	サポートされていません
結合分析と重複検出を使用したエンタープライズ検出プロファイル	サポートされていません	サポートされていません
推測された値と値頻度のドリルダウン	サポート	サポート
プロファイル結果のエクスポート	サポート	サポート
行数マッピング*	サポートされていません	サポート
*統計ヘルパーで行数の報告に失敗すると、プロファイリングは行数マッピングを実行します。		

トラブルシューティング

Hadoop 環境でプロファイルを実行した場合に、プロファイル結果にドリルダウンできますか。

はい。ステージングされたデータにドリルダウンするためのオプションを設定したプロファイルを除けば可能です。

Hadoop 環境でプロファイルを実行すると、次のエラーメッセージが表示されます: [[LDTM_1055] 統合サービスでマッピング[Profile_CUSTOMER_INFO12_14258652520457390]の Hive ワークフローを生成できませんでした]。これはどのように解決できますか。

このエラーは、Hadoop 環境でサポートされないデータソース、ルールトランスフォーメーション、またはランタイム環境が原因で発生する場合があります。Hadoop 環境で有効でないオブジェクトに関する詳細については、Hadoop 環境でのマッピングについての章を参照してください。

データソース、ルール、またはランタイム環境を変更して、プロファイルを再度実行できます。エラーの詳細については、プロファイルログファイルを参照してください。

プロファイルの実行後に、すべてのカラムのプロファイル結果に「該当なし」と表示されます。これはどのように解決できますか。

プロファイリング結果が、プロファイリングウェアハウスに含まれることを確認します。プロファイリング結果が表示されない場合は、Hadoop 接続のクラスタ環境変数プロパティでデータベースパスが正確に指定されていることを確認します。また、Administrator ツールの「監視」タブで、Hadoop ジョブトラッカのデータベースパスも確認できます。

Hive ソースでプロファイルを実行した後、結果が表示されません。Hadoop ジョブトラッカを確認すると、プロファイルジョブを開く際に「XML 解析エラー: 要素が見つかりません」のエラーが表示されます。これは何を意味しますか。

Hive データソースにレコードがなく、空になっています。プロファイルが正常に実行されるようにするには、データソースに少なくともデータが 1 行必要です。

Hive ソースでプロファイルを実行した後、カラムパターンの一部を参照できません。これはなぜですか？

Hive ソースをインポートする際に、Developer tool は文字列カラムの精度を 4000 に設定します。Developer tool では、精度が 255 より大きい文字列カラムのパターンを導出することはできません。この問題を解決するには、データソース内のこれらの文字列カラムの精度を 255 に設定して、プロファイルを再び実行します。

大規模な Hadoop ソースでプロファイルを実行すると、プロファイルジョブが失敗し、実行失敗のエラーが表示されます。考えられる原因は何ですか。

原因の 1 つは、接続の問題である可能性があります。次の手順を実行して、接続の問題を特定して解決します。

1. Hadoop ジョブトラッカを開きます。
2. プロファイルジョブを識別して開き、MapReduce ジョブを表示します。
3. 失敗したジョブのハイパーリンクをクリックして、エラーメッセージを表示します。エラーメッセージに、"java.net.ConnectException: 接続が拒否されました"のようなテキストが含まれる場合、問題の原因は Hadoop クラスタの問題です。問題を解決するには、ネットワーク管理者に問い合わせてください。

第 8 章

監視

この章では、以下の項目について説明します。

- [監視の概要, 135 ページ](#)
- [Hadoop 環境のログ, 135 ページ](#)
- [Blaze エンジンの監視, 141 ページ](#)
- [Spark エンジン監視, 149 ページ](#)

監視の概要

Administrator ツールの [モニタ] タブでは、Hadoop 環境で実行されたマッピングの統計とログイベントを表示できます。[監視] タブには、Blaze、および Spark の各エンジンで実行されるマッピングについて、現在の情報と履歴情報が表示されます。

[サマリ統計] ビューを使用すると、データ統合サービスでのオブジェクトの状態および分布のサマリをグラフで表示できます。データ統合サービスでオブジェクトを実行するために使用されたメモリおよび CPU のグラフを表示することもできます。

Hadoop 環境でマッピングを実行する場合、データ統合サービスはログイベントを生成します。Hadoop 接続障害、Hive クエリ障害、その他の Hadoop ジョブ障害などの異なる種類のエラーに関するログイベントを表示できます。

REST Operations Hub でも、データ統合サービスまたは Blaze または Spark エンジンで実行されるデブロイ済みのマッピングの監視統計を表示することができます。

注: [監視] タブで Databricks エンジンを監視することはできません。

Hadoop 環境のログ

データ統合サービスは、Hadoop 環境でのマッピングの実行時にログイベントを生成します。

Blaze および Spark エンジンのログを表示できます。Hadoop 接続、ジョブ障害などのさまざまな種類のエラーに関するログイベントを表示できます

Spark エンジンでマッピングを実行すると、論理データ変換ジェネレータが Informatica マッピングから生成するログ内で Scala コードを表示できます。

データ統合サービスに対して指定された拒否ファイルディレクトリで、拒否ファイルを表示できます。

YARN Web ユーザーインターフェース

クラスタで実行されたアプリケーションを、YARN Web ユーザーインターフェースで表示できます。Blaze、または Spark ジョブの監視 URL をクリックして、YARN Web ユーザーインターフェースにアクセスします。

Blaze、および Spark エン진은、Hadoop 接続に設定した Hadoop クラスタ上で実行されます。YARN Web ユーザーインターフェースは、エンジンが実行する各ジョブを YARN アプリケーションとして表示します。

次の図は、YARN Web ユーザーインターフェースのアプリケーション監視ページを示します。

The screenshot shows the YARN Web User Interface. On the left is a sidebar with a 'Cluster' dropdown menu and a list of application states: NEW, NEW SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, and KILLED. The main area is titled 'All Applications'. It contains 'Cluster Metrics' and 'User Metrics for dr.who'. Below these is a table of applications. The table has columns for ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Running Containers, and Allocated CPU V-Cores. The table lists several applications, all of which are in a 'FINISHED' state with a 'SUCCEEDED' final status.

ID	User	Name	Application Type	Queue	Start Time	Finish Time	State	Final Status	Running Containers	Allocated CPU V-Cores
application_1463379223882_0568	idmuid	InfSprk0	SPARK	root Idmuid	Mon May 16 13:11:59 -0700 2016	Mon May 16 13:13:03 -0700 2016	FINISHED	SUCCEEDED	N/A	N/A
application_1463379223882_0567	idmuid	InfSprk0	SPARK	root Idmuid	Mon May 16 13:11:58 -0700 2016	Mon May 16 13:13:01 -0700 2016	FINISHED	SUCCEEDED	N/A	N/A
application_1463379223882_0566	idmuid	InfSprk0	SPARK	root Idmuid	Mon May 16 13:11:56 -0700 2016	Mon May 16 13:13:00 -0700 2016	FINISHED	SUCCEEDED	N/A	N/A
application_1463379223882_0565	idmuid	InfSprk0	SPARK	root Idmuid	Mon May 16 13:11:55 -0700 2016	Mon May 16 13:12:59 -0700 2016	FINISHED	SUCCEEDED	N/A	N/A
application_1463379223882_0564	idmuid	InfSprk0	SPARK	root Idmuid	Mon May 16 13:11:54 -0700 2016	Mon May 16 13:12:59 -0700 2016	FINISHED	SUCCEEDED	N/A	N/A

【アプリケーションタイプ】は、どのエンジンが YARN アプリケーションをサブミットしたかを示します。

アプリケーション ID は、アプリケーションの一意の識別子です。アプリケーション ID は、アプリケーションサマリへのリンクになっています。URL は、Administrator ツールの監視 URL と同じです。

アプリケーションサマリで【ログ】リンクをクリックして、Hadoop クラスタ上のアプリケーションログを表示します。

アプリケーションログの情報量は、Developer tool でマッピングを設定するトレースレベルによって異なります。次の表は、各トレースレベルのアプリケーションログに表示される情報量を示しています。

トレースレベル	メッセージ
なし	ログに FATAL メッセージが表示されます。FATAL メッセージには、サービスがシャットダウンする、または利用不可能になる修復不能なシステム障害が含まれます。
簡易	ログには、FATAL と ERROR のコードメッセージが表示されます。ERROR メッセージには、接続障害、メタデータの保存や取得の失敗、サービスエラーが含まれます。
標準	ログには、FATAL、ERROR、および WARNING のメッセージが表示されます。WARNING エラーには、修復可能なシステム障害や警告が含まれます。
詳細 - 初期化	ログには、FATAL、ERROR、WARNING、および INFO の各メッセージが表示されます。INFO メッセージには、システムおよびサービスの変更メッセージが含まれます。
詳細 - データ	ログには、FATAL、ERROR、WARNING、INFO、および DEBUG の各メッセージが表示されます。DEBUG メッセージは、ユーザー要求のログです。

監視 URL へのアクセス

監視 URL を使用すると、Blaze ジョブ監視 Web アプリケーションまたは YARN Web アプリケーションユーザーインターフェイスが開きます。監視 URL には、Administrator ツールの **【実行統計】** ビューからアクセスします。

- Administrator ツールの **【モニタ】** タブで、**【実行統計】** ビューをクリックします。
- 【アドホックジョブ】** を選択するか、デプロイされたマッピングジョブまたはワークフローをナビゲータのアプリケーションから選択します。

ジョブのリストがコンテンツパネルに表示されます。

- マッピングジョブを選択し、マッピングを展開してマッピングのグリッドタスクを選択します。

監視 URL が **【プロパティ】** ビューに表示されます。

The screenshot shows the 'Ad Hoc Jobs' interface. At the top, there's a search bar and a filter dropdown set to 'Administrator'. Below is a table of jobs:

Name	Type	State	Job ID	Started By	Start Time	Elapsed Time	End Time
PassThrough	Mapping	Completed	T2GJgmceE...	Administrator	09/29/2015 19:52:38	00:01:32	09/29/2015 19:54:10
POSTSES...	Command ...	Completed	T2GJgmceE...	Administrator	09/29/2015 19:53:41	00:00:17	09/29/2015 19:53:58
MAINSESS...	Grid Task	Completed	T2GJgmceE...	Administrator	09/29/2015 19:52:57	00:00:43	09/29/2015 19:53:41
PRESESSI...	Command ...	Completed	T2GJgmceE...	Administrator	09/29/2015 19:52:38	00:00:02	09/29/2015 19:52:41

Below the table, it says 'Showing 33 results.' and there's a checkbox for 'Receive New Job Notifications'.

The selected job 'MAINSESSION_task2 - T2GJgmceEeWPuPKvpC0s5Q_MAINSESSION_task2' is expanded, showing a status message: 'This grid task is completed.'

Below that, the 'General Properties' section is expanded, showing the following details:

- Name: MAINSESSION_task2
- Type: Grid Task
- Started By: Administrator
- User Security Domain: Native
- Start Time: 09/29/2015 19:52:57
- Elapsed Time: 00:00:43
- End Time: 09/29/2015 19:53:41
- % Task Completed: 100
- Monitoring URL: <http://psrhaqadn21.informatica.com:9080/Blaze?tasktype=gridtask&id=qtid-24-1-79555597-4&isParent=false>
- Incoming Task Dependencies: PRESESSION_task0PRESESSION_task1
- Outgoing Task Dependencies: POSTSESSION_task3

Administrator ツールでの Hadoop 環境ログの表示

Blaze マッピングのログイベントを、Administrator ツールの **【監視】** タブから表示できます

- Administrator ツールで、**【モニタ】** タブをクリックします。
- 【実行統計】** ビューを選択します。
- ナビゲータでアドホックジョブ、デプロイ済みのマッピングジョブ、またはワークフローを選択して開きます。
 - アドホックジョブを選択するには、データ統合サービスを展開して **【アドホックジョブ】** をクリックします。

- デプロイ済みのマッピングジョブを選択するには、アプリケーションを展開して【デプロイ済みのマッピングジョブ】をクリックします。
- ワークフローを選択するには、アプリケーションを展開して【ワークフロー】をクリックします。

ジョブのリストがコンテンツパネルに表示されます。

4. 【アクション】 > 【選択したオブジェクトのログの表示】をクリックして、マッピングのランタイムログを表示します。

データ統合サービスが実行した Hive クエリと Blaze エンジンクエリの結果が、ログファイルに表示されます。これには、Hive セッションログと Hive セッションの履歴ファイルの場所が含まれます。

マッピングの監視

Hadoop 環境で実行するマッピングを監視できます。

1. Administrator ツールで、【モニタ】タブをクリックします。
2. 【実行統計】ビューを選択します。
3. ナビゲータでアドホックジョブ、デプロイ済みのマッピングジョブ、またはワークフローを選択して開きます。
- アドホックジョブを選択するには、データ統合サービスを展開して【アドホックジョブ】をクリックします。

The screenshot shows the Informatica Administrator interface. The 'Monitor' tab is selected, and the 'Execution Statistics' sub-tab is active. The 'Ad Hoc Jobs' section displays a table of jobs. The job 'm_FF_FF' is highlighted, and its properties are shown in the bottom panel.

Name	Type	State	Job ID	Started By	Start Time	Elapsed Time	End Time
m_FF_FF	Mapping	Completed	xL8TmiEei6Q...	Administrator	04/06/2018 16:23:44	00:01:58	04/06/2018 16:25:43
InfoSpark0	Spark Applic...	Completed	xL8TmiEei6Q...	Administrator	04/06/2018 16:23:51	00:01:50	04/06/2018 16:25:42
m_FF_FF	Mapping	Completed	vWig6cmiEei6...	Administrator	04/06/2018 16:23:29	00:01:43	04/06/2018 16:25:12
m_FF_FF	Mapping	Completed	lBq-RDmiEei6...	Administrator	04/06/2018 16:23:17	00:01:25	04/06/2018 16:24:42
m_FF_FF	Mapping	Completed	rfduJzmiEei6...	Administrator	04/06/2018 16:23:04	00:01:38	04/06/2018 16:24:42
m_FF_FF	Mapping	Completed	kIDdLTmiEei6...	Administrator	04/06/2018 16:00:48	00:02:41	04/06/2018 16:03:30
m_FF_FF	Mapping	Completed	p1MjUmEei6...	Administrator	04/06/2018 15:54:14	00:03:08	04/06/2018 15:57:23
m_FF_FF	Mapping	Completed	QBLZmiEei6...	Administrator	04/06/2018 15:51:25	00:01:46	04/06/2018 15:53:12
m_FF_FF	Mapping	Failed	OAKVpTmiEei...	Administrator	04/06/2018 15:51:07	00:00:08	04/06/2018 15:51:16
m_FF_FF	Mapping	Completed	D5nHozmiEei...	Administrator	04/06/2018 15:50:00	00:00:43	04/06/2018 15:50:43

Showing 26 results

The Mapping job has completed.

General Properties

Name	m_FF_FF
Type	Mapping
Started By	Administrator
User Security Domain	Native
Start Time	04/06/2018 16:23:44
Elapsed Time	00:01:58
End Time	04/06/2018 16:25:43
Operating System Profile	OSP_ospUser4

- デプロイ済みのマッピングジョブを選択するには、アプリケーションを展開して【デプロイ済みのマッピングジョブ】をクリックします。

The screenshot shows the Informatica Administrator web interface. The top navigation bar includes 'Manage', 'Monitor', 'Logs', 'Reports', 'Security', and 'Cloud'. The 'Monitor' tab is active, and the 'Execution Statistics' sub-tab is selected. On the left, the 'Navigator' pane shows a tree structure under 'Domain331', with 'Deployed Mapping Jobs' highlighted. The main content area, titled 'Deployed Mapping Jobs', contains a table with columns: Name, Type, State, Job ID, Started By, Start Time, Elapsed Time, and End Time. The table lists several jobs, with 'm_Ustx_Readback' (Job ID: Egk2Jjm0Eeil...) highlighted. Below the table, a message states 'The Deployed Mapping job has completed.' and the 'Properties' tab for the selected job is displayed, showing details like Name, Type, Started By, User Security Domain, Start Time, Elapsed Time, and End Time.

Name	Type	State	Job ID	Started By	Start Time	Elapsed Time	End Time
m_Ustx_Readback	Deployed Mapping	Completed	Egk2Jjm0Eeil...	Administrator	04/06/2018 21:33:39	00:08:17	04/06/2018 21:41:57
exec0	Script	Completed	Egk2Jjm0Eeil...	Administrator	04/06/2018 21:35:53	00:02:26	04/06/2018 21:38:19
exec0_qu...	Hive Query	Completed	svcoqa_2018...	Administrator	04/06/2018 21:36:34	00:01:06	04/06/2018 21:37:40
exec0_qu...	Hive Query	Completed		Administrator	04/06/2018 21:36:30	00:00:04	04/06/2018 21:36:34
TST_Monitoring...	Deployed Mapping	Completed	RxgjTmcEeil...	Administrator	04/06/2018 21:28:00	00:05:28	04/06/2018 21:33:28
m_Ustx_Readback	Deployed Mapping	Completed	HOCMBDmy...	Administrator	04/06/2018 21:19:36	00:08:02	04/06/2018 21:27:38
TST_Monitoring...	Deployed Mapping	Completed	TKRmXzmiE...	Administrator	04/06/2018 21:13:52	00:05:36	04/06/2018 21:19:28
m_Ustx_Readback	Deployed Mapping	Completed	FFQz3zmiE...	Administrator	04/06/2018 21:05:05	00:08:29	04/06/2018 21:13:34
TST_Monitoring...	Deployed Mapping	Completed	OTqYzmiEeil...	Administrator	04/06/2018 20:59:02	00:05:54	04/06/2018 21:04:56
hive_kbaes_700	Deployed Mapping	Completed	KX7VYDmiE...	Administrator	04/06/2018 20:48:20	00:00:12	04/06/2018 20:48:32

Showing 54 results

☒ Receive New Job Notifications

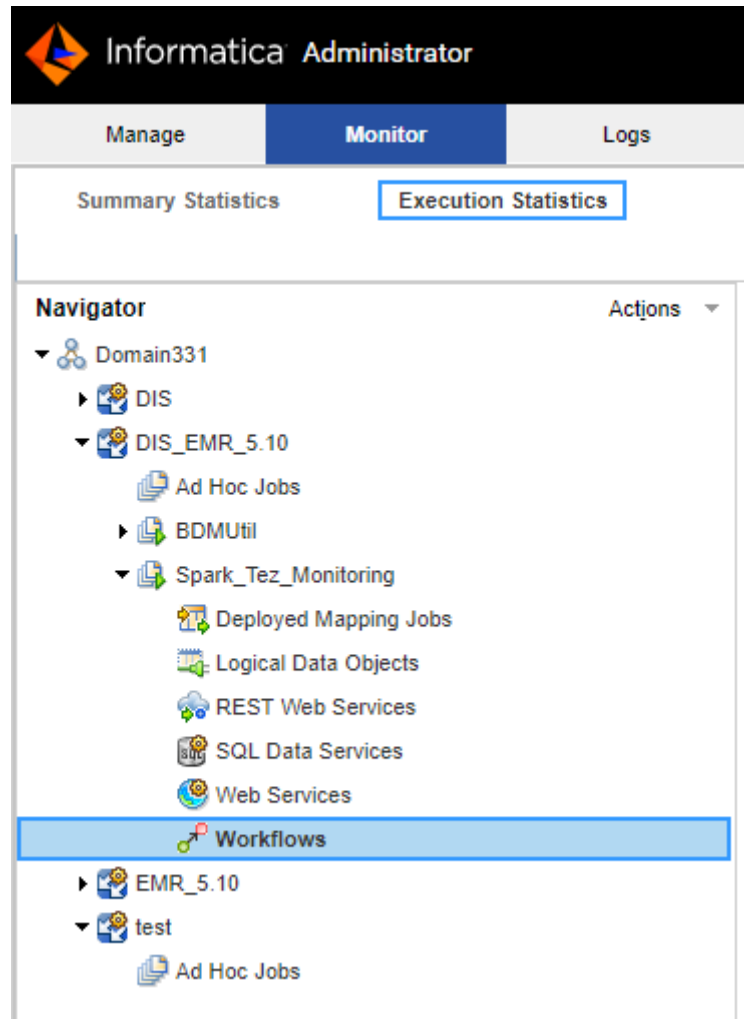
m_Ustx_Readback - Egk2Jjm0Eeil... Properties Hive Execution Plan Summary Statistics Detailed Statistics Historical Statistics

✓ The Deployed Mapping job has completed.

General Properties

Name	m_Ustx_Readback
Type	Deployed Mapping
Started By	Administrator
User Security Domain	Native
Start Time	04/06/2018 21:33:39
Elapsed Time	00:08:17
End Time	04/06/2018 21:41:57

- ワークフローを選択するには、アプリケーションを展開して【ワークフロー】をクリックします。



ジョブのリストがコンテンツパネルに表示されます。

- ジョブをクリックしてプロパティを表示します。

コンテンツパネルには、ジョブに対するデフォルトの【プロパティ】ビューが表示されます。Blaze エンジンのマッピングの場合は、詳細パネルの全般プロパティに、Blaze エンジンの監視 URL が表示されます。監視 URL は、Spark ジョブの YARN Web ユーザーインターフェースへのリンクです。

- コンテンツパネルでビューを選択して、ジョブの詳細情報を表示します。
 - マッピングの実行プランを表示するには、【実行プラン】ビューを選択します。
 - ジョブのサマリ統計を表示するには、【サマリ統計】ビューをクリックします。
 - ジョブの詳細統計を表示するには、【詳細統計】ビューをクリックします。

注: Hive ソースまたはターゲットについて、処理済みの行の数を【サマリ統計】で参照できます。Hive ソースとターゲットのその他の値は表示されません。

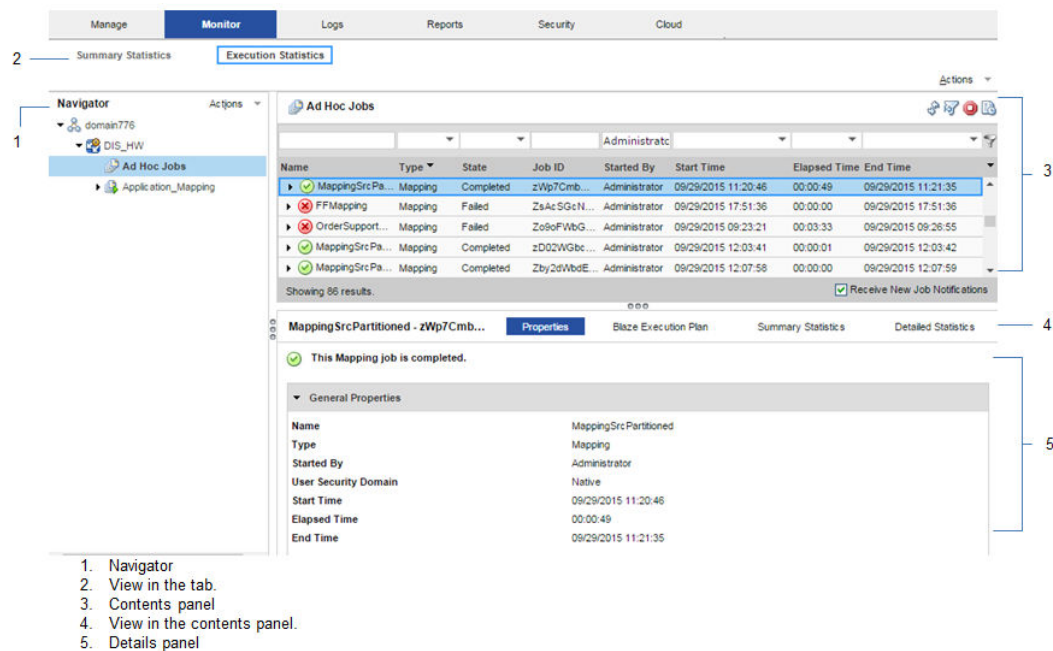
次の表は、Administrator ツールにある [コンテンツ] パネルのマッピングジョブの状態を示しています。

ジョブステータス	規則およびガイドライン
キューに格納	ジョブがキュー内にあります。
実行中	データ統合サービスはジョブを実行中です。
完了	ジョブは正常に実行されました。
強制終了	再開時にジョブがキューからフラッシュされたか、ジョブの実行中にノードが予期せずシャットダウンしました。
失敗	実行中にジョブが失敗したか、キューがいっぱいです。
キャンセル済み	ジョブがキューから削除されたか、実行中にキャンセルされました。
不明	ジョブのステータスは不明です。

Blaze エンジンの監視

Administrator ツールの [モニタ] タブで、Blaze エンジンマッピングジョブの統計情報を監視し、ログイベントを表示できます。Blaze エンジンのマッピングジョブは、Blaze ジョブ監視 Web アプリケーションでも監視できます。

次の図に、Administrator ツールの [モニタ] タブを示します。



[モニタ] タブには、次のビューが含まれています。

サマリ統計

【サマリ統計】ビューを使用すると、データ統合サービスでのオブジェクトの状態および分布のサマリをグラフで表示できます。データ統合サービスでオブジェクトを実行するために使用されたメモリおよび CPU のグラフを表示することもできます。

実行統計

【実行統計】ビューを使用すると、プロパティ、ランタイム統計、およびランタイムレポートを監視できます。ナビゲータで「データ統合サービス」を展開して【アドホックジョブ】を監視する、またはアプリケーションを展開してデプロイ済みのマッピングジョブやワークフローを監視できます。

【アドホックジョブ】、デプロイ済みのマッピングジョブ、ワークフローを【実行統計】ビューのナビゲータのアプリケーションから選択すると、コンテンツパネルにジョブのリストが表示されます。コンテンツパネルに、キューに追加された、実行中、完了、失敗、中断、キャンセルの状態にあるジョブが表示されます。データ統合サービスは、十分なリソースを使用できるときに、キューに追加された状態のジョブをクラスタに送信します。

コンテンツパネルでは、ジョブタイプに基づいて、関連ジョブがグループ化されています。ジョブタイプを展開して、その下の関連ジョブを表示できます。

【実行統計】ビューで、次のビューにアクセスします。

プロパティ

【プロパティ】ビューに、選択したジョブについて、名前、ジョブタイプ、ジョブを開始したユーザー、ジョブの開始時刻など、全般的なプロパティが表示されます。Hadoop クラスタでのジョブは、全般プロパティのマッピングに対して表示される監視 URL から監視できます。監視 URL を使用すると、Web ページに Blaze ジョブ監視が開きます。Blaze ジョブ監視には、グリッドタスク数、グリッドセグメント数、タスクレット数、各タスクレットに対して試行されたりカバリの回数などのマッピングについての詳細な監視統計が表示されます。

Blaze 実行プラン

Blaze 実行プランには、データ統合サービスがマッピングロジックに基づいて生成する Blaze エンジン スクリプトが表示されます。実行プランには、このスクリプトが依存するタスクが含まれます。各スクリプトに一意的識別子があります。

サマリ統計

コンテンツパネルでマッピングジョブを選択すると、詳細パネルに【サマリ統計】ビューが表示されます。【サマリ統計】ビューには、ジョブのスループットとリソース使用状況の統計が表示されます。

ジョブのスループットに関する以下の統計を表示できます。

- ソース。マッピングソースファイルの名前。
- ターゲット名。ターゲットファイルの名前。
- 行。読み取られたソースおよびターゲットの行数。ターゲットが Hive の場合、使用できるサマリ統計はこの統計のみです。
- 平均行/秒。1 秒間に読み取られたソースおよびターゲットの行数の平均。
- バイト。読み取られたソースおよびターゲットのバイト数。
- 平均バイト/秒。1 秒間に読み込まれたソースおよびターゲットのバイト数の平均。
- 先頭行へのアクセス。データ統合サービスがソースファイルの最初の行の読み取りを開始した日時。
- 削除された行。データ統合サービスが読み取らなかったソース行の数。

詳細統計

コンテンツパネルでマッピングジョブを選択すると、詳細パネルに【詳細統計】ビューが表示されます。【詳細統計】ビューには、ジョブ実行のスループットとリソース使用状況の統計がグラフで表示されます。

Blaze ジョブ監視アプリケーション

Blaze ジョブ監視アプリケーションを使用して、Hadoop クラスタの Blaze エンジンジョブを監視します。

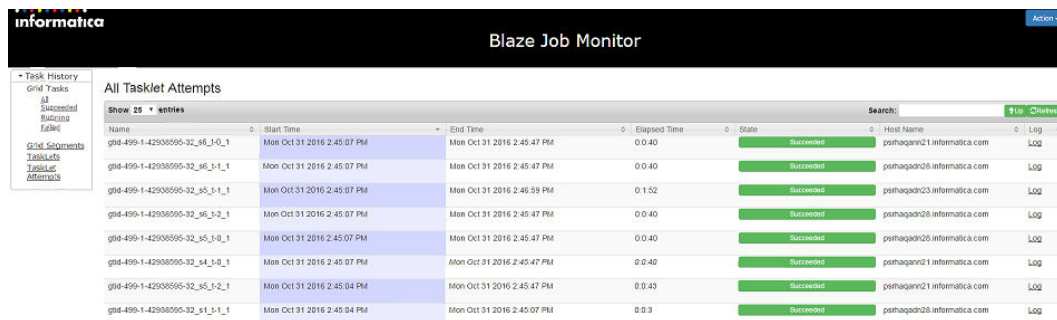
Blaze ジョブ監視を開始するホストは、Hadoop 接続プロパティで構成します。Blaze ジョブ監視アドレスは、同じクラスタ上の他のユーザーとの競合を防止するように設定することや、アクセス権限を持つノード数が制限されている場合に設定する必要があることがあります。Blaze ジョブ監視アドレスを設定しない場合、グリッドマネージャはアルファベットの最初のクラスタノード上で、デフォルトポートである 9080 を使用してホストを開始します。

Blaze エンジンのマッピングジョブを表示すると、Blaze エンジン監視 URL が、Administrator ツールの [モニタ] タブに表示されます。この URL をクリックすると、Web ページに Blaze エンジンの監視アプリケーションが開きます。

注: Blaze ジョブ監視には、LDTM ログを介してアクセスすることもできます。セッションがサマリをロードした後は、ログにグリッドタスク内のセグメントのリストが表示されます。各セグメントに、Blaze ジョブ監視へのリンクが含まれます。リンクをクリックすると、そのセグメントの実行の詳細が表示されます。

Blaze ジョブ監視を開始するホストは、Hadoop 接続プロパティで構成します。デフォルトアドレスは <hostname>:9080 です。

次の図に、Blaze ジョブ監視を示します。



Name	Start Time	End Time	Elapsed Time	State	Host Name	Log
g98-499-1-42938595-32_s6_3-0_1	Mon Oct 31 2016 2:45:07 PM	Mon Oct 31 2016 2:45:47 PM	0:0:40	Succeeded	psishaqan21.informatica.com	Log
g98-499-1-42938595-32_s6_3-1_1	Mon Oct 31 2016 2:45:07 PM	Mon Oct 31 2016 2:45:47 PM	0:0:40	Succeeded	psishaqan28.informatica.com	Log
g98-499-1-42938595-32_s6_3-1_1	Mon Oct 31 2016 2:45:07 PM	Mon Oct 31 2016 2:46:59 PM	0:1:52	Succeeded	psishaqan23.informatica.com	Log
g98-499-1-42938595-32_s6_3-2_1	Mon Oct 31 2016 2:45:07 PM	Mon Oct 31 2016 2:45:47 PM	0:0:40	Succeeded	psishaqan28.informatica.com	Log
g98-499-1-42938595-32_s6_3-0_1	Mon Oct 31 2016 2:45:07 PM	Mon Oct 31 2016 2:45:47 PM	0:0:40	Succeeded	psishaqan25.informatica.com	Log
g98-499-1-42938595-32_s4_3-0_1	Mon Oct 31 2016 2:45:07 PM	Mon Oct 31 2016 2:45:47 PM	0:0:40	Succeeded	psishaqan21.informatica.com	Log
g98-499-1-42938595-32_s6_3-2_1	Mon Oct 31 2016 2:45:04 PM	Mon Oct 31 2016 2:45:47 PM	0:0:43	Succeeded	psishaqan21.informatica.com	Log
g98-499-1-42938595-32_s1_3-1_1	Mon Oct 31 2016 2:45:04 PM	Mon Oct 31 2016 2:45:07 PM	0:0:3	Succeeded	psishaqan28.informatica.com	Log

左側のタスク履歴パネルを使用して、Blaze マッピングジョブを次の条件別にフィルタ処理できます。

- グリッドタスク。Blaze エンジン実行者がグリッドマネージャに送信した並行処理ジョブ要求。すべてのタスク、成功したタスク、実行中のタスク、失敗したタスク別にフィルタをさらに適用できます。
- グリッドセグメント。グリッドタスクに含まれるグリッドマッピングの一部。
- タスクレット。個別の DTM で実行されるグリッドセグメントのパーティション。
- タスクレット試行回数。タスクレットを再起動するためのリカバリ試行回数。【ログ】をクリックして、マッピンググリッドタスクログを表示します。

Blaze ジョブ監視に、同じ名前空間に属するマッピングジョブのタスク履歴が表示されます。タスクの開始時刻、終了時刻、経過時間、状態などのタスクのプロパティを監視できます。ログイベントを表示することもできます。マッピングジョブをグリッドセグメントでフィルタ処理した場合、グリッドセグメントの上にマウスを移動すると、セグメントの論理名が表示されます。

デフォルトでは、Blaze ジョブ監視はタスクのリストを 5 秒ごとに自動的に更新し、タスクを表示する最初のページに戻ります。複数のページを参照する場合、自動更新を無効にします。自動更新をオフにするには、【アクション】 > 【自動更新オフ】をクリックします。

Blaze ジョブ監視に、過去 7 日間に実行された最初の 100,000 件のグリッドタスクが表示されます。Blaze ジョブ監視に、実行中のグリッドタスクのグリッドセグメント数、タスクレット数、グリッドタスクに対して試行されたタスクレットの回数と、最近の 30 分間にアクセスされたグリッドタスクが表示されます。

Blaze サマリレポート

Blaze サマリレポートには、マッピングジョブに関する詳細な統計が表示されます。Blaze ジョブ監視で、成功したグリッドタスク名の横に緑色のサマリレポートボタンが表示されます。このボタンをクリックして Blaze サマリレポートを開きます。

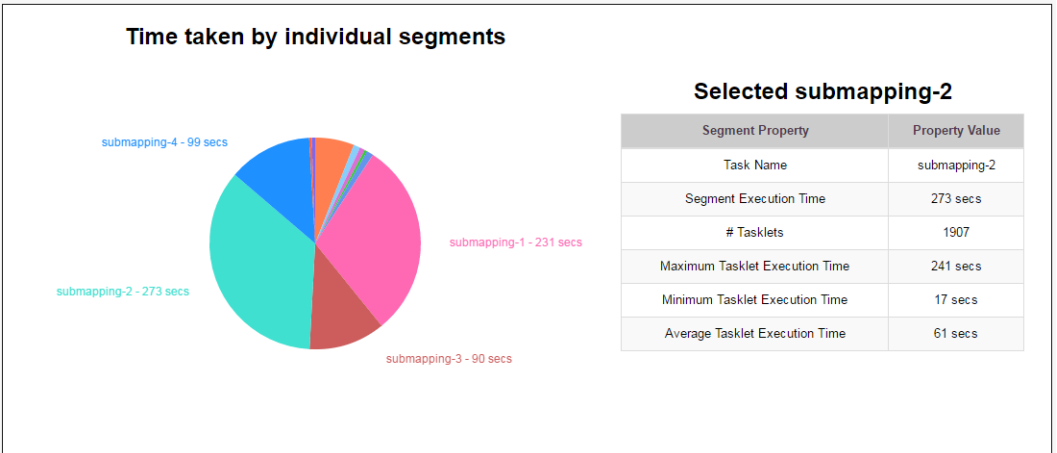
注: Blaze サマリレポートはテクニカルレビュー用に使用できます。テクニカルレビュー機能はサポートされていますが、保証対象外で本番環境には対応していません。非本番環境でのみ使用することをお勧めします。

All Grid Tasks

Show 25 entries					
Name	Start Time	End Time	Elapsed Time	State	
gtid-476-1-36233666-2	Mon Oct 17 2016 1:32:42 PM	Mon Oct 17 2016 1:53:18 PM	0:20:36	Succeeded	
gtid-476-1-36233666-1	Summary Report 2016 1:30:51 PM	Mon Oct 17 2016 1:31:20 PM	0:0:28	Succeeded	
gtid-441-1-26795155-4	Mon Oct 17 2016 11:55:06 AM	Mon Oct 17 2016 11:59:44 AM	0:4:37	Succeeded	
gtid-441-1-26795155-3	Mon Oct 17 2016 11:47:10 AM	Mon Oct 17 2016 11:51:51 AM	0:4:40	Succeeded	
gtid-441-1-26795155-2	Mon Oct 17 2016 11:02:37 AM	Mon Oct 17 2016 11:06:18 AM	0:3:40	Succeeded	
gtid-441-1-26795155-1	Mon Oct 17 2016 10:53:35 AM	Mon Oct 17 2016 10:54:27 AM	0:0:51	Failed	
gtid-437-1-25270758-1	Mon Oct 17 2016 10:28:08 AM	Mon Oct 17 2016 10:28:56 AM	0:0:47	Failed	

個々のセグメントで費やされる時間

円グラフは、1つのグリッドタスク内に含まれる個々のセグメントで費やされた時間を、視覚的に表します。



円グラフの特定のセグメントをクリックすると、そのセグメントに関する詳細情報が**選択したサブマッピング**のテーブルに表示されます。テーブルには、次のセグメントの統計情報が表示されます。

- タスク名。選択したセグメントの論理名。
- セグメントの実行時間。選択したセグメントで費やされた時間。
- タスクレット数。選択したセグメントに含まれるタスクレットの数。
- タスクレットの最短実行時間。選択したセグメント内で、実行に最も短い時間がかかったタスクレットの実行時間。
- タスクレットの最長実行時間。選択したセグメント内で、実行に最も長い時間がかかったタスクレットの実行時間。
- タスクレットの平均実行時間。選択したセグメントに含まれるすべてのタスクレットの平均実行時間。

マッピングプロパティ

マッピングプロパティテーブルには、マッピングジョブに関する基本的な情報が一覧表示されます。

Mapping Properties	
Mapping Property	Property Value
DIS Name	dis_cdh
Informatica Version	10.1.1
Mapping Name	q97_hive_mapping
Total Segments	13
Maximum Segment Execution Time	273 secs
Minimum Segment Execution Time	0 secs
Average Segment Execution Time	59 secs
Mapping Execution Time	0:10:14

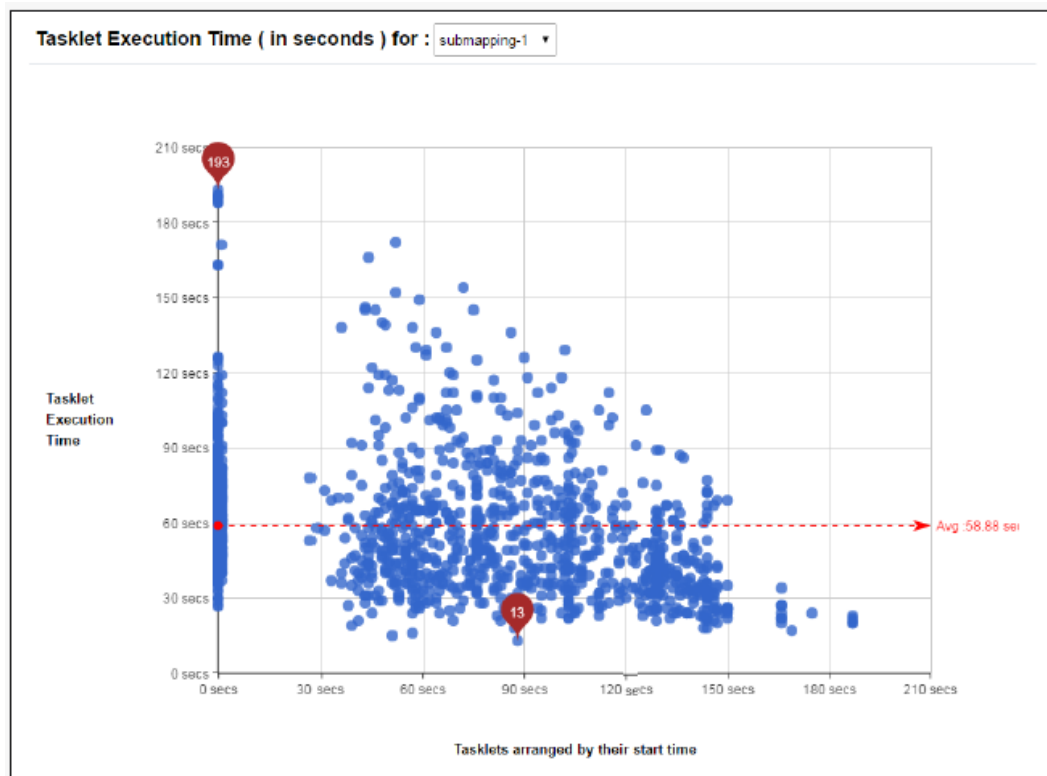
マッピングプロパティテーブルは次の情報を表示します。

- マッピングが実行された DIS 名。
- Informatica バージョン。
- マッピングの名前。
- マッピングの合計セグメント数。
- 実行に最も長い時間がかかったセグメントの実行時間。
- 実行に最も短い時間がかかったセグメントの実行時間。
- すべてのセグメントの平均実行時間。
- マッピングの合計実行時間。

タスクレットの実行時間

時系列グラフに、選択したセグメント内のすべてのタスクレットの実行時間が表示されます。

X 軸はタスクレットの開始時間を表し、Y 軸はタスクレットの実際の実行時間を表します。赤い破線はすべてのタスクレットの平均実行時間を表し、2 つの赤いマーカーはセグメント内の最短実行時間と最長実行時間を示します。



選択したタスクレットの情報

タスクレットの実行時間グラフからタスクレットを選択すると、その個別のタスクレットに関してより詳細な情報を参照できます。このデータには、ソースとターゲットの行数と、タスクレットが処理するキャッシュベ

ースのトランスフォーメーションがある場合はそのキャッシュ情報が示されます。選択したタスクレットの完全なログを参照するには、[詳細ログの取得] ボタンをクリックします。

Selected tasklet `gtid-299-1-82064486-16_s8_t-394_1`
[Get Detailed Log](#)

Show entries Search:

Source Name	# Rows Processed
empty_source1	0
Read_catalog_sales	4,937,484

Showing 1 to 2 of 2 entries Previous Next

Show entries Search:

Target Name	# Rows Processed
DETarget_Aggregator1_G0	0

Showing 1 to 1 of 1 entries Previous Next

Show entries Search:

Transformation Name	Index Cache (bytes)		Data Cache (bytes)	
	Configured	Used	Configured	Used
Joiner1	178,956,970	14,400	357,913,940	6,968

Showing 1 to 1 of 1 entries Previous Next

Row counts for all sources processed by tasklet

Row counts for all targets written by tasklet

Cache information for any cache-based transformation processed by tasklet

Blaze エンジンのログ

Blaze エンジンでのマッピング実行に関する情報は、LDTM ログファイルで見つかります。

LDTM は、Blaze エンジンでのマッピング実行の結果をログに記録します。LDTM ログは、マッピングジョブに関して Developer tool または Monitoring ツールから表示できます。

マッピング実行に関し、セッションログにログの詳細を記録するように、データ統合サービスを設定できます。LDTM マッピング実行の詳細のログ記録を有効にするには、ログのトレースレベルを [詳細 - 初期化] または [詳細 - データ] に設定します。ロギングのパフォーマンスを最適にするには、トレースレベルを [詳細 - データ] に設定しないようにします。マッピング実行をデバッグする必要がある場合に、トレースレベルを [詳細 - データ] に設定できます。

論理マッピング実行の詳細には、次の情報が含まれます。

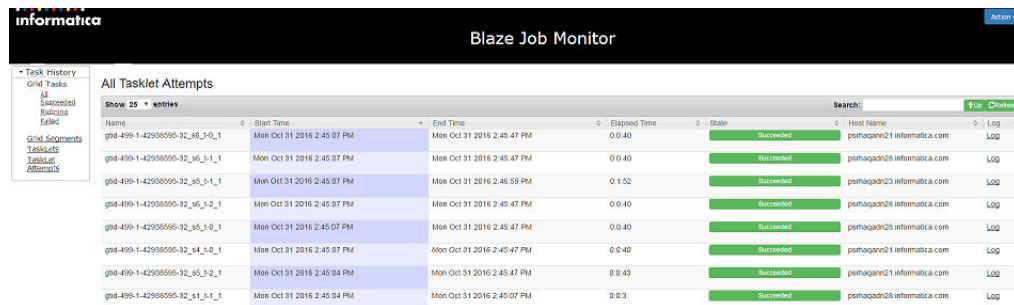
- 各タスクの開始時刻、終了時刻、および状態
- Blaze ジョブ監視の URL
- タスクレット総数、成功したタスクレット数、失敗したタスクレット数、およびキャンセルされたタスクレット数
- ソースおよびターゲットの処理された行数と拒否された行数
- 実行された各セグメントのトランスフォーメーションのデータエラー。データエラーが発生した場合に含まれます

マッピング実行の詳細には、実行された各セグメントのパーティション数などの、Blaze エンジンライントタイム処理に関する情報が含まれます。

Blaze ログの表示

Blaze マッピングのログは、Blaze ジョブ監視から表示できます。

1. Blaze ジョブ監視で、ジョブのリストからジョブを選択します。
2. 選択したジョブの行で、**[ログ]** リンクをクリックします。



The screenshot shows the 'Blaze Job Monitor' interface. On the left is a sidebar with navigation links: Task History, Grid Tasks, All, Succeeded, Running, Failed, Grid Segments, Tasks, Taskset, Attempts. The main area is titled 'All Taskset Attempts' and contains a table with columns: Name, Start Time, End Time, Elapsed Time, State, Host Name, and Log. The table lists several task attempts, all with a 'Succeeded' state. A search bar and a 'Show 25 entries' dropdown are at the top right of the table.

Name	Start Time	End Time	Elapsed Time	State	Host Name	Log
gnd-499-1-42908095-32_s5_1-0_1	Mon Oct 31 2016 2:45:07 PM	Mon Oct 31 2016 2:45:47 PM	0:0:40	Succeeded	psrhagann21.informatica.com	Log
gnd-499-1-42908095-32_s5_1-1_1	Mon Oct 31 2016 2:45:07 PM	Mon Oct 31 2016 2:45:47 PM	0:0:40	Succeeded	psrhagann28.informatica.com	Log
gnd-499-1-42908095-32_s5_1-1_1	Mon Oct 31 2016 2:45:07 PM	Mon Oct 31 2016 2:45:59 PM	0:1:52	Succeeded	psrhagann23.informatica.com	Log
gnd-499-1-42908095-32_s5_1-2_1	Mon Oct 31 2016 2:45:07 PM	Mon Oct 31 2016 2:45:47 PM	0:0:40	Succeeded	psrhagann28.informatica.com	Log
gnd-499-1-42908095-32_s5_1-0_1	Mon Oct 31 2016 2:45:07 PM	Mon Oct 31 2016 2:45:47 PM	0:0:40	Succeeded	psrhagann25.informatica.com	Log
gnd-499-1-42908095-32_s4_1-0_1	Mon Oct 31 2016 2:45:07 PM	Mon Oct 31 2016 2:45:47 PM	0:0:40	Succeeded	psrhagann21.informatica.com	Log
gnd-499-1-42908095-32_s5_1-2_1	Mon Oct 31 2016 2:45:04 PM	Mon Oct 31 2016 2:45:47 PM	0:0:43	Succeeded	psrhagann21.informatica.com	Log
gnd-499-1-42908095-32_s1_1-1_1	Mon Oct 31 2016 2:45:04 PM	Mon Oct 31 2016 2:45:07 PM	0:0:3	Succeeded	psrhagann28.informatica.com	Log

ログイベントが、別のブラウザウィンドウに表示されます。

オーケストレータのサンセット時間

オーケストレータのサンセット時間は、オーケストレータサービスの最大存続期間です。サンセット時間によって、Blaze エンジンがマッピングジョブを実行できる最長の時間が決まります。デフォルトのサンセット時間は 24 時間です。24 時間が経過すると、オーケストレータはシャットダウンし、その結果 Blaze グリッドマネージャもシャットダウンされます。

オーケストレータのサンセット時間は、24 時間よりも長くまたは短く設定できます。Hadoop 接続で次のプロパティを設定します。

プロパティ	説明
infagrid.orchestrator.svc.sunset.time	オーケストレータサービスの最長存続期間（時間単位）。デフォルトは 24 時間です。

プロパティを 0（ゼロ）またはマイナスの値に設定して、サンセットを無効にすることもできます。サンセットを無効にすると、オーケストレータがマッピングの実行中にシャットダウンされることはありません。

Blaze 監視のトラブルシューティング

Blaze エンジンでマッピングを実行してグリッドタスクログを表示しようとする、Blaze ジョブ監視が完全なログを取得しない。

グリッドタスクログが大きすぎる可能性があります。Blaze ジョブ監視が取得できるのは、集計後のログで最大で 2 MB までです。ログの最初の行にこの情報がレポートされており、HDFS 上の完全なログの場所が示されています。HDFS へのリンクをたどって、グリッドマッピングの集計後のログを検索します。完全なログへのリンクにはグリッドタスク番号が含まれます。

Blaze ジョブ監視が開始されない。

Hadoop 環境のログを確認して、問題を特定します。問題が見つからない場合は、`infacmd stopBlazeService` コマンドを使用してグリッドマネージャを停止し、マッピングを再度実行します。

監視 URL が Administrator ツールの [プロパティ] ビューに表示されない。

YARN ログで URL を特定します。

Blaze プロセスが予期せず停止した場合に、Blaze が予定された場所にログを保存しない。

Blaze が予期せずに停止した場合は、YARN モニタを通じて Blaze サービスログにアクセスできます。次の方法のいずれかを使用します。

- グリッドマネージャのログには、すべての Blaze ジョブコンテナ ID が含まれており、Blaze が実行されたホストを識別できます。グリッドマネージャのログ URL を、Blaze ホストのコンテナ ID とホスト名を使用して変更します。
- コマンド `yarn logs -applicationID <Blaze グリッドマネージャアプリケーション ID>` を実行します。

数日間実行されていた Blaze ジョブ監視が、Hortonworks クラスタのアプリケーションタイムラインサーバーへの接続を失った。

Blaze エンジンには、クラスタ上で実行中のアプリケーションタイムラインサーバーを必要とします。Blaze エンジンがマッピングの実行を開始すると、Blaze ジョブ監視はアプリケーションタイムラインサーバーの状態を確認します。アプリケーションタイムラインサーバーが実行されていない場合は、グリッドマネージャがそれを開始します。アプリケーションタイムラインサーバーへの接続が失われると、Blaze エンジンでは再接続を試行します。アプリケーションタイムラインサーバーが Blaze マッピングの実行中に停止した場合は、グリッドマネージャを再起動することでタイムラインサーバーを再起動できます。

注: アプリケーションタイムラインサーバーがクラスタで実行されるようにデフォルトで設定されていない場合、クラスタ管理者はタイムラインサーバーをクラスタで手動で再起動する必要があります。

マッピングの実行に 24 時間より長い時間がかかる場合に、マッピングが失敗する。

マッピングが Blaze エンジンで 24 時間より長く実行されると、一部のマッピングが失敗することがあります。これは、オーケストレータサービスのデフォルトサンセット時間が 24 時間であるためです。24 時間が経過すると、オーケストレータはシャットダウンし、その結果 Blaze グリッドマネージャもシャットダウンされます。

サンセット時間を 24 時間よりも長くするには、Hadoop 接続の詳細プロパティで次のプロパティを設定します。

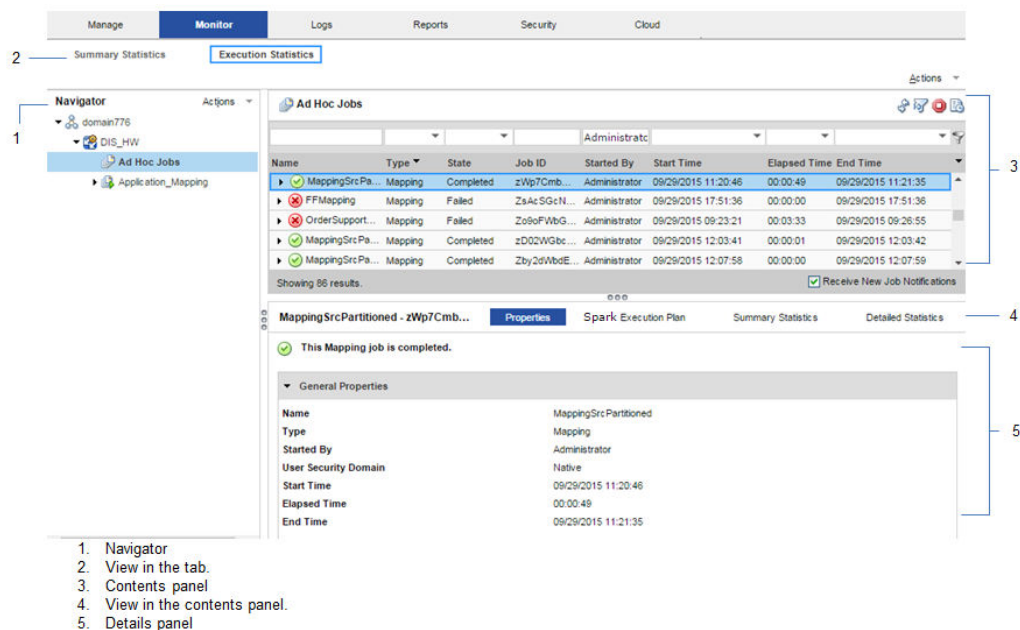
```
infagrid.orchestrator.svc.sunset.time=[HOURS]
```

プロパティを 0（ゼロ）またはマイナスの値に設定して、サンセットを無効にすることもできます。サンセットを無効化すると、Blaze グリッドマネージャはシャットダウンしません。

Spark エンジン監視

Administrator ツールの [モニタ] タブで、Spark エンジンマッピングジョブの統計情報を監視し、ログイベントを表示できます。Spark エンジンのマッピングジョブは、YARN Web ユーザーインターフェースでも監視できます。

次の図に、Administrator ツールの [モニタ] タブを示します。



「モニタ」タブには、次のビューが含まれています。

サマリ統計

【サマリ統計】ビューを使用すると、データ統合サービスでのオブジェクトの状態および分布のサマリをグラフで表示できます。データ統合サービスでオブジェクトを実行するために使用されたメモリおよび CPU のグラフを表示することもできます。

実行統計

【実行統計】ビューを使用すると、プロパティ、ランタイム統計、およびランタイムレポートを監視できます。ナビゲータで「データ統合サービス」を展開して「アドホックジョブ」を監視する、またはアプリケーションを展開してデプロイ済みのマッピングジョブやワークフローを監視できます。

【アドホックジョブ】、デプロイ済みのマッピングジョブ、ワークフローを【実行統計】ビューのナビゲータのアプリケーションから選択すると、コンテンツパネルにジョブのリストが表示されます。コンテンツパネルに、キューに追加された、実行中、完了、失敗、中断、キャンセルの状態にあるジョブが表示されます。データ統合サービスは、十分なリソースを使用できるときに、キューに追加された状態のジョブをクラスタに送信します。

コンテンツパネルでは、ジョブタイプに基づいて、関連ジョブがグループ化されています。ジョブタイプを展開して、その下の関連ジョブを表示できます。

【実行統計】ビューで、次のビューにアクセスします。

プロパティ

【プロパティ】ビューに、選択したジョブについて、名前、ジョブタイプ、ジョブを開始したユーザー、ジョブの開始時刻など、全般的なプロパティが表示されます。

Spark 実行プラン

マッピングの Spark 実行プランを表示するときに、データ統合サービスは、マッピングを Scala プログラムと任意選択のコマンドセットに変換します。実行プランによって、これらのコマンドと Scala プログラムコードが表示されます。

サマリ統計

コンテンツパネルでマッピングジョブを選択すると、詳細パネルに【サマリ統計】ビューが表示されます。【サマリ統計】ビューに、次のジョブのスループット統計が表示されます。

- プレジョブタスク Spark ジョブ実行前にソースデータを読み取り、行データを一時テーブルにステージングする各ジョブタスクの名前。各秒に処理されたバイト数と平均バイト数を表示することもできます。
- ソース。ソースの名前。
- ターゲット。ターゲットの名前。
- 行。ソースでは、Spark アプリケーションによって読み取られた行数。ターゲットでは、ターゲットおよび拒否行に書き込まれた行の総数。
- ポストジョブタスク。ステージングされたテーブルからターゲットデータを書き込む各ジョブタスクの名前。各秒に処理されたバイト数と平均バイト数を表示することもできます。

マッピングに複数のアップストリームソースを含む共有体トランスフォーメーションが含まれる場合、ソースは、[ソース] の下の 1 行にカンマ区切りリストの形式で表示されます。

注: DD_UPDATE 条件を含むアップデイトストラテジトランスフォーメーションを含む Hive マッピングでは、Spark ジョブの実行後、ターゲットに一時テーブルのみが含まれています。マッピングジョブ統計の結果は、ポストジョブタスクに表示され、更新されたレコード数の 2 倍を示します。

次の図に、Spark エンジンで実行されるマッピングの詳細パネル内の【サマリ統計】ビューを示します。

The screenshot shows the Informatica Administrator interface. The 'Monitor' tab is selected, and the 'Execution Statistics' sub-tab is active. The 'Ad Hoc Jobs' section is expanded, showing a list of jobs. The 'SparkOracle' job is selected, and the 'Summary Statistics' view is displayed. The view shows a table with columns: Name, Type, State, Job ID, Started By, Start Time, Elapsed Time, and End Time. The table lists various tasks including SparkOracle, Post_UserDefinedJob_Task_1, and Pre_UserDefinedJob_Task_1. The Summary Statistics section is highlighted with a red circle, showing details for the SparkOracle task, including Rows, Average Rows/Sec, Bytes, and Average Bytes/Sec.

Spark の実行ステージ情報を、[実行統計] の [モニタ] タブの【サマリ統計】ビューの詳細ペインで参照することもできます。ソースの後、ターゲットの前にリストとして表示されます。

Spark 実行ステージに、Spark アプリケーションのステージ統計と関連する行とバイトの絶対数とスループットが表示されます。行はステージが書き込む行数を表し、バイトはステージ内でブロードキャストされるバイト数を表します。

次の図に、Spark 実行ステージを示します。

The screenshot shows the Informatica Ad Hoc Jobs interface. The top table lists several jobs, including 'q3_hive_mapping_default_with_monitor'. Below this, the 'Summary Statistics' tab is selected for the job 'q3_hive_mapping_default_with_monitor - NaW4rCwCEiXdpGN8ZxuTw'. The 'Throughput' section shows source and target statistics. The 'Spark Run Stages' section shows a table of Spark stages.

Name	Type	State	Job ID	Started By	Start Time	Elapsed Time	End Time
q3_hive_mapping_default_with_monitor	Mapping	Completed	TdyKyvXEv5BH0Ym...	Administrator	03/20/2018 13:48:43	00:03:49	03/20/2018 13:52:32
lgnh_1	Mapping	Completed	ES7y-4uXEv5BH0Ym...	Administrator	03/20/2018 13:47:06	00:03:56	03/20/2018 13:51:03
q3_hive_mapping_default_with_monitor	Mapping	Completed	DV1L-SuTEwQzHyJMPQ1...	Administrator	03/20/2018 13:16:20	00:03:52	03/20/2018 13:22:13
lgnh_1	Mapping	Completed	LJ31bCwREwQzHyJMPQ1Vw	Administrator	03/20/2018 13:04:53	00:03:19	03/20/2018 13:08:13
hnhf_filter	Mapping	Completed	AMSRhuwHEwQzHyJMPQ1...	Administrator	03/20/2018 11:52:02	00:02:09	03/20/2018 11:54:12

Source	Rows	Average Rows/Sec
Read_dsm	52000	
Read_dsm	73049	1974
Read_dsm	115203420	294000

Target	Rows	Average Rows/Sec
Write_tgt_d3	144	144

Spark Run Stages	Rows	Average Rows/Sec	Bytes	Average Bytes/Sec
Stage_0_infaspark0	73049	1974	518000	14000
Stage_1_infaspark0	73049	N/A	518000	N/A
Stage_2_infaspark0	52000	52000	500240	500240

例えば、Spark 実行ステージカラムには、stage_<ID>で始まる Spark アプリケーションのステージングされた情報が格納されます。例では、Stage_0 に、ID=0 の値を持つ Spark アプリケーションの Spark 実行ステージと関連する統計が表示されています。

Spark エンジンが、詳細データが有効化された自己結合を含むソースデータを読み取る場合を考えます。このシナリオでは、Spark アプリケーションから最適化されたマッピングには、Spark エンジンログ内の同じソースの 2 番目のインスタンスの情報は何も含まれません。

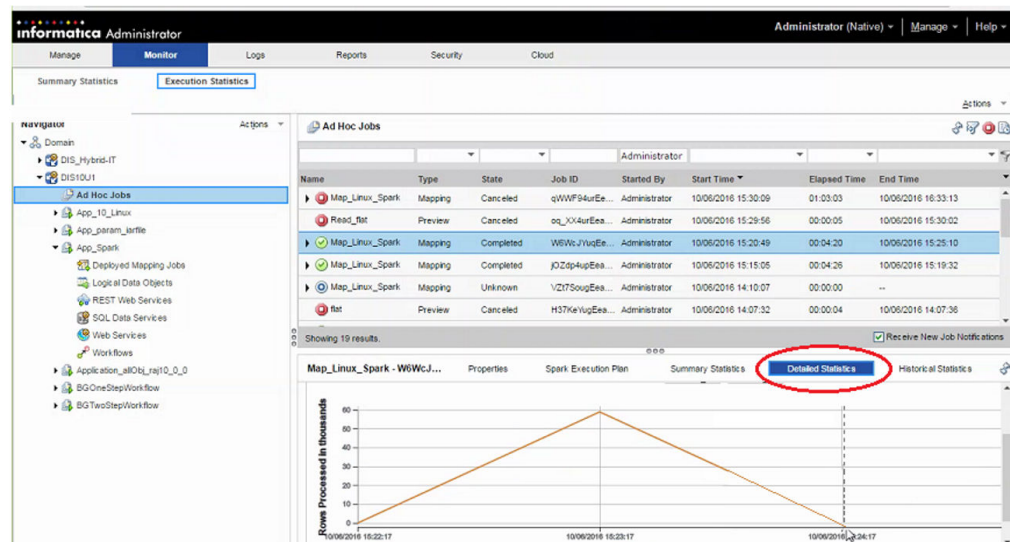
一時テーブルからのデータ読み取るときに、カスタマイズしたデータオブジェクトの Hive クエリによって、データのシャッフルが発生する場合を考えます。このシナリオでは、フィルタされたソース統計が、Spark エンジンログ内の一時ソーステーブルからの読み取りの代わりに表示されます。

Spark 監視が有効な状態でマッピングを実行するときは、パフォーマンスは、マッピングの複雑さに基づいて異なります。監視が有効な状態では、通常の処理時間の最大 3 倍の時間がかかる可能性があります。デフォルトでは、監視は無効です。

詳細統計

コンテンツパネルでマッピングジョブを選択すると、詳細パネルに【詳細統計】ビューが表示されます。【詳細統計】ビューに、ジョブ実行の行数のグラフが表示されます。

次の図に、Spark エンジンで実行されるマッピングの詳細パネル内の【詳細統計】ビューを示します。



Hive タスクの表示

トランザクションテーブルを含む Hive ソースがある場合、Spark ジョブに関連する Hive タスクを表示できます。

Hive タスクを起動するマッピングを Spark 上で実行する場合、セッションログと Administrator ツールで、Spark アプリケーションと一緒に、Hive クエリの統計を監視のために表示できます。例えば、アップデイトストラジトランスフォーメーションに関連する情報と、Spark 上のマッピングに関連付けられた SQL 認証を監視できます。

Hive タスクのサマリ統計を Administrator ツールで表示できます。Spark 統計は引き続き表示されます。Spark エンジンが Hive タスクを起動すると、Hive タスクの統計と Spark データフレームを含む、ソースロードの要約とターゲットロードの要約を表示できます。これとは異なり、Spark タスクのみがある場合、セッションログにはソースロードの要約とターゲットロードの要約は表示されません。

ターゲットロードの要約で、すべての Hive インスタンスには「Hive_Target_」というプレフィックスが付加されます。同じインスタンス名は、Administrator ツールにも表示されます。

Spark エンジンが次のいずれかのシナリオで起動する Hive テーブルとの間で読み書きする際に、Administrator ツールで Tez ジョブ統計を表示できます。

- AmazonBAKELT を表すリソースがある。
- トランザクショナル Hive テーブルがある。
- 厳密に定義された認証を使用して保護されたテーブルカラムがある。

すべての Hive ソースおよびターゲットに関し、各秒の平均行数、バイト数、各秒の平均バイト数、拒否された行数にゼロの行を示す誤った統計情報が表示されます。処理された行のみに、正しい値が含まれており、その他のカラムに 0 または N/A が含まれることがわかります。

Spark エンジンのログ

Spark エンジンのログは、LDTM ログに表示されます。LDTM は、マッピングに対して実行された Spark エンジンの実行プランの結果を記録します。LDTM ログは、マッピングジョブに関して Developer tool または Monitoring ツールから表示できます。

Spark エンジンのログには、マッピングを内部形式に変換する手順、マッピングの最適化手順、マッピングを Spark コードにレンダリングする手順、コードを Spark 実行者に送信する手順が示されます。また、論理データの変換ジェネレータがマッピングロジックから作成するスカラコードも、ログに表示されます。

Sqoop マッピングを Spark エンジンで実行すると、データ統合サービスによって Sqoop ログイベントがマッピングログに出力されます。

Spark ログの表示

Spark マッピングのログを、YARN Web ユーザーインターフェースから表示できます。

1. YARN Web ユーザーインターフェースで、表示するアプリケーションの ID をクリックします。
2. アプリケーションの **【詳細】** をクリックします。
3. アプリケーションの詳細で **【ログ】** の URL をクリックして、アプリケーションインスタンスのログを表示します。

ログイベントが、別のブラウザウィンドウに表示されます。

Spark エンジン監視のトラブルシューティング

Spark エンジン監視用にポートを設定する必要がありますか。

Spark エンジン監視には、データ統合サービスとソケットを介して通信するためのクラスタノードが必要です。データ統合サービスは、ドメインに設定されているポート範囲から、ソケットポートをランダムに選びます。ネットワーク管理者は、データ統合サービスが、クラスタノードからポート範囲にアクセスできることを確認する必要があります。管理者がポート範囲へのアクセスを提供できない場合は、SparkMonitoringPort カスタムプロパティで、データ統合サービスが固定ポートを使用するように設定できます。ネットワーク管理者は、データ統合サービスが、クラスタノードから設定したポートにアクセスできることを確認する必要があります。

リカバリされたジョブの監視統計の経過時間が 0 と表示される

ジョブがリカバリされたとき、[監視] タブに表示されるジョブの開始時刻と終了時刻は同じで、経過時間は 0 になります。この表示により、リカバリされたジョブを識別することができます。ジョブの経過時間をより正確に表示するには、クラスタで Spark ジョブログを表示するかデータ統合サービスでセッションログを表示してください。

ビッグデータデタリカバリを有効にしたが、[監視] タブで、リカバリされたジョブの統計情報が見つからないか、不正確に表示される

データ統合サービスがジョブをリカバリしたときに、ジョブは完了しているのに、Administrator tool の [監視] タブに未完了のジョブ統計が表示されることがあります。例えば、ジョブ統計に処理された行数が正しく表示されないことがあります。

第 9 章

階層データ処理

この章では、以下の項目について説明します。

- [階層データ処理の概要, 155 ページ](#)
- [マッピングの開発方法階層データの処理, 156 ページ](#)
- [復号データ型, 158 ページ](#)
- [複合ポート, 163 ページ](#)
- [複合データ型定義, 165 ページ](#)
- [型設定, 169 ページ](#)
- [複合演算子, 174 ページ](#)
- [複合関数, 176 ページ](#)
- [階層データプレビュー, 177 ページ](#)

階層データ処理の概要

Spark および Databricks Spark エンジンでは、配列、構造、マップなどの複合データ型を使用するマッピングを処理できます。複合データ型を使用すると、ランタイムエンジンは、複合ファイル内の階層データに対して直接読み取り、処理、および書き込みを行います。

Spark および Databricks Spark エンジンでは、Avro、JSON、および Parquet の複合ファイルの階層データを処理できます。これらのエンジンは、複合データ型を使用して、複合ファイル内の階層データのネイティブデータ型を表します。例えば、Avro ファイルのレコードタイプの階層データは、ランタイムエンジンでは構造データ型として表されます。

階層データ処理シナリオ

次の階層データ処理シナリオ向けにマッピングを開発できます。

- 階層データを生成および変更するため。
- リレーショナルデータを階層データに変換するため。
- 階層データをリレーショナルデータに変換するため。
- ある複合ファイル形式のデータを別の形式に変換するため。例えば、階層データを Avro ソースから読み取り、JSON ターゲットに書き込みます。

階層データ処理設定

複合ファイルにアクセスする接続を作成し、複合ファイル内のデータを表すデータオブジェクトを作成します。次に、複合データ型としてのプロジェクトカラムに対する、データオブジェクトの読み取り操作および書き込

み操作を設定します。複合ファイル内の階層データの読み取りおよび書き込みを行うには、マッピングを作成し、読み取り操作に基づいた読み取りトランスフォーメーションを追加して、書き込み操作に基づいた書き込みトランスフォーメーションを追加します。階層データを処理するには、マッピングの次のオブジェクトとトランスフォームプロパティを設定します。

- 複合ポート。マッピングの階層データを渡すために、複合ポートを作成します。複合ポートは、ポートに複合データ型を割り当てて作成します。
- 複合データ型定義。構造タイプの階層データを処理するには、構造データのスキーマを表す複合データ型定義を作成またはインポートします。
- 型設定。複合ポートのプロパティを定義するには、タイプ設定を指定または変更します。
- 複合演算子および複合関数。階層データを生成または変更するには、複合演算子や複合関数を使用して式を作成します。

階層変換ウィザードを使用して、マッピング開発タスクの一部を簡略化することもできます。

マッピングの開発方法階層データの処理

複合ポート、演算子、関数を持つマッピングを開発し、Spark または Databricks Spark エンジンで階層データを処理します。

注: マッピングを開発するためのタスクとタスクを実行する順序は、マッピングシナリオに応じて異なります。

次に、複合ファイル内の階層データの読み取り、書き込み、処理するためのマッピングを開発して実行する高度なタスクの概要を示します。

接続の作成

ファイルストレージに基づいて複合ファイル内のデータにアクセスする接続を作成します。

データオブジェクトを作成します。

1. ソースまたはターゲットとして複合ファイルを表すデータオブジェクトを作成します。
2. データオブジェクトのプロパティを設定します。
3. 読み取り操作および書き込み操作で、複合ファイル内のプロジェクトカラムに対してカラムファイルプロパティを複合データ型として有効にします。

マッピングを作成し、マッピングオブジェクトを追加します。

1. マッピングを作成し、読み取りトランスフォーメーション、書き込みトランスフォーメーションを追加します。
複合ファイルとの間で読み取りおよび書き込みを行うには、データオブジェクトの読み取りおよび書き込み操作に基づいて、読み取りトランスフォーメーションおよび書き込みトランスフォーメーションを追加します。

Avro ファイルまたは Parquet ファイルに書き込むにも、マッピング内の既存のトランスフォーメーションから複合ファイルターゲットを作成できます。

2. マッピングロジックに基づいて、ランタイムエンジンでサポートされるその他のトランスフォーメーションを追加します。

構造データを生成します。

マッピングシナリオに基づいて、階層変換ウィザードのいずれかを使用し、構造データを生成します。次の手順を手動で実行することもできます。

構造ポートの複合データ型定義を作成するかインポートします。

1. 構造データのスキーマを表す複合データ型定義を作成するかインポートします。
複合データ型定義は、モデルリポジトリオブジェクトである型定義ライブラリに格納されます。
型定義ライブラリのデフォルトの名前は、`m_Type_Definition_Library` です。
2. マッピングが1つまたは複数のマップレットを使用する場合、マッピングとマップレットの型定義ライブラリの名前を変更し、それらの名前が一意であるようにします。

トランスフォーメーション内の構造ポートを作成して設定します。

1. トランスフォーメーション内にポートを作成し、構造データ型を割り当てます。
2. 構造ポートの型設定を指定します。
構造ポートの複合データ型定義を参照する必要があります。
3. 複合関数を持つ式を作成し、構造データを生成します。

構造データを変更します。

構造データをリレーショナルデータまたは階層データに変換できます。構造データにプリミティブデータ型の要素が含まれる場合、要素をリレーショナルデータとして抽出できます。構造データに複合データ型の要素が含まれる場合、要素を階層データとして抽出できます。マッピングシナリオに基づいて、階層変換ウィザードのいずれかを使用し、構造データを変更します。次の手順を手動で実行することもできます。

1. 抽出する構造データの要素と一致するポートプロパティを持つ出力ポートを作成します。
2. 複合演算子または複合関数を持つ式を作成し、構造データを変更します。

配列データを生成します。

1. トランスフォーメーション内にポートを作成し、配列データ型を割り当てます。
2. 配列ポートの型設定を指定します。
3. 複合関数を持つ式を作成し、配列データを生成します。

配列データを変更します。

配列データをリレーショナルデータまたは階層データに変換できます。配列データにプリミティブデータ型の要素が含まれる場合、要素をリレーショナルデータとして抽出できます。配列データに複合データ型の要素が含まれる場合、要素を階層データとして抽出できます。マッピングシナリオに基づいて、階層変換ウィザードのいずれかを使用し、配列データを変更します。次の手順を手動で実行することもできます。

1. 抽出する配列データの要素と一致するポートプロパティを持つ出力ポートを作成します。
2. 複合演算子または複合関数を持つ式を作成し、配列データを変更します。

マップデータを生成します。

1. トランスフォーメーション内にポートを作成し、マップデータ型を割り当てます。
2. マップポートの型設定を指定します。
3. 複合関数を持つ式を作成しマップデータを生成します。

マップデータを変更します。

マップデータをリレーショナルデータまたは階層データに変換できます。マップデータにプリミティブデータ型の要素が含まれる場合、要素をリレーショナルデータとして抽出できます。マップデータに複合データ型の要素が含まれる場合、要素を階層データとして抽出できます。マッピングシナリオに基づいて、階層変換ウィザードのいずれかを使用し、マップデータを変更します。次の手順を手動で実行することもできます。

1. 抽出するマップデータの要素と一致するポートプロパティを持つ出力ポートを作成します。
2. 複合演算子または複合関数を持つ式を作成し、マップデータを変更します。

トランスフォーメーションを設定する。

マッピングロジックに基づいて、ポートをリンクしトランスフォーメーションプロパティを設定します。

マッピングプロパティを設定します。

以下のマッピングランタイムプロパティを設定します。

- Spark エンジンで実行するには、Spark 検証環境と、実行環境として Hadoop を選択します。
- Databricks Spark エンジンで実行するには、Databricks Spark 検証環境と、実行環境として Hadoop を選択します。

マッピングを検証して実行します。

1. マッピングを検証し、エラーがあれば解決します。
2. 必要に応じて、エンジン実行プランを表示し、ロジックをデバッグします。
3. マッピングを実行します。

復号データ型

複合データ型は、単一のカラム位置内に複数のデータ値を表すトランスフォーメーションデータ型です。データ値は要素と呼ばれます。複合データ型内の要素は、プリミティブデータ型または複合データ型にすることができます。複合データ型を使用すると、Spark または Databricks Spark エンジンで実行されるマッピング内で階層データを処理できます。

トランスフォーメーションデータ型には、次のデータ型があります。

プリミティブデータ型

単一のカラム位置内に単一のデータ値を表すトランスフォーメーションデータ型です。小数点型、整数型、文字列型などのデータ型は、プリミティブデータ型です。プリミティブデータ型は、任意のトランスフォーメーション内のポートに割り当てられます。

複合データ型

単一のカラム位置内に複数のデータ値を表すトランスフォーメーションデータ型です。配列型、マップ型、構造型などのデータ型は、複合データ型です。複合データ型は一部のトランスフォーメーション内のポートに割り当てられます。

ネストされたデータ型

複合データ型の要素を格納する複合データ型です。他の構造的配列を持つ構造的配列型または構造型などのデータ型は、ネストされたデータ型です。

次の表に、複合データ型の一覧を示します。

複合データ型	説明
array	配列は、要素の順序付けられたコレクションです。 要素は、プリミティブデータ型または複合データ型にすることができます。配列内のすべての要素は、同じデータ型である必要があります。
map	マップは、キーと値のペア要素の順序が指定されていないコレクションを表します。 キーはプリミティブデータ型である必要があります。この値は、プリミティブデータ型または複合データ型にすることができます。
struct	構造は、さまざまなデータ型の要素のコレクションです。 要素は、プリミティブデータ型または複合データ型にすることができます。構造は、データの構造を定義する 1 つのスキーマを持ちます。

次の図に、トランスフォーメーション内のポートに割り当てられたプリミティブデータ型、複合データ型、ネストされたデータ型を示します。

Properties		Data Viewer	
General			
Ports			
Port Selectors			
Dependencies			
Windowing			
Run-time Linking			
Advanced			
Mapping Outputs			
	Name	Type	Type Configuration
1	emp_name	string	N/A
2	emp_sal	decimal	N/A
3	emp_phone	array	string []
4	emp_id_dept	map	< integer, string >
5	emp_address	struct	(typedef_adrs)
6	emp_bonus	array	(typedef_perf) []

1. プリミティブデータ型
2. 複合データ型
3. ネストされたデータ型

emp_name ポートと emp_sal ポートは、プリミティブデータ型です。emp_phone、emp_id_dept、emp_address の各ポートは、複合データ型です。emp_bonus ポートはネストされたデータ型です。この配列に、構造型の要素が格納されています。

array データ型

array データ型は、要素の順序付けられたコレクションを表します。配列データを渡す、生成する、または処理するには、array データ型をポートに割り当てます。

配列は、0 から始まるインデックス付きリストです。配列インデックスは、配列要素の位置を示します。例えば、配列インデックス 0 は、配列の最初の要素を示します。トランスフォーメーション言語には、配列データを生成して処理する配列要素と関数にアクセスするための演算子があります。

配列は、1 次元または多次元にすることができます。1 次元配列は線形配列です。多次元配列は配列の配列です。array トランスフォーメーションデータ型は、最大 5 つの次元を持つことができます。

形式

array <data_type> []

このデータ型の引数を次の表に示します。

引数	説明
array	配列のカラムまたはポートの名前。
data_type	配列内の要素のデータ型。 要素は、プリミティブデータ型または複合データ型にすることができます。配列内のすべての要素は、同じデータ型である必要があります。
[]	添字として表される配列の次元。 単一の添字[]は、1次元配列を表します。2つの添字[][]は、2次元配列を表します。 各次元の要素は、同じデータ型です。

配列の要素には名前がありません。配列内の要素の数は、各行によって異なる場合があります。

配列の例

1 次元配列

次の配列カラムは、顧客の電話番号を含む文字列要素の1次元配列を表します。

custphone string[]

次の例は、custphone 列のデータ値を示しています。

custphone

[205-128-6478,722-515-2889]

[107-081-0961,718-051-8116]

[107-031-0961,NULL]

2 次元配列

次の配列カラムは、顧客の作業と個人の電子メールアドレスを含む文字列要素の2次元配列を表します。

email_work_pers string[][]

次の例は、email_work_pers 列のデータ値を示しています。

email_work_pers

[john_baer@xyz.com,jbaer@xyz.com][john.baer@fgh.com,jbaer@ijk.com]

[bobbi_apperley@xyz.com,bapperl@xyz.com][apperlbob@fgh.com,bobbi@ijk.com]

[linda_bender@xyz.com,lbender@xyz.com][l.bender@fgh.com,NULL]

Map データ型

map データ型は、キーと値のペア要素の順序が指定されていないコレクションを表します。マップ要素は、1つのものを別のものにマップするキーと値のペアです。マップデータを渡す、生成する、または処理するには、ポートに map データ型を割り当てます。

キーはプリミティブデータ型である必要があります。この値は、プリミティブデータ型または複合データ型にすることができます。複合データ型の値を持つ map データ型は、ネストされたマップになります。ネストされたマップには、最大で3つのレベルの map データ型ネストを含めることができます。

トランスフォーメーション言語には、マップ要素にアクセスするための添字演算子が含まれます。また、マップデータを生成して処理するための関数も含まれます。

形式

map <primitive_type -> data_type>

このデータ型の引数を次の表に示します。

引数	説明
map	マップカラムまたはポートの名前。
primitive_type	マップ要素内のキーのデータ型。 キーはプリミティブデータ型である必要があります。
data_type	マップ要素内の値のデータ型。 この値は、プリミティブデータ型または複合データ型にすることができます。

マップの例

次のマップカラムは、顧客 ID と顧客名をマッピングするための整数キーと文字列値を持つマップデータを表します。

custid_name <integer -> string>

次の例は、custid_name カラムのデータ値を示しています。

custid_name

<26745 -> 'John Baer'>

<56743 -> 'Bobbi Apperley'>

<32879 -> 'Linda Bender'>

Struct データ型

struct データ型は、異なるデータ型の要素のコレクションを表します。struct データ型には、データの構造を定義するスキーマが関連付けられています。構造データを渡す、生成、または処理するには、struct データ型をポートに割り当てます。

struct データ型のスキーマによって、構造データの要素名と要素数が決まります。スキーマは、構造データ内の要素の順序も決定します。Informatica は、構造データのスキーマを表す複合データ型定義を使用します。

トランスフォーメーション言語には、構造要素にアクセスするための演算子があります。また、構造データを生成および処理したり、データのスキーマを変更するための関数もあります。

形式

```
struct {element_name1:value1 [, element_name2:value2, ...]}
```

構造のスキーマは次の形式で表されます。

```
schema {element_name1:data_type1 [, element_name2:data_type2, ...]}
```

このデータ型の引数を次の表に示します。

引数	説明
struct	構造のカラムまたはポートの名前。
schema	データの構造の定義、 schema は、構造要素の名前とデータ型を決定する名前と型のペアです。
element_name	構造要素の名前。
value	構造要素の値。
data_type	要素値のデータ型。 要素の値は、プリミティブデータ型または複合データ型にすることができます。構造の各要素は、異なるデータ型を持つことができます。

構造の例

次のスキーマは、顧客の住所を格納する構造データです。

```
address  
{st_number:integer,st_name:string,city:string,state:string,zip:string}
```

次の例は、cust_address カラムの構造データ値を示しています。

cust_address

```
{st_number:154,st_name:Addison Ave,city:Redwood City,state:CA,zip:94065}
```

```
{st_number:204,st_name:Ellis St,city:Mountain View,state:CA,zip:94043}
```

```
{st_number:357,st_name:First St,city:Sunnyvale,state:CA,zip:94085}
```

複合データ型のルールとガイドライン

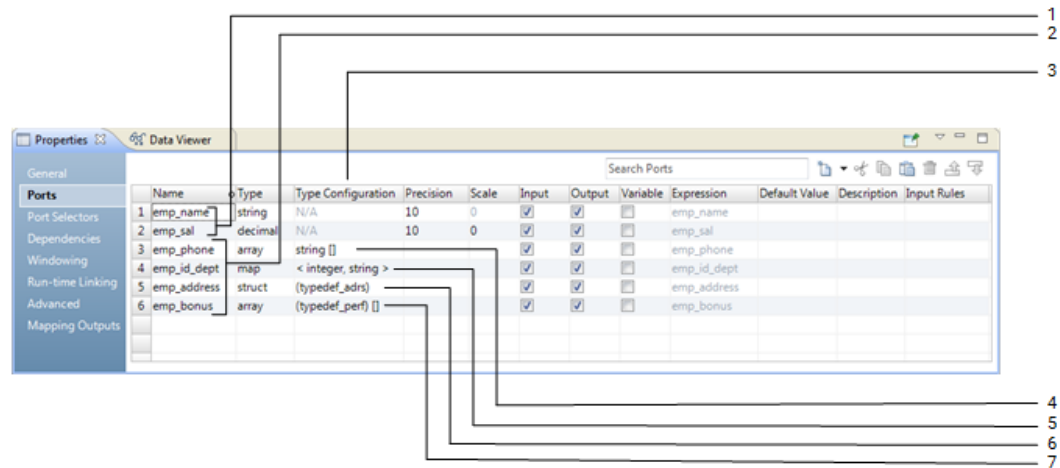
複合データ型を使用する際には、次のルールとガイドラインを考慮します。

- ネストされたデータ型には、最大 10 レベルのネストを含めることができます。
- ネストされたマップには、最大で 3 つのレベルの map データ型ネストを含めることができます。
- 配列データ型に、配列型の要素を直接含めることはできません。複数次元の配列を使用して、ネストされた配列を作成します。例えば、2 つの次元を持つ配列は複数配列の配列になります。
- 複数次元の配列には、最大で 5 つのレベルのネストを含めることができます。配列の次元は、ネストのレベルを決定します。
- 複数次元の各配列には、同じデータ型の要素を含める必要があります。

複合ポート

複合ポートは、複合データ型に割り当てられるポートです。複合データ型に基づき、複合ポートプロパティを指定する必要があります。複合ポートは、マッピング内の階層データを渡すまたは処理するためにトランスフォーメーション内で使用します。

次の図に、トランスフォーメーション用の「ポート」タブの複合ポートと複合ポートプロパティを示します。



1. ポート
2. 複合ポート
3. 型設定
4. 配列ポートの型設定
5. マップポートの型設定
6. 構造ポートの型設定
7. ネストされたデータ型のポートの型設定

トランスフォーメーションには、データ型に基づいて次のポートとポートプロパティを含められます。

ポート

任意のトランスフォーメーション内で作成できるプリミティブデータ型のポートです。

複合ポート

一部のトランスフォーメーション内で作成できる複合データ型またはネストされたデータ型のポートです。配列ポート、マップポート、構造ポートは複合ポートです。「タイプ設定」カラム内に、複合データ型に基づいて複合ポートプロパティを指定する必要があります。

タイプ設定

複合ポートに指定するプロパティのセットです。タイプ設定によって、複合データ型要素のデータ型またはデータのスキーマが決まります。配列ポートおよびマップポートの要素のデータ型を指定します。構造ポートの複合データ型定義を指定します。

配列ポートの型設定

配列要素のデータ型を決定するプロパティです。この図では、emp_phone 配列ポートは、文字列要素の順序付けられたコレクションを持つ 1 次元配列です。文字列要素を持つ 1 つの配列は、文字列の配列とも呼ばれます。

マップポートの型設定

マップ要素のキーと値のペアのデータ型を決定するプロパティです。この図では、emp_id_dept マップポートは、整数型と文字列型のキーと値のペアの順序が指定されていないコレクションです。

構造ポートの型設定

データのスキーマを決定するプロパティです。スキーマを表すには、複合データ型定義を作成またはインポートします。この図では、emp_address 構造ポートは、typedef_adrs 複合データ型定義を参照します。

ネストされたデータ型のポートの型設定

ネストされたデータ型を決定するプロパティです。この図では、emp_bonus 配列ポートは、構造要素の順序付けられたコレクションを持つ 1 次元配列です。この構造要素は、typedef_bonus 複合データ型定義を参照します。構造要素を持つ 1 つの配列は、構造の配列とも呼ばれます。

トランスフォーメーション内の複合ポート

複合ポートは、Spark および Databricks Spark エンジンでサポートされている一部のトランスフォーメーション内で作成できます。読み取りトランスフォーメーションと書き込みトランスフォーメーションによって、階層データを複合データ型として渡すポートを表せます。

次の各トランスフォーメーションで複合ポートを作成できます。

- アグリゲータ
- 式
- フィルタ
- Java
- ジョイナ
- ルックアップ
- ノーマライザ
- ルーター
- ソータ
- 共有体

注: Databricks Spark エンジンでは Java トランスフォーメーションはサポートされません。

読み取りトランスフォーメーションと書き込みトランスフォーメーションでは、複合ファイル内の階層データを読み書きできます。階層データを読み書きするには、読み取りトランスフォーメーションと書き込みトランスフォーメーションは次の要件を満たしている必要があります。

- トランスフォーメーションは、複合ファイルデータオブジェクトに基づいている必要があります。
- データオブジェクトの読み取り操作および書き込み操作は、複合データ型としてのプロジェクトカラムである必要があります。

複合ポートのルールとガイドライン

複合ポートを使用する場合は、次のルールとガイドラインを考慮します。

- アグリゲータトランスフォーメーション。グループ化の値は複合ポートとして定義できません。
- フィルタトランスフォーメーション。>、<、>=、および<=の演算子は、フィルタ条件で複合ポートのデータを比較するために使用できません。

- ジョイナトランスフォーメーション。>、<、>=、および<=の演算子は、結合条件で複合ポートのデータを比較するために使用できません。
- ルックアップトランスフォーメーション。複合ポートはルックアップ条件で使用できません。
- ランクトランスフォーメーション。グループ化の値またはランクの値は複合ポートとして定義できません。
- ルータトランスフォーメーション。>、<、>=、および<=の演算子は、グループフィルタ条件で複合ポートのデータを比較するために使用できません。
- ソータトランスフォーメーション。ソートキー値は複合ポートとして定義できません。
- 複合演算子を使用して、プリミティブデータ型の複合ポートの要素を指定できます。
例えば、配列ポート"emp_names"には文字列要素が含まれます。グループ化の値を、文字列型である emp_names[0]として定義できます。

複合ポートの作成

トランスフォーメーション内に、Spark または Databricks Spark エンジンで実行されるマッピング内で階層データを渡す、または処理する複合ポートを作成します。

1. マッピングでトランスフォーメーションを選択します。
 2. ポートを作成します。
 3. ポートの【**タイプ**】カラムで、複合データ型を選択します。
[タイプ] カラム内に、ポートの複合データ型が表示されます。
- 複合ポートの作成後、複合ポートの型設定を指定します。

複合データ型定義

複合データ型定義は、データのスキーマを表します。複合データ型定義は、構造ポート用のデータのスキーマを定義するために使用します。複合ポートが構造型であるか、構造型の要素が含まれる場合、複合データ型定義を参照する型設定を指定する必要があります。

複合データ型定義は作成またはインポートできます。複合データ型定義は、複合ファイルからインポートします。複合データ型定義は、モデルリポジトリオブジェクトである型定義ライブラリに格納されます。

マッピングまたはマップレットを作成すると、Developer tool が Type_Definition_Library という名前で型定義ライブラリを作成します。型定義ライブラリと複合データ型定義は、マッピングのアウトラインビューから表示して名前を変更できます。複合データ型定義は、マッピングエディタの [型定義ライブラリ] タブに表示されます。このタブの名前は、型定義ライブラリの名前に基づきます。複合データ型定義は、[型定義ライブラリ] タブで作成またはインポートします。マッピングが1つまたは複数のマップレットを使用する場合、マッピングとマップレットの型定義ライブラリの名前を変更し、それらの名前が一意であるようにします。

型定義ライブラリと複合データ型定義は、1つのマッピングまたはマップレットに固有です。マッピングまたはマップレットには、1つ以上の複合データ型定義を持たせることができます。マッピングまたはマップレット内の1つまたは複数の複合ポートで、同じ複合データ型定義を参照できます。

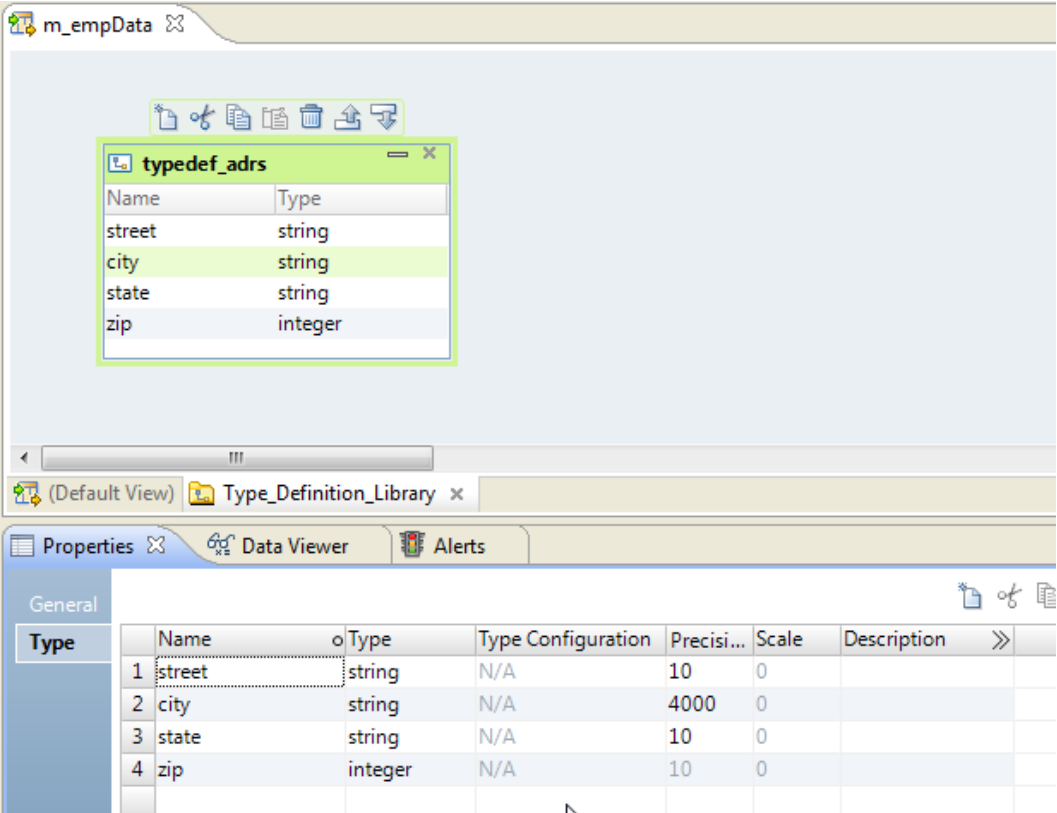
複合データ型定義の例

work_address と home_address の各複合ポートは構造型であり、次のスキーマがあります。

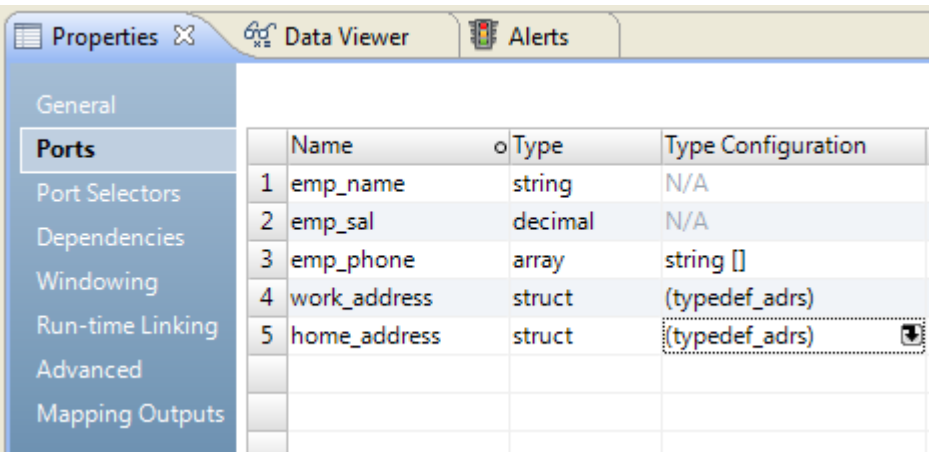
```
{street:string, city:string, state:string, zip:integer}
```

このマッピング内で、データのスキーマを定義する typedef_adrs 複合データ型定義を作成します。次に、構造ポートである work_address と home_address の型設定を、typedef_adrs 複合データ型定義を使用するように指定します。

次の図に、typedef_adrs 複合データ型定義を示します。



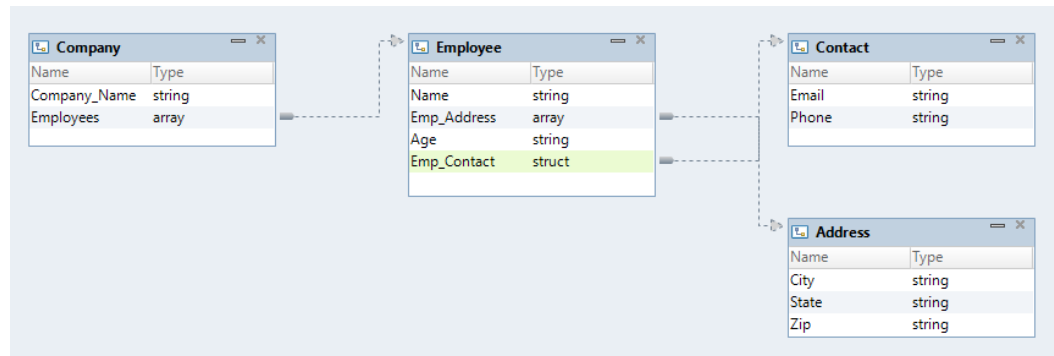
次の図に、typedef_adrs 複合データ型定義を参照する work_address と home_address の 2 つの構造ポートを示します。



ネストされたデータ型定義

複合データ型定義の要素は、型定義ライブラリ内の 1 つ以上の複合データ型定義を参照できます。このような複合データ型定義は、ネストされたデータ型定義と呼ばれます。

次の図は、型定義ライブラリのタブでの、ネストされたデータ型定義 Company を示しています。



ネストされたデータ型定義 Company は、次の複合データ型定義を参照しています。

- 複合データ型定義 Company で、配列要素 Employees は複合データ型定義 Employee を参照します。
- 複合データ型定義 Employee で、要素 Emp_Address は複合データ型定義 Address を参照します。
- 複合データ型定義 Employee で、要素 Emp_Contact は複合データ型定義 Contact を参照します。

注: 再帰的なデータ型定義では、任意のレベルの複合データ型定義のいずれかは、その親のいずれかと同じになります。再帰的なデータ型定義が、複合ポートの構造ポートまたは構造要素を参照することはできません。

複合データ型定義のルールとガイドライン

複合データ型定義を使用する際には、次のルールとガイドラインを考慮します。

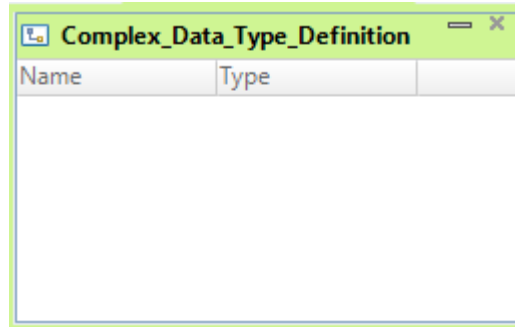
- 複合ポートの構造ポートと構造要素は、複合データ型定義を参照する必要があります。
- あるマッピング内の複合データ型定義から、別のマッピングの複合ポートを参照することはできません。
- 再帰的なデータ型定義が、複合ポートの構造ポートまたは構造要素を参照することはできません。
- 複合データ型定義の要素に加えた変更は、複合データ型定義を参照する複合ポートに自動的にプロパゲートされます。
- 型定義ライブラリまたは複合データ型定義の名前を変更した場合、STRUCT_AS、RESPEC、および CAST などの複合関数を使用する式の名前を更新する必要があります。

複合データ型定義の作成

複合データ型定義は、構造型のデータのスキーマを表します。マッピングまたはマップレットの複合データ型定義は、マッピングエディタの [型定義ライブラリ] タブで作成できます。

1. Developer tool で、複合データ型定義のインポート先のマッピングまたはマップレットを開きます。
2. [アウトライン] ビューで、[型定義ライブラリ] を選択し、[Type_Definition_Library] タブをマッピングエディタで開きます。
3. エディタ内の何もない領域を右クリックし、[新しい複合データ型定義] をクリックします。

空の複合データ型定義が、マッピングエディタの [型定義ライブラリ] タブに表示されます。



4. 必要に応じて、複合データ型定義の名前を変更します。
5. **【新規】** ボタンをクリックし、名前と型を持つデータ型定義に要素を追加します。
要素のデータ型は、プリミティブデータ型または複合データ型にすることができます。

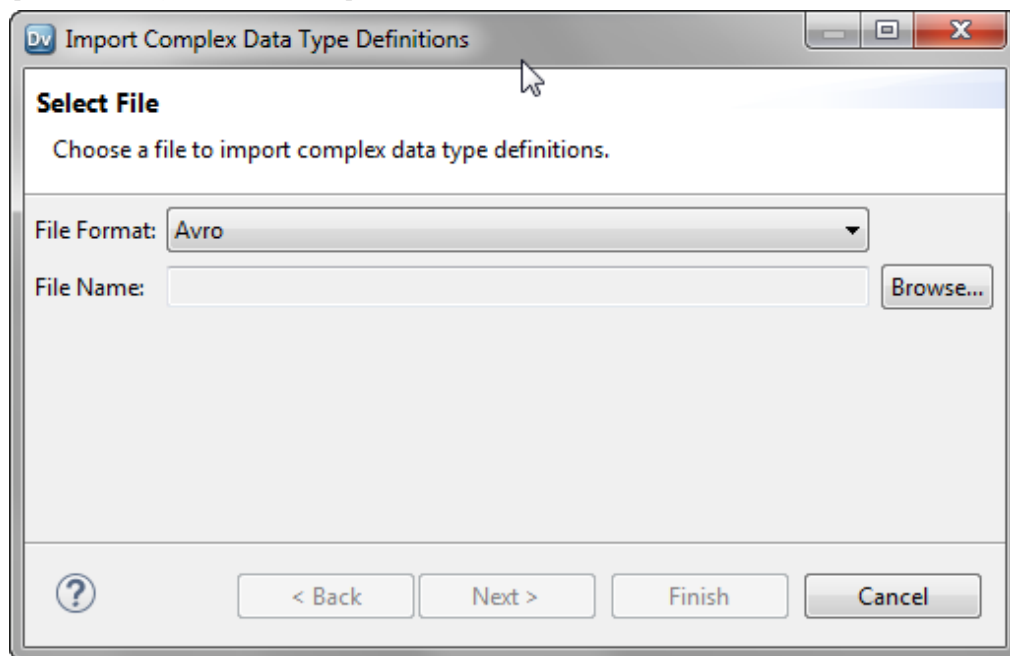
複合データ型定義のインポート

複合データ型定義は、構造型のデータのスキーマを表します。複合ファイル内の階層データのスキーマを、複合データ型定義として型定義ライブラリにインポートできます。

複合データ型定義は、Avro、JSON、または Parquet スキーマファイルからインポートします。例えば、Avro スキーマを .avsc ファイルから複合データ型定義としてインポートできます。

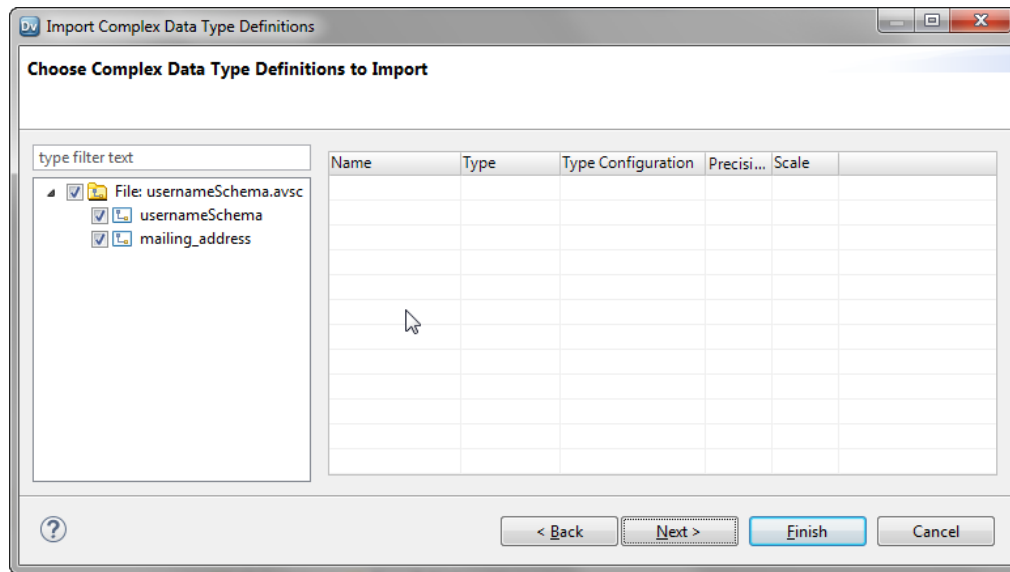
1. Developer tool で、複合データ型定義のインポート先のマッピングまたはマップレットを開きます。
2. **【アウトライン】** ビューで、**【型定義ライブラリ】** を選択し、**【Type_Definition_Library】** タブをマッピングエディタで開きます。
3. エディタ内の何もない領域を右クリックし、**【複合データ型定義のインポート】** をクリックします。

【複合データ型定義のインポート】 ダイアログボックスが表示されます。



4. 複合ファイル形式を **【ファイル形式】** リストから選択します。
5. **【参照】** をクリックし、複合データ型定義のインポート元とする複合ファイルスキーマを選択します。

6. 複合ファイルスキーマに移動し、**【開く】** をクリックします。
 7. **【次へ】** をクリックします。
- 【インポートする複合データ型定義の選択】** ページが表示されます。



8. インポート用のリストから、1 つまたは複数のスキーマを選択します。
 9. **【次へ】** をクリックします。
- 【インポートする複合データ型定義】** ページが表示されます。
10. インポートする複合データ型定義を見直し、**【完了】** をクリックします。
- 複合データ型定義は、マッピングまたはマップレットの **【型定義ライブラリ】** タブに表示されます。

型設定

型設定は、複合データ型要素のデータ型、またはデータのスキーマを指定する複合ポートプロパティのセットです。複合ポートの作成後、**【ポート】** タブで複合ポートの型設定を指定するか変更します。

複合ポートの型設定は、ポートに割り当てられた複合データ型に依存します。構造ポートの複合データ型定義、または複合ポート内の構造要素を指定する必要があります。配列ポートおよびマップポートには、デフォルトの型設定があり、これは変更できます。

次の表に、複合ポートの型設定プロパティの一覧を示します。

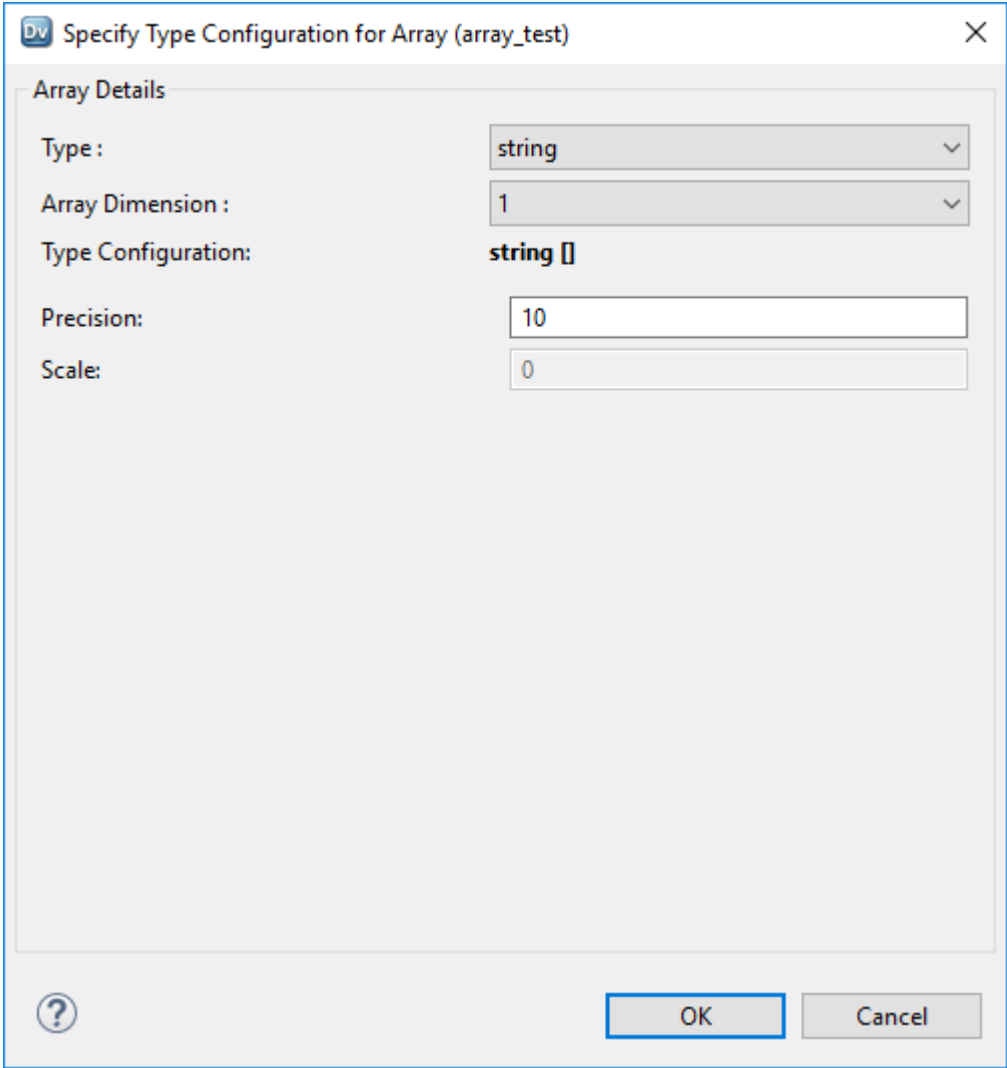
複合ポート	型設定プロパティ
配列	- タイプ。配列内の要素のデータ型。デフォルトは <code>string</code> です。 - 配列の次元配列のレベル数。デフォルトは <code>1</code> です。 - 精度。数値データ型の最大有効桁数または文字列データ型の最大文字数です。 - スケール。数値の小数点以下の最大桁数です。
マップ	- タイプ。キーと値のペアのデータ型。デフォルトは、キーと値に対して <code>string</code> です。 - 精度。数値データ型の最大有効桁数または文字列データ型の最大文字数です。 - スケール。数値の小数点以下の最大桁数です。
構造	- 複合データ型定義。型定義ライブラリ内で複合データ型定義として作成またはインポートしたデータのスキーマです。

配列ポートのタイプ設定の変更

配列のデフォルトのタイプ設定は、文字列の 1 次元の配列である `string[]` です。配列ポートのタイプ設定は変更できます。配列要素と配列の次元のデータ型を指定します。

1. トランスフォーメーションで、タイプ設定を指定する配列ポートを選択します。
2. **【タイプ設定】** カラムで、**【開く】** ボタンをクリックします。

【配列のタイプ設定の指定】ダイアログボックスが表示されます。



The dialog box is titled "Specify Type Configuration for Array (array_test)". It contains the following fields:

- Type :** A dropdown menu with "string" selected.
- Array Dimension :** A dropdown menu with "1" selected.
- Type Configuration:** A text field displaying "string []".
- Precision:** A text field with "10".
- Scale:** A text field with "0".

At the bottom, there is a help icon (question mark in a circle), an "OK" button, and a "Cancel" button.

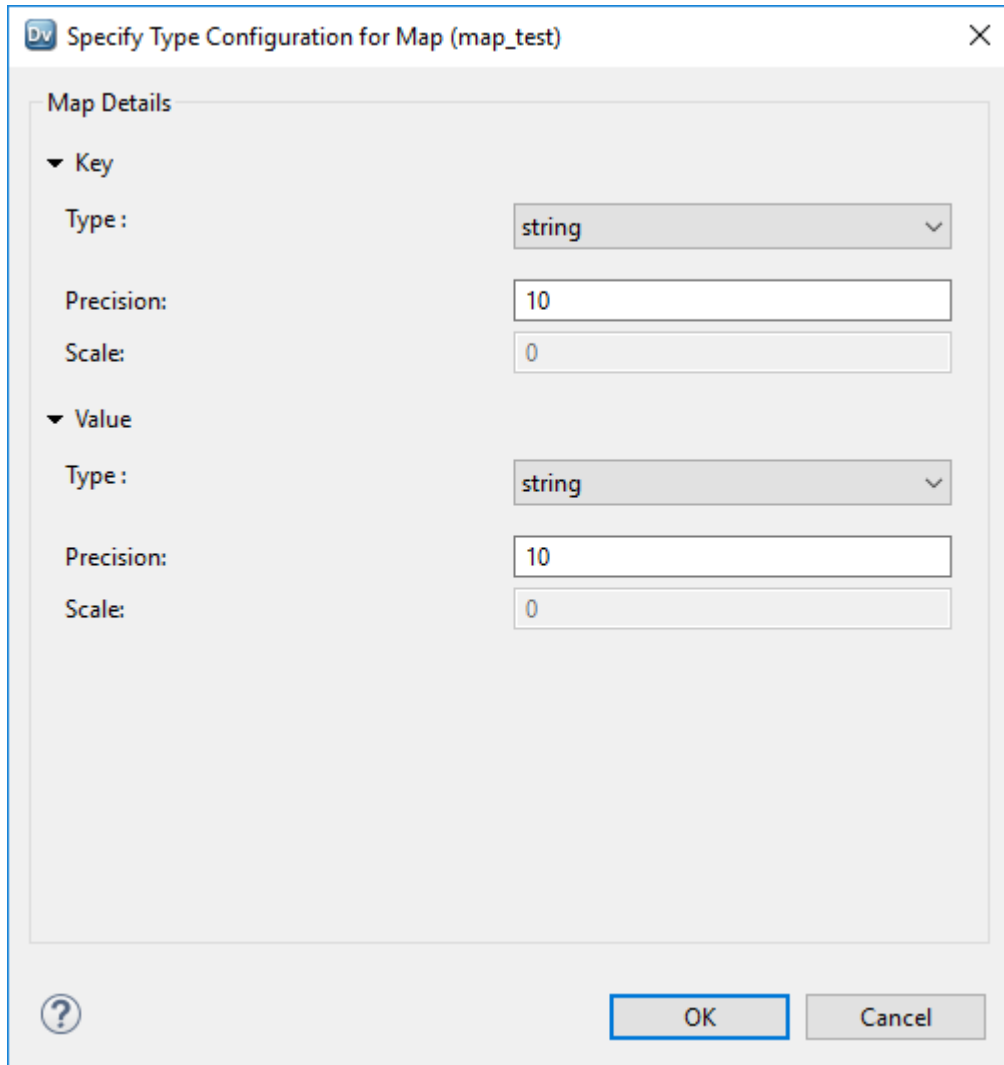
3. **【タイプ】** ドロップダウンメニューで、配列内の要素のデータ型を変更します。
例えば、配列内の要素が整数型である場合、**【整数】** を選択します。
タイプ設定値は **integer []** として表示されます。
4. 必要に応じて、**【配列の次元】** ドロップダウンメニューで、ネストされた配列を作成するための数値を選択します。
例えば、配列の次元を [3] とすると、3 レベルのネストされた配列が作成されます。タイプ設定値は **integer [] [] []** として表示されます。
5. 必要に応じて、次のプロパティを設定します。
 - **精度**。数値データ型の最大有効桁数または文字列データ型の最大文字数です。
 - **位取り**。数値の小数点以下の最大桁数です。
6. **【OK】** をクリックします。
【ポート】 タブで、配列ポートの **【タイプ設定】** カラムに、指定したタイプ設定が表示されます。

マップポートのタイプ設定の変更

マップのデフォルトのタイプ設定は<string, string>です。マップポートのタイプ設定は変更できます。キーと値のペアのデータ型を指定します。

1. トランスフォーメーションで、タイプ設定を設定するマップポートを選択します。
2. **【タイプ設定】** カラムで、**【開く】** ボタンをクリックします。

【タイプ設定】 ダイアログボックスが表示されます。

A screenshot of a dialog box titled "Specify Type Configuration for Map (map_test)". The dialog has a close button (X) in the top right corner. It contains two sections: "Key" and "Value", each with a dropdown menu for "Type", a text box for "Precision", and a text box for "Scale". Both "Key" and "Value" sections have "string" selected in the "Type" dropdown, "10" in the "Precision" box, and "0" in the "Scale" box. At the bottom left is a help icon (question mark), and at the bottom right are "OK" and "Cancel" buttons.

Specify Type Configuration for Map (map_test)

Map Details

▼ Key

Type: string

Precision: 10

Scale: 0

▼ Value

Type: string

Precision: 10

Scale: 0

?

OK Cancel

3. マップデータ型のキーと値のプロパティを指定します。
 - a. **【タイプ】** ドロップダウンメニューで、データ型を選択します。
 - b. 必要に応じて、**【精度】** ボックスに、数値データ型の最大有効桁数または文字列データ型の最大文字数を指定します。
 - c. 必要に応じて、**【スケール】** ボックスに、数値の小数点以下の最大桁数を指定します。
4. **【OK】** をクリックします。

【ポート】 タブで、マップポートの **【タイプ設定】** カラムに、指定したタイプ設定が表示されます。

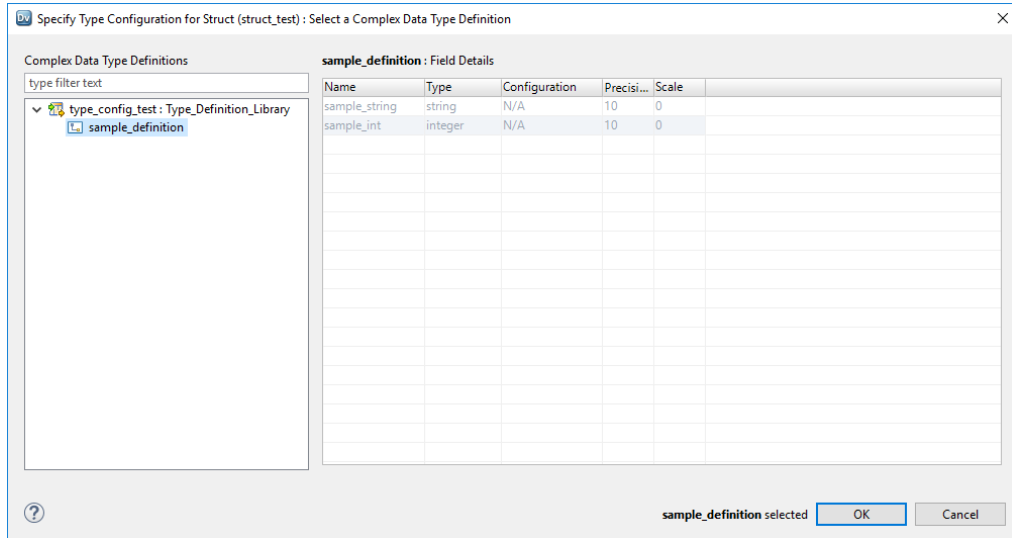
構造ポートの型設定の指定

構造ポートの型設定には、型定義ライブラリ内に複合データ型定義が必要です。構造データのスキーマを表す複合データ型定義を指定します。

構造ポートの型設定を指定する前に、複合データ型定義を作成するかインポートします。型定義ライブラリ内の既存の複合データ型定義を参照することもできます。

1. トランスフォーメーションで、タイプ設定を指定する構造ポートを選択します。
2. **【タイプ設定】** カラムで、**【開く】** ボタンをクリックします。

[タイプ設定] ダイアログボックスが表示されます。



- マッピングまたはマップレットの型定義ライブラリ内の定義のリストから、複合データ型定義を選択します。
- [OK]** をクリックします。
[ポート] タブで、構造ポートの **[タイプ設定]** カラムに、指定したタイプ設定が表示されます。

複合演算子

複合演算子は、複合データ型の要素にアクセスするための演算子の一種です。トランスフォーメーション言語には、複合データ型用の複合演算子が含まれます。複合演算子は、階層データ内の要素へのアクセスまたは抽出のために使用します。

次の表に、複合演算子の一覧を示します。

複合演算子	説明
[]	添字演算子。配列要素およびマップ要素にアクセスするために使用します。 構文: <i>array[index]</i> - <i>array</i> は、アクセスする要素が含まれる配列です。 - <i>index</i> は、配列内の要素の位置です。 例えば、1 次元の配列内の最初の要素にアクセスするには、[0]を使用します。 構文: <i>map[key]</i> - <i>map</i> は、アクセスするキーに対応する値が含まれるマップです。 - <i>key</i> は、マップ値を取得する対象のマップキーです。
.	ドット演算子。構造要素にアクセスするために使用します。 構文: <i>struct.element_name</i> - <i>struct</i> は、アクセスする要素が含まれる構造です。 - <i>element_name</i> は、構造要素の名前です。

複合演算子は、階層データをリレーショナルデータに変換するために式の中で使用します。

演算子の構文、戻り値、例の詳細については、『Informatica Developer トランスフォーメーション言語リファレンス』を参照してください。

複合演算子の例

- JSON ファイル内の配列データをリレーショナルデータに変換します。
JSON ファイルには、整数要素を持つ配列型である `quarterly_sales` カラムがあります。階層データをリレーショナルデータに変換するためのマッピングを開発します。`quarterly_sales` 配列ポートには、4 つの要素が格納されています。添字演算子を式に使用し、この 4 つの要素を `q1_sales`、`q2_sales`、`q3_sales`、`q4_sales` の 4 つの整数ポートに抽出します。

整数出力ポートの式は次のようになります。

```
quarterly_sales[0]  
quarterly_sales[1]  
quarterly_sales[2]  
quarterly_sales[3]
```

- Parquet ファイル内の構造データをリレーショナルデータに変換します。
Parquet ファイルには、顧客住所を構造型として格納する住所カラムがあります。階層データをリレーショナルデータに変換するためのマッピングを開発します。住所構造ポートには、文字列型の市区町村、都道府県、郵便番号の各要素が格納されています。ドット演算子を式に使用し、この 3 つの構造要素を市区町村、都道府県、郵便番号の 3 つの文字列ポートに抽出します。

文字列出力ポートの式は次のようになります。

```
address.city  
address.state  
address.zip
```

添字演算子を使用した配列要素の抽出

添字演算子を式の中で使用し、配列内の 1 つ以上の要素を抽出します。

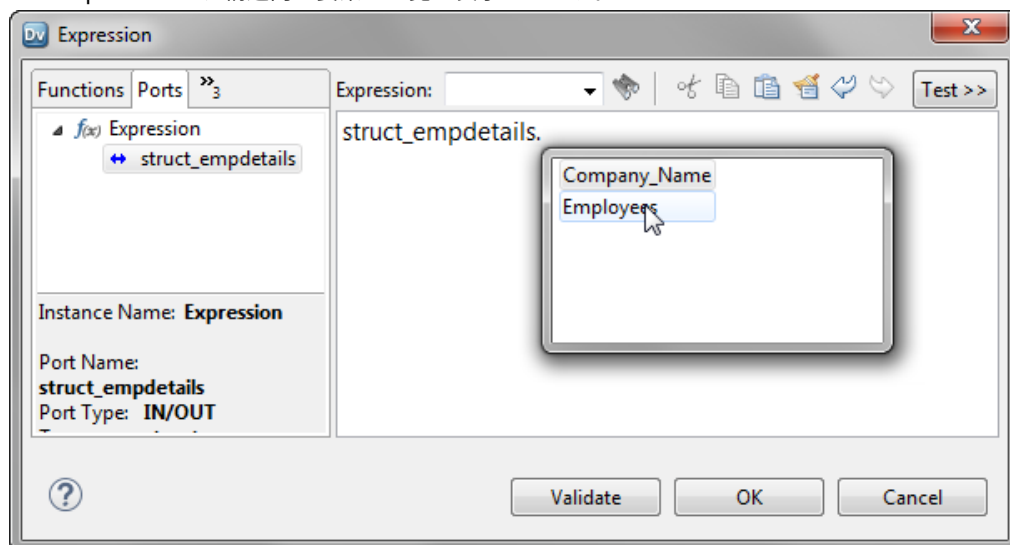
1. トランスフォーメーション内で、抽出する配列要素の戻り値と同じデータ型の出力ポートを作成します。
配列内の複数の要素を抽出するには、配列出力ポートを作成する必要があります。配列ポートの型設定内の要素のデータ型は、配列要素の戻り値のデータ型と一致する必要があります。
2. 出力ポートの【式】カラムで、**【開く】** ボタンをクリックします。
式エディタが表示されます。
3. エディタ内で既存の式を削除します。
4. **【ポート】** タブをクリックし、配列ポートを選択します。
5. 添字演算子内に 1 つまたは複数のインデックス値を入力します。
複数次元配列内の配列要素を抽出するには、各添字演算子内にインデックス値を入力します。
6. **【検証】** をクリックして式を検証します。
7. **【OK】** をクリックして、**【式の検証】** ダイアログボックスを終了します。
8. **【OK】** をクリックし、**式エディタ**を終了します。

ドット演算子を使用した構造要素の抽出

ドット演算子を式の中で使用し、構造の要素を抽出します。

1. トランスフォーメーション内で、抽出する構造要素の戻り値と同じデータ型の出力ポートを作成します。
2. 出力ポートの【式】カラムで、**【開く】** ボタンをクリックします。
式エディタが表示されます。
3. エディタ内で既存の式を削除します。
4. **【ポート】** タブをクリックし、構造ポートを選択します。
5. ポート名の後にドットを入力します。

Developer tool に、構造内の要素の一覧が表示されます。



6. 抽出する要素を選択します。
7. **【検証】** をクリックして式を検証します。

8. **[OK]** をクリックして、[式の検証] ダイアログボックスを終了します。
9. **[OK]** をクリックし、式エディタを終了します。

複合関数

複合関数は、入力値または戻り値の型が複合データ型である定義済み関数の一種です。トランスフォーメーション言語には、複合データ型用の複合関数が含まれます。複合関数は、階層データを生成および処理するために使用します。

次の表に、配列データ型の複合関数を示します。

複合関数	説明
<code>ARRAY (element1 [,element2] ...)</code>	指定した要素を持つ配列を生成します。引数のデータ型によって、配列のデータ型が決まります。
<code>COLLECT_LIST (value)</code>	指定したポート内の要素を持つ配列を返します。引数のデータ型によって、配列のデータ型が決まります。COLLECT_LIST は集計関数です。
<code>CONCAT_ARRAY (' ', array_of_strings)</code>	指定したセパレータに基づいて配列内の文字列要素を連結し、文字列を返します。
<code>SIZE (array)</code>	配列のサイズを返します。

次の表に、マップデータ型の複合関数を示します。

複合関数	説明
<code>MAP (key1, value1 [,key2, value2] ...)</code>	指定したキーと値のペアに基づく要素を持つマップを生成します。引数のデータ型によって、マップ要素のデータ型が決まります。
<code>MAP_FROM_ARRAYS (key_array, value_array)</code>	指定したキーと値の配列からマップを生成します。引数のデータ型によって、マップ要素のデータ型が決まります。
<code>MAP_KEYS (map)</code>	指定したマップのキーの配列を返します。
<code>MAP_VALUES (map)</code>	指定したマップの値の配列を返します。
<code>COLLECT_MAP (value)</code>	指定した引数に基づく要素を持つマップを返します。引数のデータ型によって、マップのデータ型が決まります。COLLECT_MAP は集計関数です。
<code>SIZE (map)</code>	マップのサイズを返します。

次の表に、構造データ型の複合関数を示します。

複合関数	説明
STRUCT_AS (<i>type_definition</i> , <i>struct</i>)	指定した複合データ型定義と、引数として渡す値に基づいて、スキーマを持つ構造を生成します。
STRUCT (<i>element_name1</i> , <i>value1</i>)	要素に対して指定した名前と値を持つ構造を生成します。値のデータ型は、指定した値の引数に基づきます。
RESPEC (<i>type_definition</i> , <i>struct</i>)	指定した複合データ型定義の要素の名前に基づいて、指定した構造値の各要素の名前を変更します。
CAST (<i>type_definition</i> , <i>struct</i>)	指定した構造値に対する各要素のデータ型と名前を、指定した複合データ型定義の対応するデータ型と名前に基づいて変更します。

関数の構文、戻り値、例の詳細については、『Informatica Developer トランスフォーメーション言語リファレンス』を参照してください。

階層データプレビュー

Developer tool で、マッピング内の階層データをプレビューできます。マッピング内のプレビューポイントとして、ソースおよびトランスフォーメーションを選択できます。階層データのプレビューは、ビッグデータのマッピングを設計およびデバッグする際に役立ちます。

階層タイプのデータのプレビューを実行すると、データ統合サービスは Spark エンジンに要求を渡します。結果を表示するときに、複合データ型、スキーマ、および値を表示できます。ブレッドクラムを使用して構造をナビゲートすることもできます。結果を JSON ファイルにエクスポートできます。

注: 階層データのデータプレビューは、テクニカルプレビュー用に用意されています。テクニカルプレビュー機能はサポートされていますが、保証対象外で本番環境には対応していません。非本番環境でのみ使用することをお勧めします。

注: データプレビューは、Spark エンジンのみでサポートされています。Databricks Spark エンジン向けに設定されたマッピングで、階層データをプレビューすることはできません。

階層データプレビュープロセス

データをプレビューすると、データ統合サービスにより、ジョブをネイティブ環境と Spark エンジンのどちらで実行するかが決定されます。

プレビューポイントまたはマッピング内の任意のアップストリームトランスフォーメーションに階層データ型が含まれる場合は、データ統合サービスはジョブを Spark エンジンにプッシュします。それ以外の場合は、データ統合サービスがデータプレビューを実行します。プレビューポイントは、データを表示するように選択したマッピング内のオブジェクトです。

データ統合サービスは、データプレビュージョブを Spark エンジンにプッシュするときに次のプロセスを使用します。

1. データ統合サービスが、プレビューポイントに基づいてターゲットを含むマッピングを生成します。
2. マッピングとプレビュー要求を Spark エンジンに渡します。

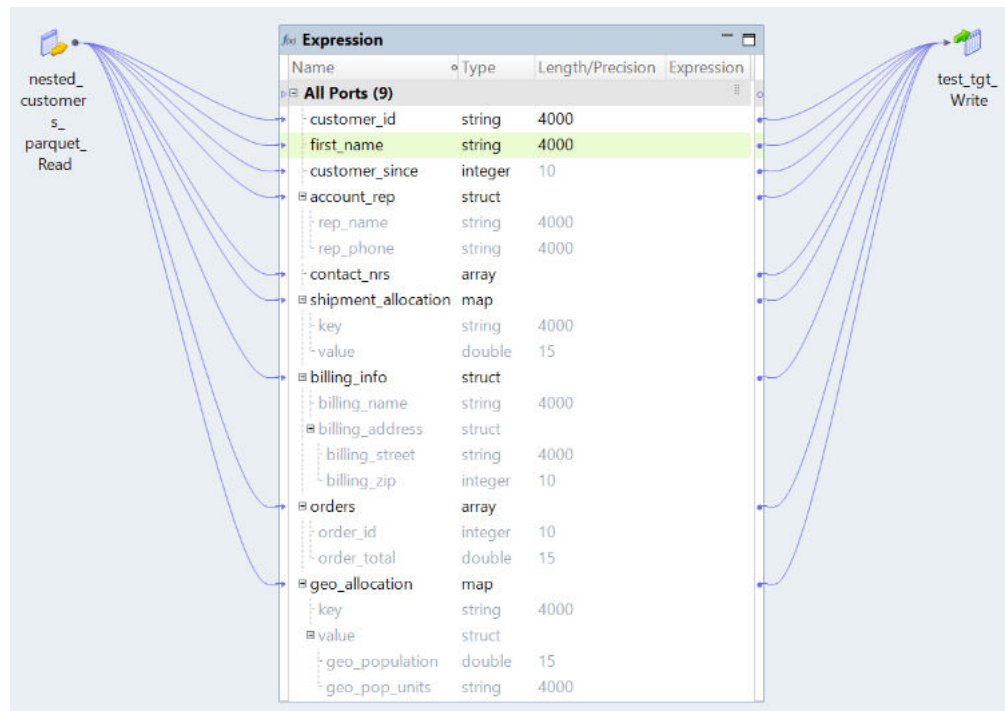
3. Spark エンジンがマッピングを実行し、設定されたステージングディレクトリに基づいてデータをステージングします。
4. データ統合サービスがステージングされたデータを Developer tool に渡してから、ステージングされたデータを削除します。
5. プレビューの結果が Developer tool のデータビューアに表示されます。

注: データプレビューを実行すると、**【ランタイム】** ビューで選択した検証環境がデータ統合サービスによって検証されます。

データプレビューインタフェーステーブル

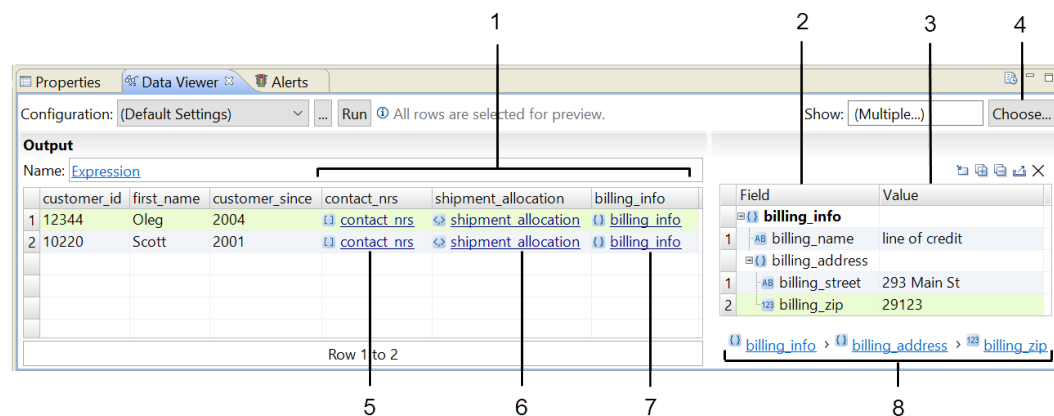
データプレビューインタフェースは、データビューアビューと階層タイプのパネルで構成されます。プレビューポイントに基づいて、階層データをプレビューできます。

次の図に、複合データタイプを含むマッピングを示します。



式が評価された後の階層データを表示するには、プレビューポイントとして式トランスフォーメーションを使用します。

次の図に、このデータプレビューの結果を示します。



1. 階層キー
2. スキーマ
3. 値
4. ポートフィルタ
5. 配列データ型
6. Map データ型
7. Struct データ型
8. ブレッドクラム

データビューア

階層データをプレビューするときには、データビューアでカラムとして表されているプレビューポイントのポートを表示します。階層タイプを識別するために、次の句読アイコンが使用されています。

- [] は配列データ型を示すのに使用されています。
- { } は構造データ型を示すのに使用されています。
- <, > はマップデータ型を示すのに使用されています。

ツールチップを表示して、階層データタイプの詳細を表示することもできます。表示するポート数を制限する必要がある場合は、出力に表示するポートを選択します。

階層タイプパネルで、詳細を表示するフィールドを強調表示します。任意の階層フィールドを右クリックすると、そのフィールドのデータを JSON 形式でエクスポートできます。

データのエクスポート

【データビューア】 ビューに表示されているデータを JSON ファイルにエクスポートできます。データのローカルコピーを作成するには、データをエクスポートします。

1. **【データビューア】** ビューで、結果を右クリックして **【データのエクスポート】** 選択します。
2. ファイル名を入力します。
3. ファイルの保存先を選択します。
4. **【OK】** をクリックします。

階層タイプパネル

階層フィールドをクリックすると、階層フィールドの詳細が階層タイプパネルに表示されます。

階層タイプパネルでは、選択した階層データのスキーマと値を表示できます。ブレッドキラムを使用して構造をナビゲートできます。構造が非常に深い場合は、フィールドを展開および折りたたむことができます。構造を展開すると、階層内のデータを表示できます。例えば、階層タイプパネルで請求情報を表示すると、請求名、番地、ZIP コードの値を表示できます。

階層データのプレビュー

[データビューア] ビューでデータプレビューを実行するときに、階層フィールドの構造とデータの表示、表示するポートのフィルタリングなどのタスクを実行できます。結果を JSON ファイルにエクスポートすることもできます。

データプレビューを実行する前に、マッピングに複合ファイルオブジェクトを追加して、Spark がマッピングランタイム環境として設定されていることを検証します。

1. 階層データプレビューを実行するには、Developer tool でマッピングを開いて、プレビューポイントを選択します。
2. データプレビューを実行するには、[データビューア] ビューで [実行] をクリックします。
データプレビューの出力は、[データビューア] ビューで表示できます。
3. 出力のポートをフィルタするには、表示するポートを選択します。
4. フィールドの構造とデータを表示するには、階層フィールドを選択します。
階層フィールドの詳細は、階層タイプパネルに表示されます。
5. 結果を JSON ファイルにエクスポートするには、階層フィールドを右クリックして [データのエクスポート] を選択します。

階層データプレビューのルールおよびガイドライン

階層データプレビューを使用する際には、次のルールとガイドラインを考慮します。

- 階層データ型を含む動的マッピングはサポートされません。
- 日付フィールドは、データビューアで、追加のタイムスタンプコンポーネント (00:00:00) を持ちます。
- <Install location>/logs/<node name>/services/DataIntegrationService/disLogs/ms にあるログを表示できます。
ログを Developer tool ツールから表示することはできません。
- ウィンドウ化用に設定されている式トランスフォーメーションの階層データはプレビューできません。
- 複数の出力グループを含むトランスフォーメーションの階層データはプレビューできません。

ノーマライザ

ルーター

共有体

アップデートストラテジ

第 10 章

階層データ処理設定

この章では、以下の項目について説明します。

- [階層データの変換, 181 ページ](#)
- [リレーショナルデータまたは階層データの構造データへの変換, 182 ページ](#)
- [リレーショナルデータまたは階層データのネストされた構造データへの変換, 184 ページ](#)
- [階層データからの要素の抽出, 192 ページ](#)
- [階層データのフラット化, 194 ページ](#)

階層データの変換

Spark エンジン上の階層データを処理するマッピングを開発するためのタスクを簡素化する**階層変換ウィザード**を、Developer tool で使用します。このウィザードは、マッピングに 1 つ以上のトランスフォーメーションを追加し、出力ポートを作成します。出力ポートの式では複合関数と演算子を使用して、リレーショナルデータを階層データに、または階層データをリレーショナルデータに変換します。

次の表は、階層変換ウィザードと、これらのウィザードの用途を一覧表示します。

階層変換ウィザード	用途
構造ポートの作成	リレーショナルデータと階層データを構造データに変換するため。例えば、1 つ以上のカラムの任意のデータ型のデータを、構造型の階層データカラムに変換する場合。
ネストされた複合ポートの作成	2 つのテーブルのリレーショナルデータと階層データを、ネストされた構造データに変換するため。任意のデータ型の 1 つ以上のカラムを選択可能。例えば、2 つのテーブルのカラム内のデータを、ネストされたスキーマを持つ構造型の階層データカラムに変換する場合。このウィザードは、構造の配列を持つ構造を作成する。
複合ポートから抽出	階層データをリレーショナルデータに変換するため、または階層データを変更するため。 実行する変換の例: - 階層データに含まれるプリミティブデータ型の 1 つ以上の要素を、リレーショナルデータカラムに変換する。 - 階層データに含まれる複合データ型の 1 つ以上の要素を、階層データカラムに変換する。
複合ポートのフラット化	階層データをリレーショナルデータに変換するため。例えば、階層データの 1 つ以上の要素をリレーショナルデータカラムに変換する場合。

リレーショナルデータまたは階層データの構造データへの変換

1 つまたは複数のカラム内のリレーショナルデータまたは階層データは、構造型の階層データに変換できます。Developer tool で **【構造ポートの作成】** ウィザードを使用し、変換を実行します。

例えば、リレーショナルテーブルに市区町村、都道府県、郵便番号の 3 つのカラムが含まれるとします。この 3 つのカラム内のデータを、1 つの階層カラムに変換するマッピングを作成します。トランスフォーメーションの 3 つのポートを選択し、**【構造ポートの作成】** ウィザードを使用して、選択したポートを要素として持つ構造データを生成します。

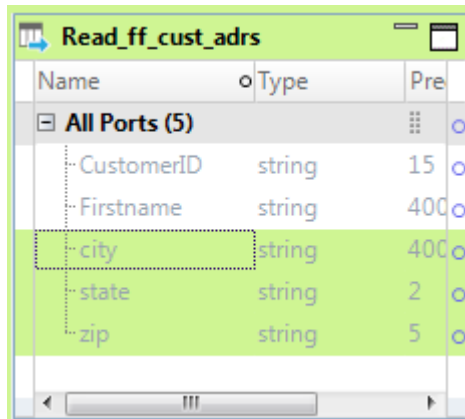
このウィザードは、次のタスクを実行します。

- 選択するポートに基づいて、複合データ型定義を作成します。
- 式トランスフォーメーションをマッピングに追加します。
- 構造データを表す構造出力ポートを作成します。
- 構造データを生成するための、STRUCT_AS 関数を使用する式を作成します。

構造ポートの作成

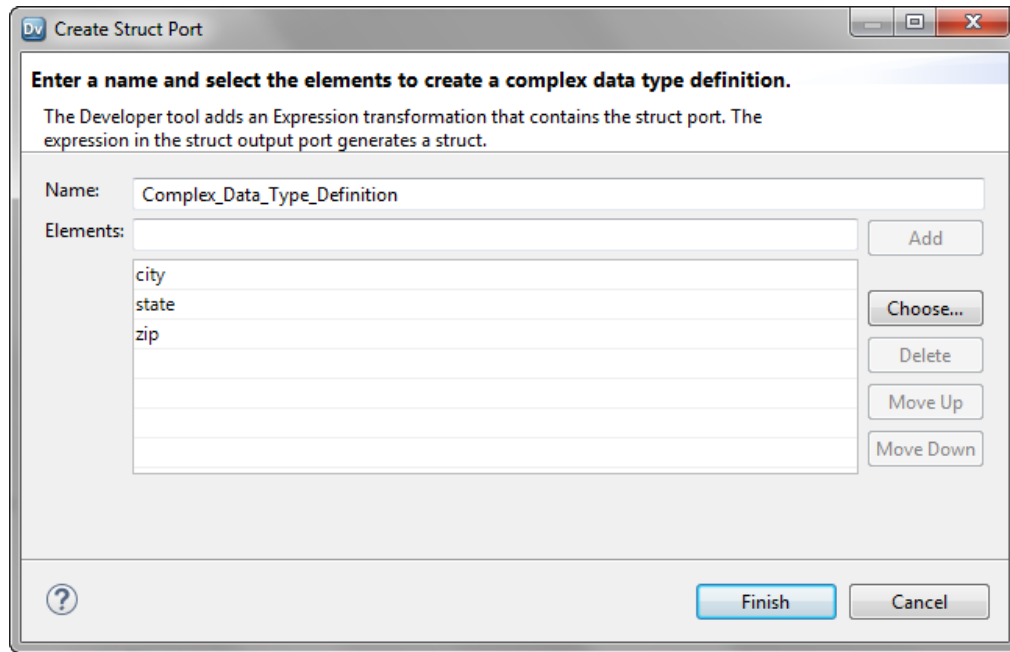
【構造ポートの作成】 ウィザードを使用して、1 つまたは複数のポートを介して構造データに渡されるデータを変換します。

1. トランスフォーメーションで、構造データの要素として変換する 1 つまたは複数のポートを選択します。
選択したポートによって、複合データ型定義の要素も決まります。



2. 選択したポートを右クリックし、**【階層変換】** > **【構造ポートの作成】** を選択します。

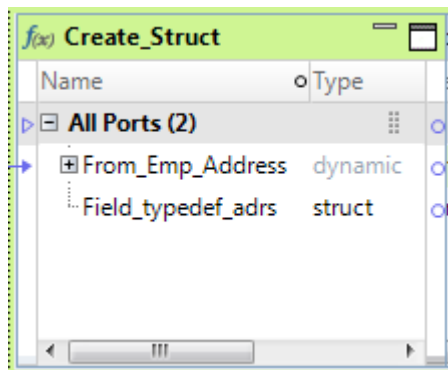
選択したポートのリストを含む【構造ポートの作成】ウィザードが表示されます。



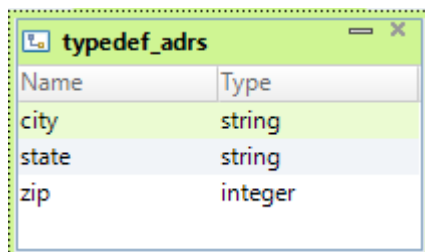
3. 必要に応じて、【名前】ボックスで、複合データ型定義の名前を変更します。
例えば、typedef_address とします。
4. 必要に応じて、【選択】をクリックし、トランスフォーメーション内のその他のポートを選択します。
5. 【完了】をクリックします。

マッピング内に、次の変更内容が表示されます。

- マッピングに、アップストリームトランスフォーメーションからの1つの構造出力ポートと1つの動的ポートを持つ、新しい式トランスフォーメーション Create_Struct が含まれます。



- 型定義ライブラリに、新しい複合データ型定義が含まれます。



- 構造出力ポートは、この複合データ型定義を参照します。
- 構造出力ポートに、STRUCT_AS 関数を持つ式が含まれます。以下に例を示します。

```
STRUCT_AS(:Type.Type_Definition_Library.typedef_address,city,state,zip)
```

リレーショナルデータまたは階層データのネストされた構造データへの変換

2つのトランスフォーメーション内の1つまたは複数のカラム内のリレーショナルデータまたは階層データは、ネストされた構造データに変換できます。Developer tool で **【ネストされた複合ポートの作成】** ウィザードを使用し、変換を実行します。

このウィザードによって、2つのテーブルからのリレーショナルデータは、ネストされたデータ型定義を持つ構造データに変換されます。

例えば、リレーショナルテーブルに、従業員の銀行口座の詳細が格納されているとします。もう1つのテーブルに、銀行の詳細が格納されています。この2つのテーブル内のデータを、支払いシステムが処理できる1つの階層形式に変換するマッピングを作成します。2つのトランスフォーメーション内のポートを選択し、**【ネストされた複合ポートの作成】** ウィザードを使用します。このウィザードによって、各銀行用の配列要素が格納された構造データが生成されます。この配列には、従業員の銀行口座の詳細用の構造要素が格納されます。

リレーショナル入力

Emp_Details テーブルに、従業員 ID、名前、住所、銀行 ID、銀行口座番号の各カラムが格納されています。

Bank_Details テーブルに、銀行 ID、銀行名、SWIFT コードの各カラムが格納されています。

構造出力

```
banks{
  bank_swift:integer
  [bank_name{account_number:integer,employee_id:integer,name:string,address:string}]
}
```

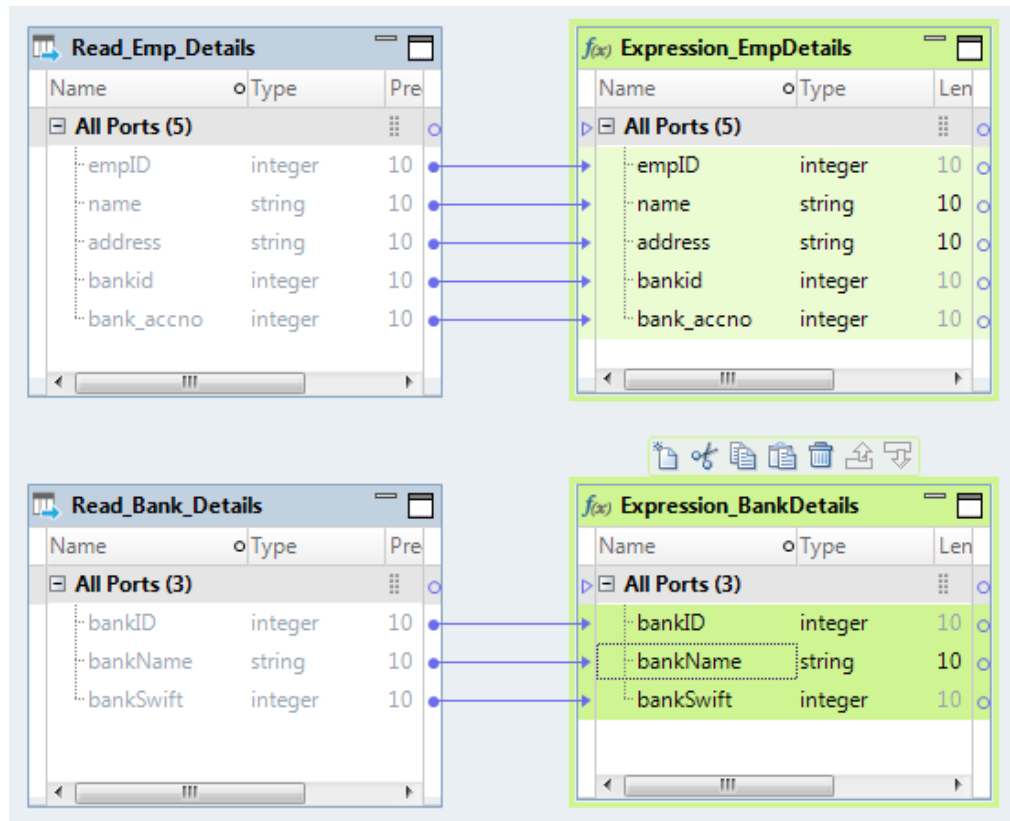
このウィザードは、次のタスクを実行します。

- 親テーブルと子テーブルからのソースデータを結合キーに基づいて結合するためのジョイナトランスフォーメーションを追加します。
親複合データ型定義のポートを含むトランスフォーメーションが、親トランスフォーメーションであり、もう一方のトランスフォーメーションが、子トランスフォーメーションです。
- 子複合データ型定義を作成し、子構造ポートを含む式トランスフォーメーションを追加します。
構造ポート内の式は、子複合データ型定義に基づいて、構造を生成します。
- 配列ポートを含むアグリゲータトランスフォーメーションを追加します。
配列ポート内の式は、子構造を配列要素として持つ配列を生成し、ポートごとのグループに基づいて、構造値をグループ化します。
- 親複合データ型定義を作成し、ネストされた複合ポートを含む式トランスフォーメーションを追加します。
ネストされた複合ポート内の式は、構造の配列を配列要素の1つとして持つ構造を生成します。

ネストされた複合ポートの作成

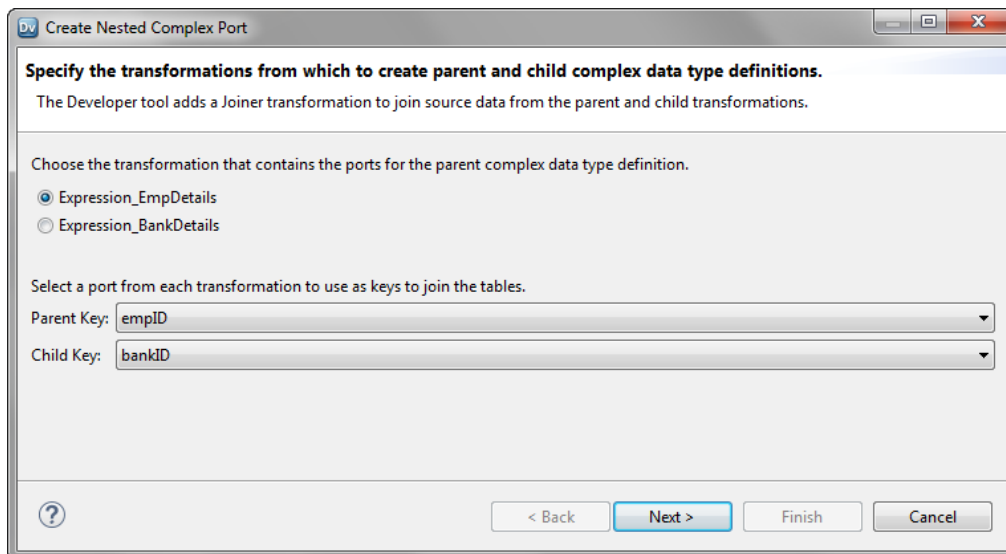
【ネストされた複合ポートの作成】ウィザードを使用して、2つのトランスフォーメーション内の1つまたは複数のポートを介し、ネストされたデータ型定義を参照する構造データに渡されるデータを変換します。ウィザードで親と子の複合データ型定義を作成する元になるトランスフォーメーションを指定します。

1. ネストされた複合ポートの作成先のマッピングまたはマップレットを開きます。
2. Ctrl キーを押し、ネストされた複合ポートの要素として変換する、2つのトランスフォーメーション内のポートを選択します。

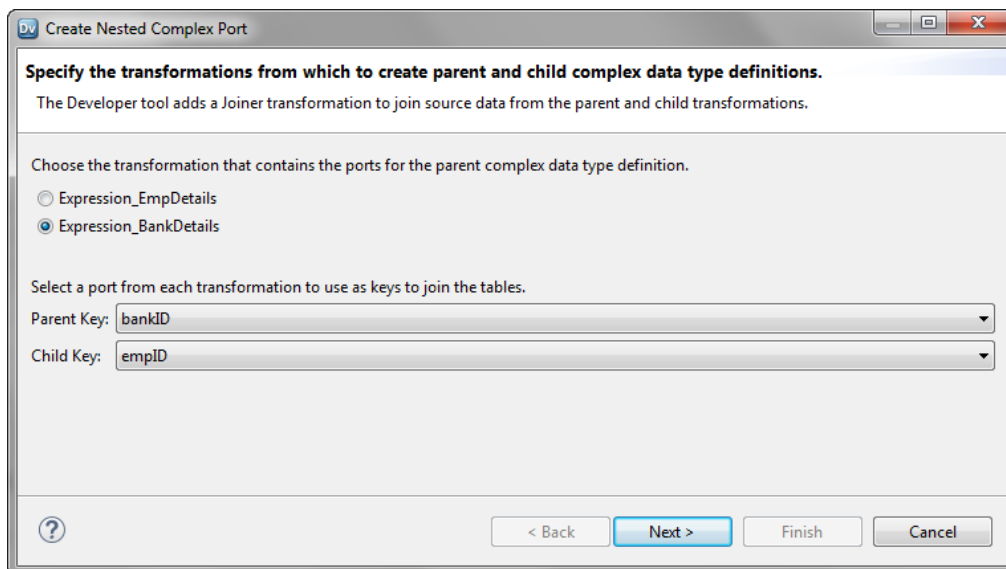


3. 選択したポートを右クリックし、【階層変換】 > 【ネストされた複合ポートの作成】を選択します。

【ネストされた複合ポートの作成】ウィザードが表示されます。



4. 親複合データ型定義のポートを含むトランスフォーメーションを選択し、テーブルを結合するための結合キーを選択します。



5. 【次へ】をクリックします。

子複合データ型定義を作成するためのウィザードページが表示されます。

Create Nested Complex Port

Enter a name and select the elements to create the child complex data type definition.

The Developer tool adds an Expression transformation that contains the child struct port. The expression in the output port generates a struct.

Name:

Elements:

empiD	
name	
address	
bankid	
bank_accno	

Buttons: Add, Choose..., Delete, Move Up, Move Down

Navigation: < Back, Next >, Finish, Cancel

- 必要に応じて、子複合データ型定義の名前を変更し、要素に変更を加えます。

Create Nested Complex Port

Enter a name and select the elements to create the child complex data type definition.

The Developer tool adds an Expression transformation that contains the child struct port. The expression in the output port generates a struct.

Name:

Elements:

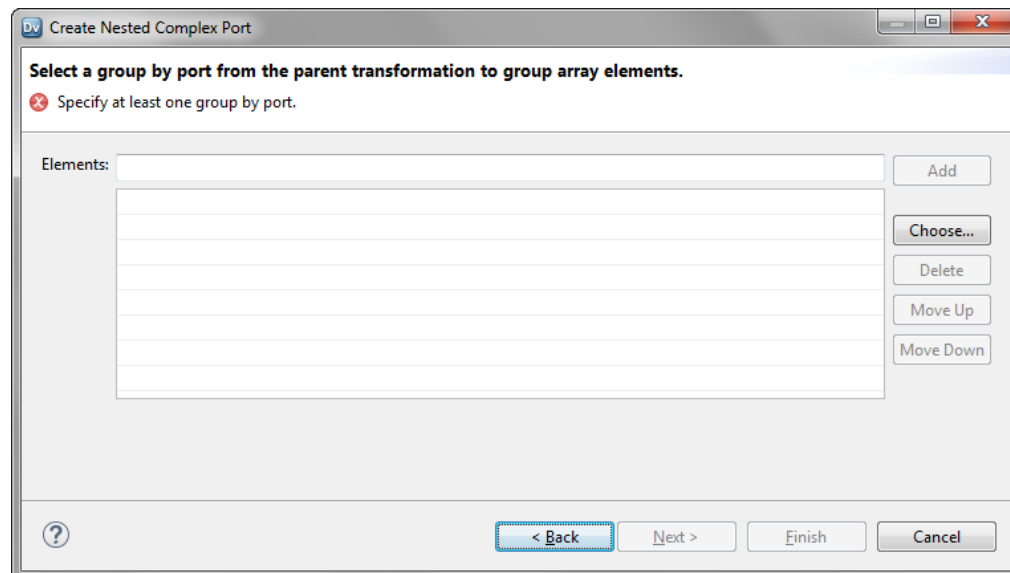
empiD	
name	
address	
bank_accno	

Buttons: Add, Choose..., Delete, Move Up, Move Down

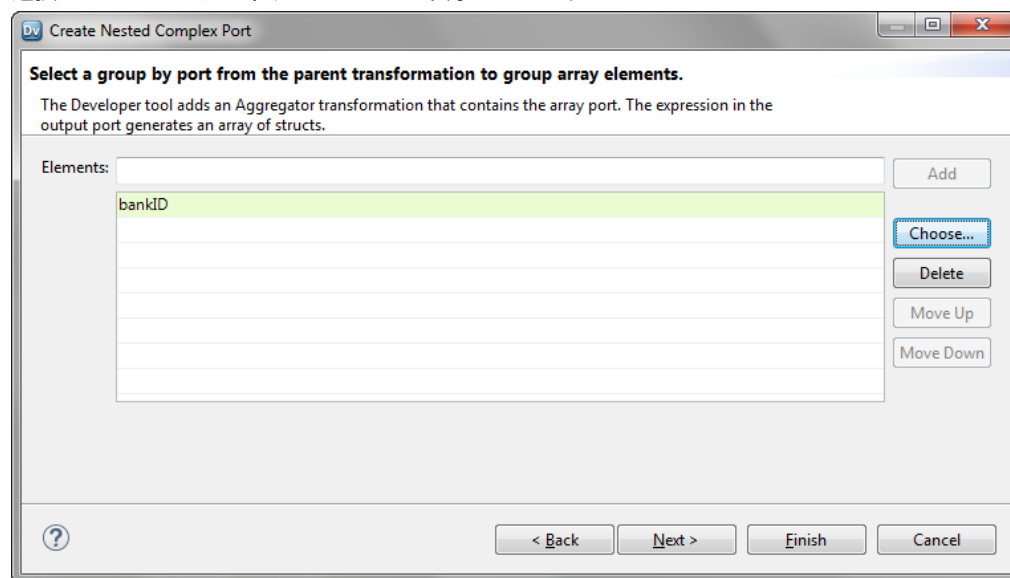
Navigation: < Back, Next >, Finish, Cancel

- 【次へ】をクリックします。

親トランスフォーメーションからグループ化ポートを選択するためのウィザードページが表示されます。

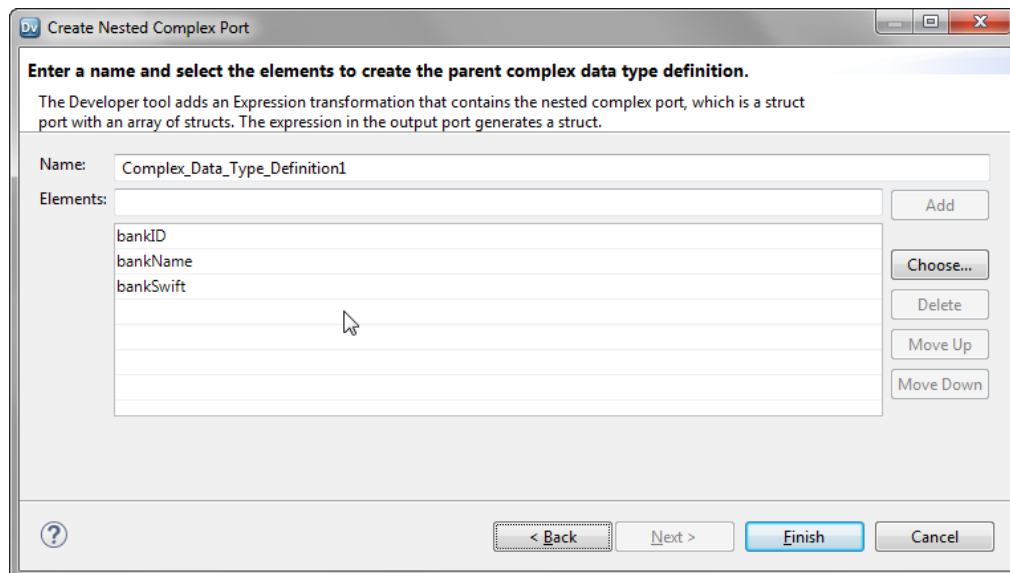


8. **【選択】** をクリックします。
【ポート】 ダイアログボックスが表示されます。
9. 親トランスフォーメーションからグループ化ポートを選択し、**【OK】** をクリックします。
選択したポートが、ウィザードページに表示されます。

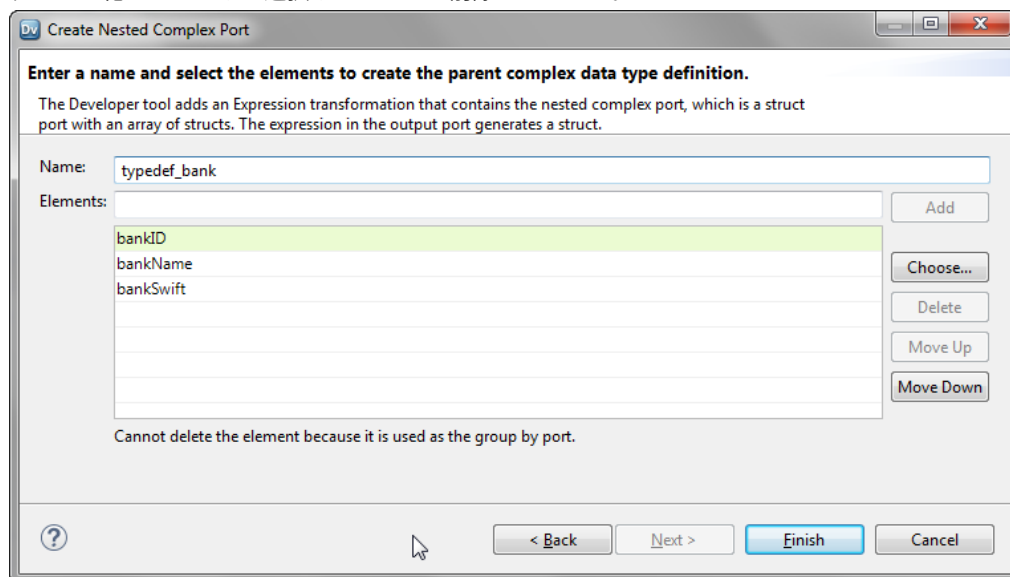


10. **【次へ】** をクリックします。

親複合データ型定義と、ネストされた複合ポートを作成するための最後のウィザードページが表示されます。



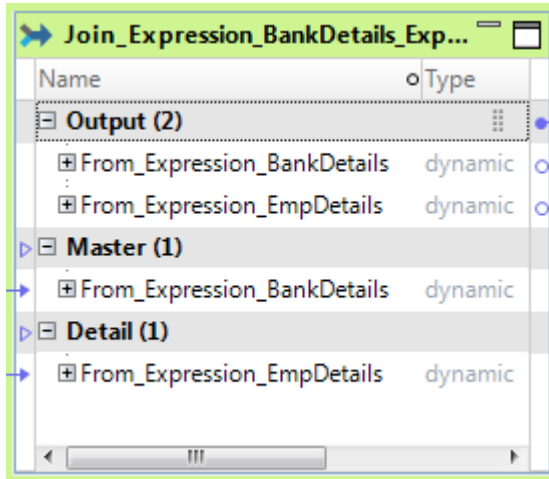
11. 必要に応じて、親複合データ型定義の名前を変更し、要素に変更を加えます。
グループ化ポートとして選択したポートは削除できません。



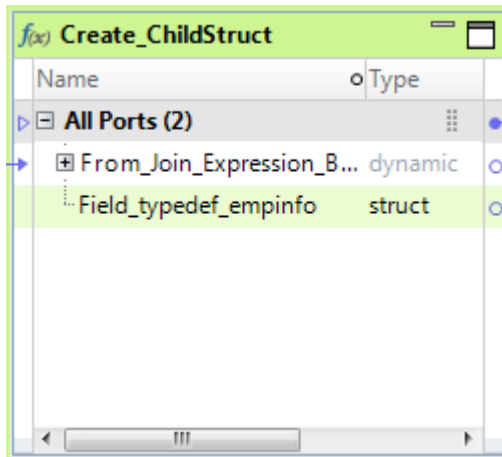
12. [完了] をクリックします。

マッピング内に、次の変更内容が表示されます。

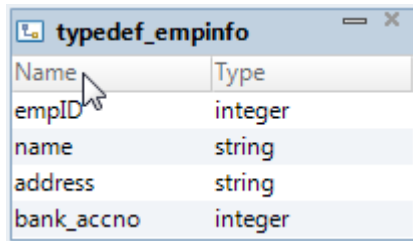
- このマッピングには、2つのテーブルを結合する1つの新しいジョイントトランスフォーメーション `Join_<transformation1>_<transformation1>`が含まれています。



- マッピングに、アップストリームトランスフォーメーションからの1つの構造出力ポートと1つの動的ポートを持つ、新しい式トランスフォーメーション `Create_ChildStruct`が含まれます。



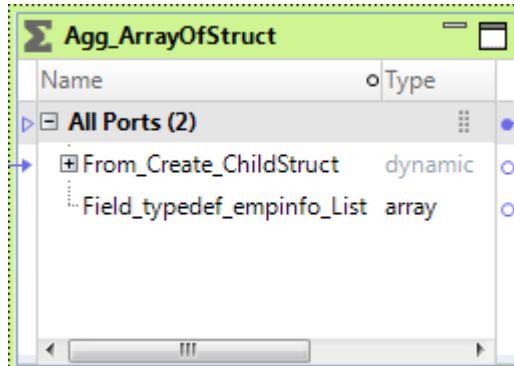
- 型定義ライブラリに、新しい子複合データ型定義が含まれます。構造出力ポートは、この子複合データ型定義を参照します。



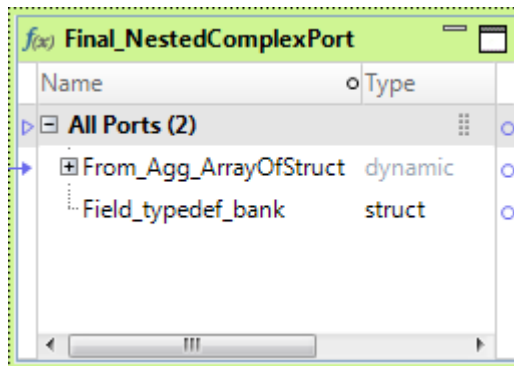
出力ポートに、STRUCT_AS 関数を持つ式が含まれます。

`STRUCT_AS(:Type.Type_Definition_Library.<child_typedef>,<comma_delimited_child_elements>)`

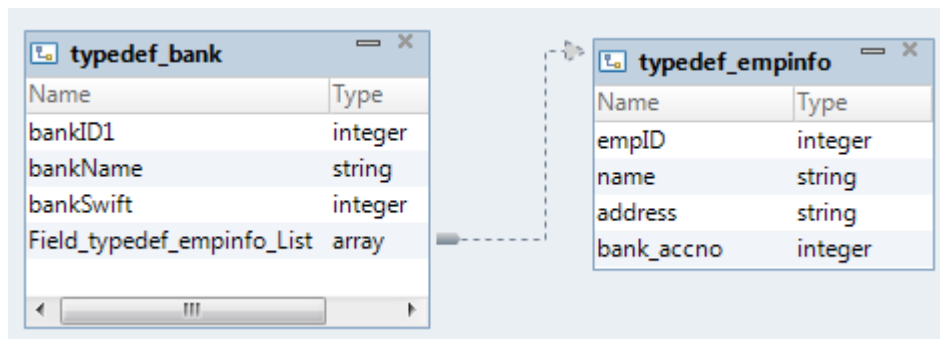
- マッピングに、子構造を配列にグループ化するためのグループ化ポートを持つ新しいアグリゲータトランスフォーメーション `Agg_ArrayOfStruct` が含まれます。



- マッピングに、アップストリームトランスフォーメーションからの1つの構造出力ポートと1つの動的ポートを持つ、新しい式トランスフォーメーション `Final_NestedComplexPort` が含まれます。



- 型定義ライブラリに、新しい親複合データ型定義が含まれます。これは、ネストされたデータ型定義です。構造出力ポートは、この親複合データ型定義を参照します。



出力ポートに、`STRUCT_AS` 関数を持つ式が含まれます。

`STRUCT_AS(:Type.Type_Definition_Library.<parent_typedef>,<comma_delimited_struct_elements>)`

階層データからの要素の抽出

プリミティブデータ型または複合データ型の要素を階層データから抽出できます。Developer tool で【複合ポートから抽出】ウィザードを使用し、変換を実行します。

選択する複合ポート内の要素のデータ型に基づいて、ウィザードは次の変換を実行します。

- 要素がプリミティブデータ型である場合、ウィザードは要素をリレーショナルデータに変換します。
- 要素が複合データ型である場合、ウィザードは要素を階層データに変換します。

ウィザードは、1 つ以上の抽出されたポートを含む式トランスフォーメーションを追加します。抽出するために選択する要素数によって、トランスフォーメーション内の出力ポート数が決まります。要素のデータ型によって、出力ポートのデータ型が決まります。出力ポート内の式は、複合演算子を使用して、複合ポートから要素を抽出します。

次の表に、選択する複合ポートのデータ型に基づくウィザードの変換を示します。

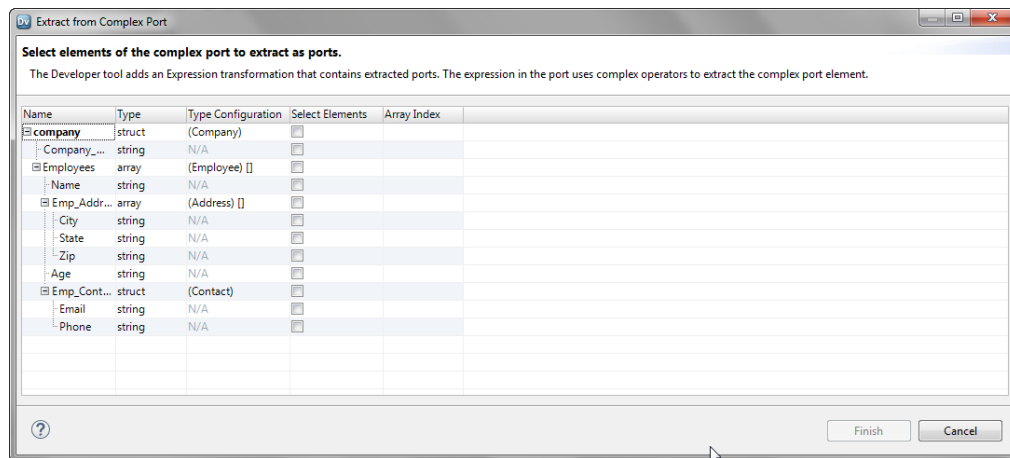
複合ポートデータ型	ウィザードの変換
配列	リレーショナルデータ。配列インデックスを指定した場合、ウィザードは配列内の要素を抽出します。デフォルトは 0 です。ウィザードは配列の最初の要素を抽出します。 階層データ。配列インデックスを指定しない場合、ウィザードは配列全体を抽出します。
構造	リレーショナルデータ。ウィザードは選択した構造内の要素を抽出します。
構造の配列	リレーショナルデータ。構造内の 1 つの要素を選択した場合、ウィザードはその要素を抽出します。ウィザードは配列インデックス値を必要とします。デフォルトは 0 です。 階層データ。配列を選択し、配列インデックスを指定した場合、ウィザードは配列内の構造要素を抽出します。配列インデックスを指定しない場合、ウィザードは配列全体を抽出します。
マップの配列	リレーショナルデータ。配列内のキーまたは値要素を選択し、配列インデックスを指定した場合、ウィザードはその要素を抽出します。 階層データ。配列を選択し、配列インデックスを指定した場合、ウィザードは配列内のマップ要素を抽出します。配列インデックスを指定しない場合、ウィザードは配列全体を抽出します。
構造の構造	リレーショナルデータ。親構造または子構造内の 1 つの要素を選択した場合、ウィザードはその要素を抽出します。 階層データ。親構造または子構造を選択した場合、ウィザードはその構造を抽出します。
マップの構造	階層データ。マップ要素を選択した場合、ウィザードはそのマップデータを抽出します。構造を選択した場合、ウィザードは別のポート内のその構造データを抽出します。

複合ポートからの要素の抽出

【複合ポートから抽出】ウィザードを使用し、階層データから要素を抽出します。

1. トランスフォーメーション内で、要素を抽出する複合ポートを選択します。
2. 選択したポートを右クリックし、【階層変換】 > 【複合ポートから抽出】を選択します。

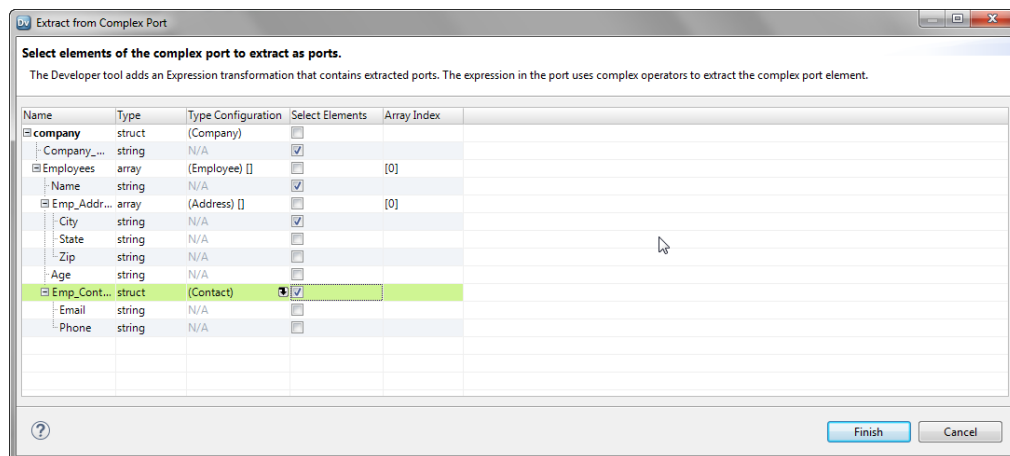
複合ポート内の要素のリストを含む【複合ポートから抽出】ウィザードが表示されます。



3. 【要素を選択】 カラムで、抽出する各要素のチェックボックスを選択します。

4. 配列要素を抽出するには、配列インデックスを【配列インデックス】カラム内で指定します。

配列インデックスを削除する場合、ウィザードは配列全体を抽出します。構造の 1 つの配列のために複合ポートから構造内の 1 つの要素を抽出するには、配列インデックスの値を指定する必要があります。

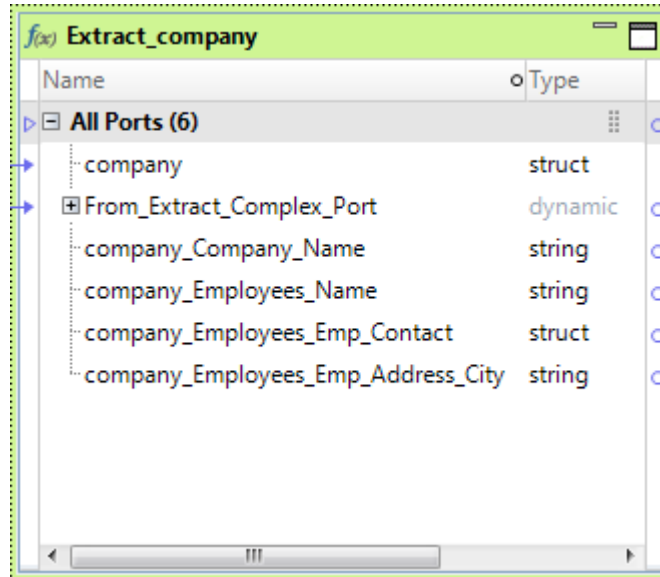


5. 【完了】をクリックします。

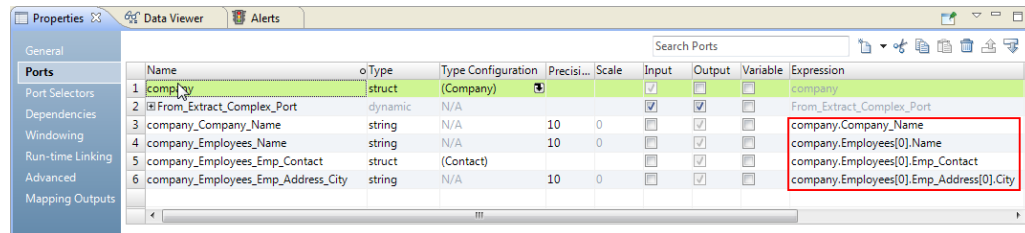
マッピング内に、次の変更内容が表示されます。

- このマッピングには、次のポートを持つ新しい式トランスフォーメーション `Extract_<complex_port_name>` が含まれます。
 - 入力ポートとしての要素の抽出元の複合ポート。
 - 抽出した要素の 1 つまたは複数の出力ポート。抽出するために選択した要素数によって、トランスフォーメーション内の出力ポート数が決まります。

- アップストリームトランスフォーメーションからのポートを持つ 1 つの動的ポート。



- 出力ポートには、要素を抽出するために複合演算子を使用する式が含まれます。
次の図に、[ポート] タブの出力ポートの [式] カラム内の出力ポートのための式を示します。



階層データのフラット化

階層データの要素をリレーショナルデータにフラット化できます。Developer tool で **【複合ポートのフラット化】** ウィザードを使用し、変換を実行します。

ウィザードは、階層データをリレーショナルデータに変換します。ネストされたデータ型の階層データがあるときは、複合データ型の特定の要素またはすべての要素を選択し、フラット化できます。

複合ポート内のデータ型に基づいて、ウィザードは次のタスクを実行します。

構造

- フラット化された出力ポートを持つ式トランスフォーメーションを追加します。出力ポートの式は、ドット演算子を使用して、構造内の要素を抽出します。
- フラット化された構造ポートを含むアップストリームトランスフォーメーションからのすべてのポートを持つ動的ポートが含まれる、最終式トランスフォーメーションを追加します。

配列

フラット化された出力ポートを持つノーマライズトランスフォーメーションを追加します。ウィザードは、ノーマライズビュー内の配列フィールドをフラット化します。

フラット化された配列ポートを含むアップストリームトランスフォーメーションからのすべてのポートを持つ動的ポートが含まれる、最終式トランスフォーメーションを追加します。

ネストされたデータ型

フラット化された出力ポートを持つ1つまたは複数の式トランスフォーメーションとノーマライズトランスフォーメーションを追加します。ネストされた複合ポートの子要素を選択すると、ウィザードによって親要素と子要素の両方がフラット化されます。

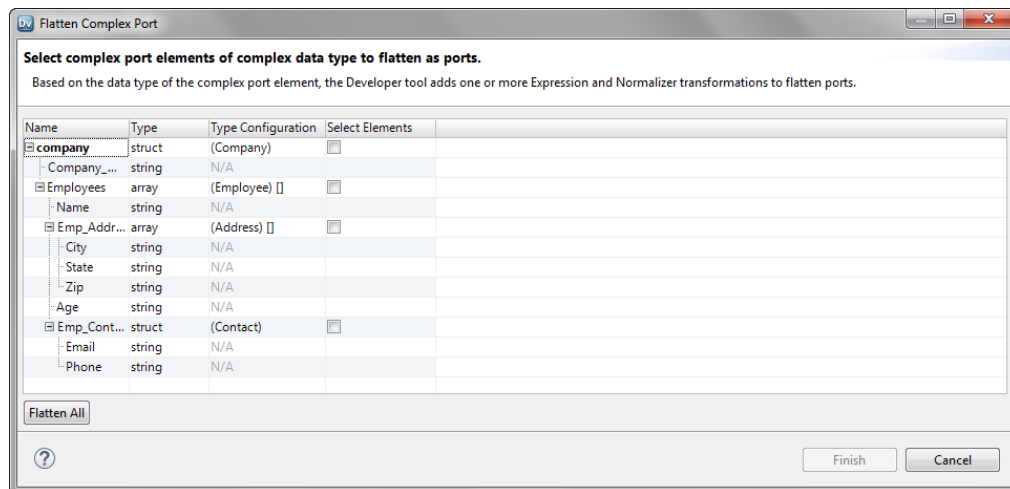
フラット化されたポートを含むアップストリームトランスフォーメーションからのすべてのポートを持つ動的ポートが含まれる、最終式トランスフォーメーションを追加します。

複合ポートのフラット化

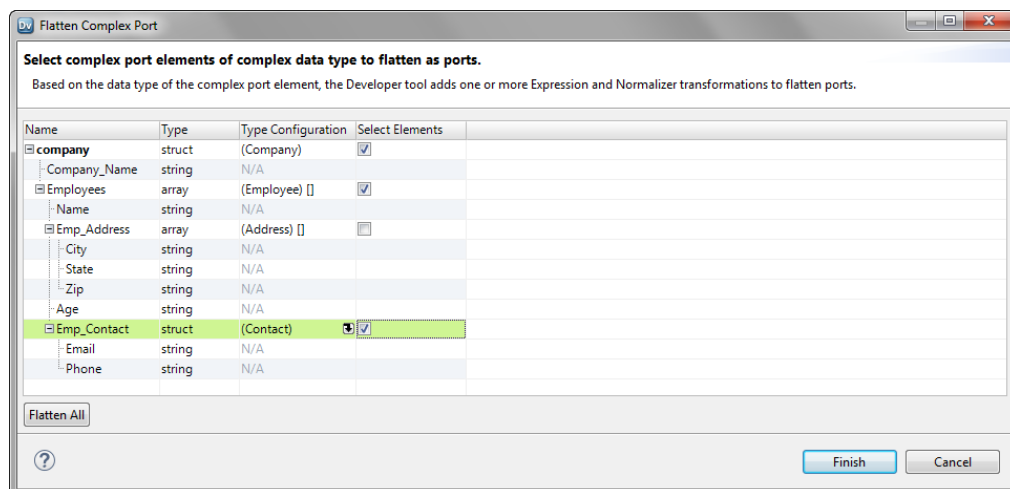
【複合ポートのフラット化】ウィザードを使用し、階層データをリレーショナルデータに変換します。

1. トランスフォーメーション内で、フラット化する複合ポートを選択します。
2. 選択したポートを右クリックし、【階層変換】 > 【複合ポートのフラット化】を選択します。

複合ポート内の要素のリストを含む【複合ポートのフラット化】ウィザードが表示されます。



3. 【要素を選択】カラムで、抽出する構造または配列データ型の要素のチェックボックスを選択します。
ネストされた複合ポートの子要素を選択すると、ウィザードによってフラット化する親要素が自動的に選択されます。



4. 必要に応じて、[すべてフラット化] をクリックし、複合ポート内の構造またはデータ型のすべての要素をフラット化します。

ウィザードは、複合ポート内の構造またはデータ型のフラット化するすべての要素を選択します。

5. [完了] をクリックします。

マッピング内に、次の変更内容が表示されます。

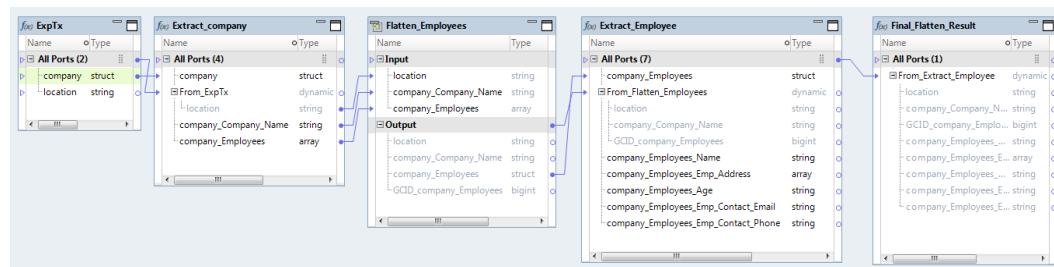
- フラット化するために選択した各構造要素について、マッピングに、次のポートを持つ新しい式トランスフォーメーション `Extract_<element_of_struct_type>` が含まれます。

- 要素を抽出する複合ポート。
- 抽出した要素の 1 つまたは複数の出力ポート。
出力ポートには、要素を抽出するために複合演算子を使用する式が含まれます。
- アップストリームトランスフォーメーションからのポートを持つ 1 つの動的ポート。

- フラット化するために選択した各配列要素について、マッピングには、1 つの入力グループと 1 つの出力グループを持つ新しいノーマライズトランスフォーメーション `Flatten_<element_of_array_type>` が含まれます。
出力グループには、配列要素のフラット化および正規化されたフィールドが含まれます。

- 最終式トランスフォーメーション `Final_Flatten_Result` は、アップストリームトランスフォーメーション内の複合ポートとその他のポートのためのフラット化されたポートを含む動的ポートを持ちます。

次の図に、マッピングの変更の例を示します。



第 11 章

スキーマが変更された階層データの処理

この章では、以下の項目について説明します。

- [スキーマが変更された階層データの処理の概要, 197 ページ](#)
- [階層データのスキーマの変更を処理する動的マッピングの開発方法, 198 ページ](#)
- [動的複合ポート, 199 ページ](#)
- [動的複合ポートの入力ルール, 201 ページ](#)
- [動的複合ポートのポートセレクト, 204 ページ](#)
- [動的式, 204 ページ](#)
- [複合演算子, 205 ページ](#)
- [複合関数, 206 ページ](#)
- [動的複合ポートのルールおよびガイドライン, 206 ページ](#)
- [最適化されたマッピング, 206 ページ](#)

スキーマが変更された階層データの処理の概要

動的複合ポートを使用して、階層データに対するスキーマの変更を管理できます。動的ポートを使用して、新規または変更された階層データの列を動的複合ファイルソースから受け取ります。その後、複合ポートのスキーマまたはメタデータの変更を管理するために、動的複合ポートを使用します。

Spark エンジンで階層データを処理するには、トランスフォーメーションで複合ポートを使用します。実行時のメタデータの変更を処理するには、動的ポートを含む動的マッピングを作成します。動的ポートは、複合データ型の生成されたポートを使用して、階層データをプロパゲートできます。ただし、実行時に階層データのスキーマが変更された場合は、動的複合ポートを使用して、マッピングに階層データを渡す必要があります。

複合ポートでは、配列またはマップの要素のデータ型または構造のスキーマに基づいて、型設定を指定する必要があります。動的複合ポートでは、階層列の型設定を指定する必要はありません。例えば、動的構造の動的データ型定義を指定する必要はありません。

動的複合ポートでは、動的マッピングで次のタスクを実行できます。

- 複合ポートのスキーマが実行時に変更された場合、新規または変更された、複合ポートの要素を受信する。
- 動的複合ポートの入力ポートで、階層データを変更する、または階層データの要素をフィルタする。
- 複合演算子を使用して実行時にスキーマが変更された可能性のある階層データの要素を抽出する。

- 複合関数を使用する動的式で、配列および構造を生成する。

動的マッピングの詳細については、『*Informatica Developer マッピングガイド*』を参照してください。

注: Databricks Spark エンジンでは、スキーマが変更された階層データを処理することはできません。

階層データのスキーマの変更を処理する動的マッピングの開発方法

Spark エンジンでスキーマが変更された階層データを処理するには、動的複合ソースとターゲット、動的ポート、複合ポート、動的複合ポート、複合演算子と関数、ポートセレクト、および動的式を使用して動的マッピングを開発します。

注: 動的マッピングを開発するためのタスクとタスクを実行する順序は、マッピングシナリオに応じて異なります。

次に、ランタイムに変化する可能性のある階層データを読み取り、書き込み、処理するためのマッピングを開発して実行するタスクの概要を示します。

動的複合ソースとターゲットを作成します。

複合ファイルソースが変更された場合、変更に対応するために読み取りトランスフォーメーションを設定できます。例えば、異なるデータソースを使用するか、データソースに基づいてデータオブジェクトを更新するために読み取りトランスフォーメーションを設定できます。複合ファイルターゲットが変更された場合、ターゲットの変更に対応するために書き込みトランスフォーメーションを設定できます。例えば、マッピングフローに基づいてカラムを生成するために書き込みトランスフォーメーションを設定できます。

実行時に複合ファイルデータオブジェクトからカラムを取得するために読み取りまたは書き込みトランスフォーメーションを設定できます。

動的ポートを作成し、入力ルールを定義します。

マッピング内のトランスフォーメーションを設定するときに、データフローに基づいて新規または変更されたカラムを受信する動的ポートを作成します。動的ポートが受信するカラムを決定し、生成されたポートの名前または順序を変更するように入力ルールを設定します。

動的複合ポートを作成し、入力ルールを定義します。

複合ポートのスキーマに対する変更を処理する動的複合ポートを作成します。動的複合ポートのスキーマまたは要素タイプを決定する入力ルールを設定します。

動的複合出力ポートのポートセレクトと動的式を作成します。

式で動的ポートまたはポートセレクトを使用して動的式を作成します。動的ポートを含めると、式は、動的ポートが生成する各ポートに基づいた要素を持つ複合データ型を生成します。ポートセレクトを含めると、式は、選択ルールによって決定された各ポートに基づいた要素を持つ複合データ型を生成します。

パラメータの作成。

パラメータを使用して実行時に値を変更できます。パラメータを使用して、マッピング内のソース、ターゲット、接続、ルールなどの値を変更します。

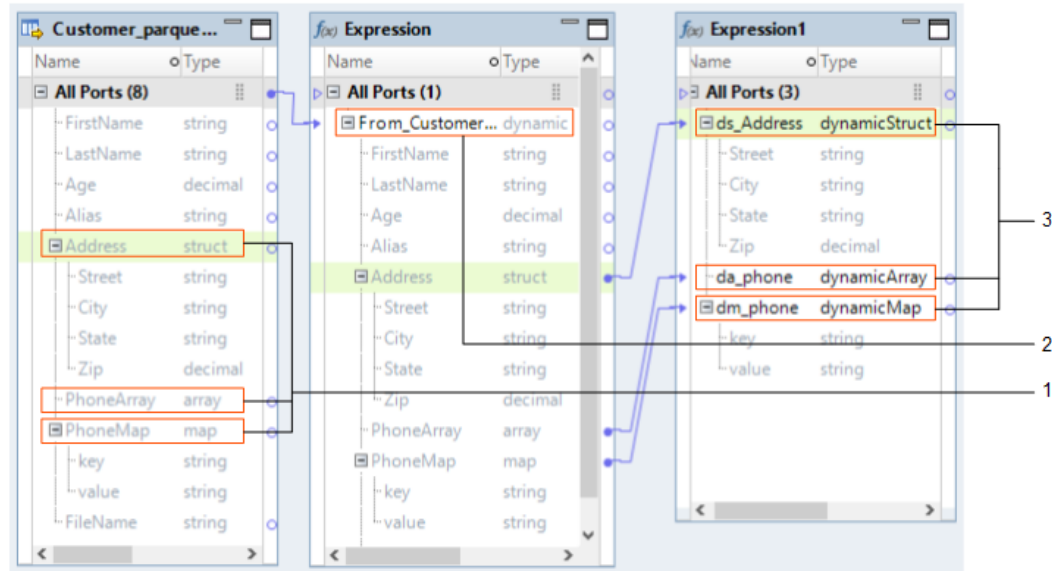
ランタイムリンクを作成します。

トランスフォーメーションは、マッピングを設計するときに直接リンクを作成できないように変更される場合があります。設計時にリンクを作成できない場合、ランタイムリンクを設定できます。

動的複合ポート

動的複合ポートは、入力ルールに基づいて、複合ポートまたは階層データカラムの 1 つ以上の要素を受信します。動的配列、動的マップ、動的構造などの動的複合ポートは、Spark エンジンで一部のトランスフォーメーションで使用できます。

次の図に、スキーマが変更された階層データを処理するために、マッピングで利用できるポートを示します。



1. 複合ポート
2. 動的ポート
3. 動的複合ポート

複合ポート

階層データを渡せる、または処理できるトランスフォーメーションの複合データ型のポート。配列ポート、マップポート、構造ポートは複合ポートです。型設定として、複合データ型に基づいて複合ポートプロパティを指定します。

動的ポート

アップストリームトランスフォーメーションから 1 つ以上のカラムを受信できるトランスフォーメーションのポート。動的ポートは、マッピングを介して渡すメタデータに基づいて生成されたポートとして、新規または変更されたカラムを受信できます。動的ポートの入力ルールで、生成されたポートが決定されます。

動的複合ポート

階層データのスキーマ変更に基づいて、新規または変更された要素を持つ階層データを渡せる、または処理できるトランスフォーメーションのポート。動的複合ポートの入力ルールで、階層データ要素または階層データのスキーマのデータ型が決定されます。

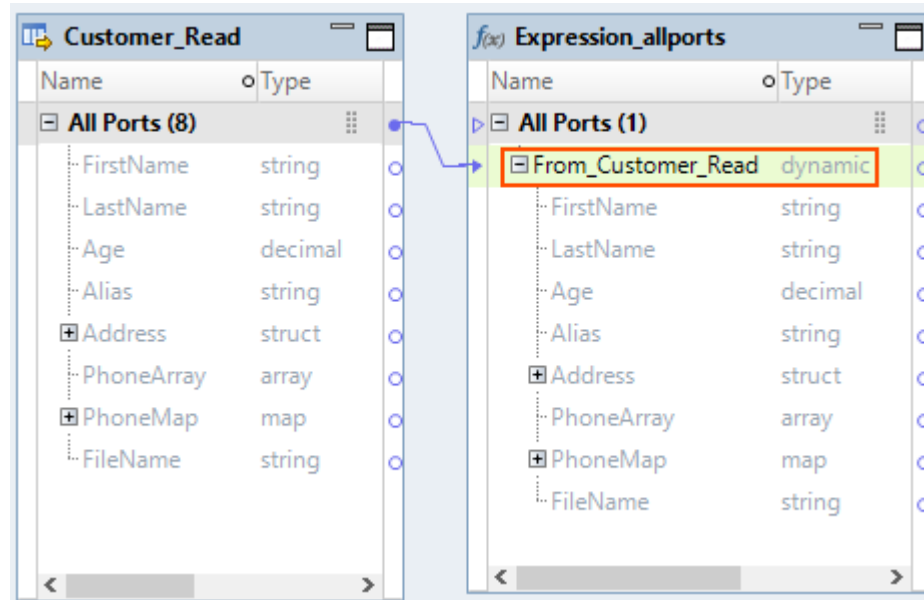
動的ポートと動的複合ポート

動的複合ポートと動的ポートは異なるものです。動的複合ポートは複合データ型の 1 つの階層カラムを表しますが、動的ポートはいずれかのデータ型の 1 つ以上の列を表します。

動的ポートは、マッピングを介して渡すメタデータに基づいて、実行時のカラムの変更に対応するために作成します。動的複合ポートは、実行時の階層カラムのスキーマの変更に対応するために作成します。

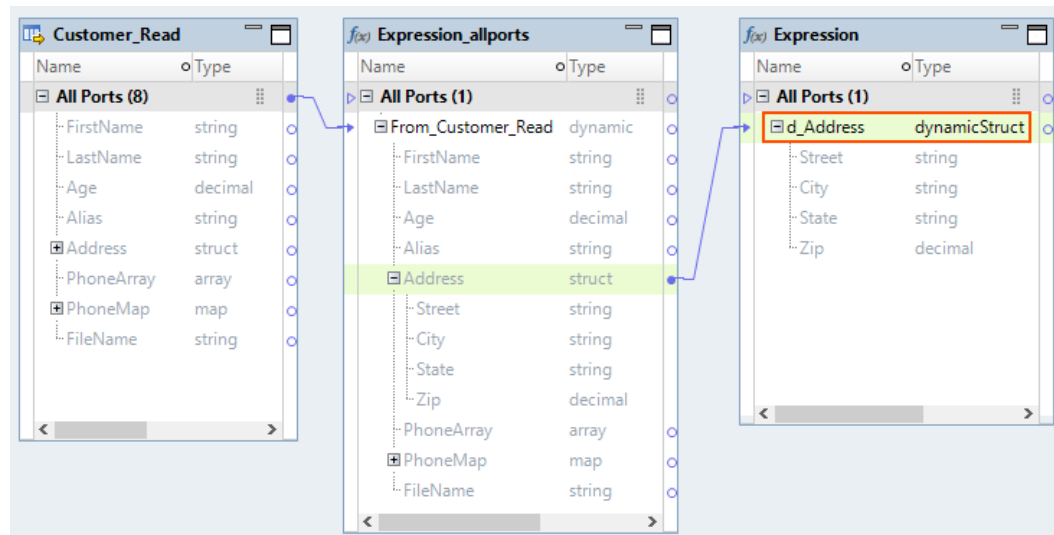
動的複合ポートはマップレットで使用できませんが、動的ポートはマップレットで使用できます。例えば、動的ポートはマップレット内のトランスフォーメーションに追加できます。ただし、動的ポートをマップレットの入力および出力インスタンスで使うことはできません。

次の図に、動的ポート From_Customer_Read を示します。



Expression_allports トランスフォーメーションの動的ポート From_Customer_Read には、型が string、decimal、struct、array、および map の生成されたポートが含まれています。生成されたポートは、アップストリームトランスフォーメーションのカラムに基づいて、実行時に変化する可能性があります。

次の図に、動的複合ポート d_Address を示します。



式トランスフォーメーションの動的構造ポート d_Address には、型が string と decimal の要素が含まれています。動的複合ポートの要素は、アップストリームトランスフォーメーションの階層カラムのスキーマ変更に基づいて、実行時に変化する可能性があります。

トランスフォーメーション内の動的複合ポート

動的複合ポートは、Spark エンジンの一部のトランスフォーメーション内で作成できます。

次のトランスフォーメーションで動的複合ポートがサポートされます。

- アグリゲータ
- 式
- フィルタ
- ジョイナ
- ルックアップ
- ノーマライザ
- ランク
- ルータ
- ソーター

動的複合ポートの入力ルール

動的複合ポートの入力ルールは、階層データ要素または階層データのスキーマのデータ型を決定する 1 つ以上のルールです。入力ルールは、アップストリームトランスフォーメーションの複合ポートとして表される階層データを変更するために定義します。

動的複合ポートに基づいて、対応する **【入力ルール】** ダイアログボックスで、次のように入力ルールを定義します。

動的配列

配列要素として 1 つ以上のデータ型を選択します。デフォルトの入力ルールには、すべてのデータ型の要素が含まれています。

動的マップ

マップ要素のキーと値として 1 つ以上のデータ型を選択します。キーとマップのデフォルトの入力ルールには、すべてのデータ型の要素が含まれています。

動的構造

要素名またはデータ型に基づき、構造に含めるまたは除外する要素を指定します。デフォルトの入力ルールではすべての要素が含まれます。

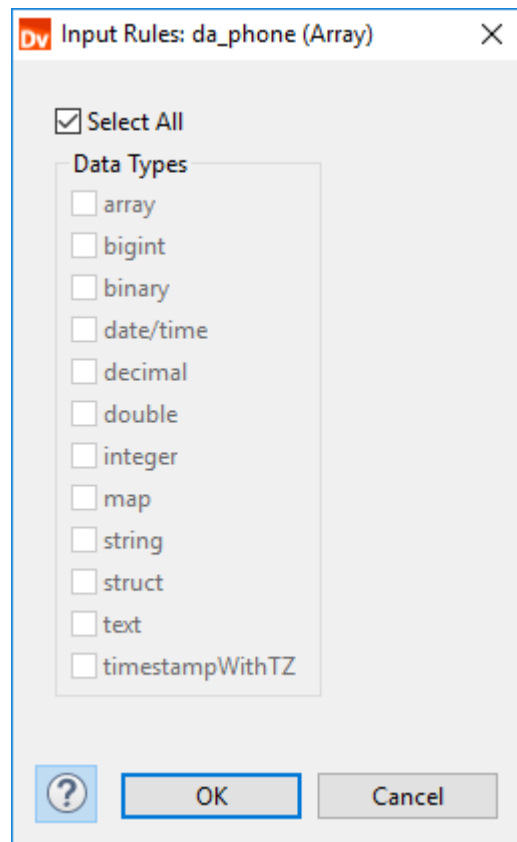
動的配列の入力ルール

動的配列の入力ルールによって、配列要素のデータ型が決定されます。1 つまたは複数のデータ型を選択できます。デフォルトの入力ルールには、すべてのデータ型の要素が含まれています。

例えば、2 つの異なる複合ファイルソースを処理するように動的マッピングを開発します。一方の複合ファイルの配列は string 要素のもの、もう一方の複合ファイルの配列は integer 要素のものとしします。すると、動的配列ポートの入力ルールとして string および integer データ型を選択できます。

配列要素のデータ型に確認がない場合は、すべてのデータ型を選択できます。

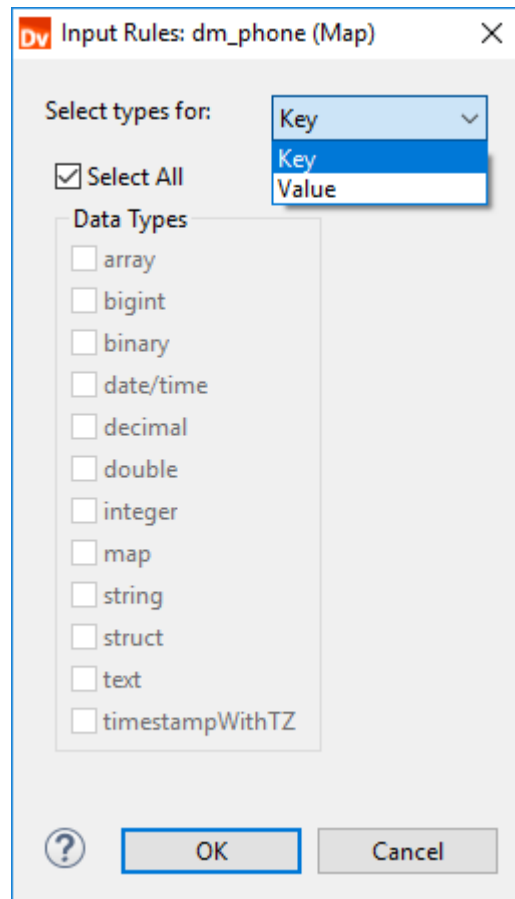
次の図に、動的配列ポートの [動的配列入力ルール] ダイアログボックスを示します。



動的マップの入カールール

動的マップの入カールールによって、マップのキーと値のペアのデータ型が決定されます。キーと値として1つ以上のデータ型を選択できます。キーとマップのデフォルトの入カールールには、すべてのデータ型の要素が含まれています。

次の図に、動的マップポートの「入カールール」ダイアログボックスを示します。



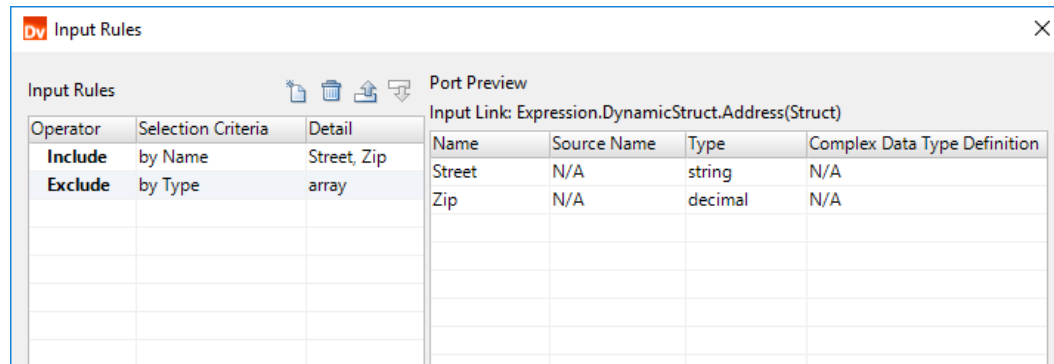
動的構造の入カールール

動的構造の入カールールによって、構造のスキーマが決定されます。入カールールは、アップストリームトランスフォーメーションの構造要素の名前または型に基づいて、定義することができます。

要素名またはデータ型に基づき、構造に含めるまたは除外する要素を指定します。複数のルールを定義できます。データ統合サービスは、入カールールリストに表示される順序でルールを適用します。デフォルトの入カールールではすべてのポートが含まれます。動的構造ポートの「含む」入カールールを1つ以上作成します。

注: 入カールールを使用して構造ポートの要素名を変更することはできません。

次の図に、動的構造ポートの「入力ルール」ダイアログボックスを示します。



動的複合ポートのポートセレクトラ

動的複合ポートのポートセレクトラを作成して、動的式または結合条件で使用できます。

ポートセレクトラは、複合動的ポートの次のトランスフォーメーションで使用できます。

- Expression トランスフォーメーション。STRUCT および ARRAY 関数でポートセレクトラを動的式として参照して、動的配列または動的構造を構築できます。
- ジョイントランスフォーメーション。結合条件の STRUCT および ARRAY 関数でポートセレクトラを参照できます。

ポートセレクトラの詳細については、『*Informatica Developer マッピングガイド*』を参照してください。

動的式

動的式を設定すると、動的複合ポートを構築できます。動的式では、STRUCT または ARRAY 関数を使用して動的構造または動的配列を構築します。動的複合出力ポートの動的式で、ポートセレクトラまたは動的ポートを参照できます。

プリミティブデータ型の出力ポートの動的式は、各ポートに対して式を実行して複数の出力ポートを生成します。動的複合出力ポートの動的式は、式を実行して動的複合ポートを構築し、複合データ型の出力ポートを 1 つ返します。

動的式のベースポートを指定する必要があります。ベースポートの設定で、出力複合ポートの要素の見え方が決定されます。例えば、ポートセレクトラの要素では、動的出力配列の要素が決定されます。構造のベースポートでは、出力構造ポートの要素のタイプが決定されます。

注: 動的式を設定するときに、Developer tool ではベースポートが指定されたかどうかは検証されません。

例 - 動的構造を構築する動的式

式トランスフォーメーションには、構造を構築する動的ポートがあります。

式トランスフォーメーションには、生成された以下の入力ポートがあります。

Name	String
Street	String

```
City      String
State     String
Zip       Integer
```

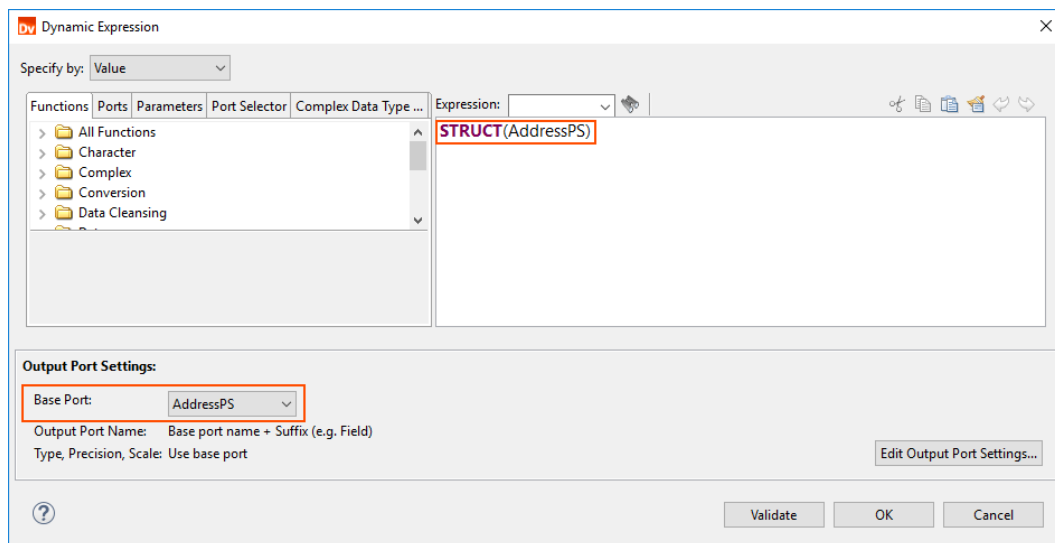
トランスフォーメーションには、d_structAddress という動的複合出力ポートが含まれています。構造を構築し、結果を d_structAddress に返そうとしています。

ポートセクタ AddressPS を作成し、名前によって以下のポートを含める選択ルールを定義します。

```
Street    String
City      String
State     String
Zip       Integer
```

次に、STRUCT 関数を含む動的式を作成します。

次の図に、ポートセクタ AddressPS を参照する動的式を示します。



【出力ポート設定】 領域で、ベースポートとしてポートセクタ AddressPS を指定します。

この動的式は、ポートセクタの要素を使用して次の構造を作成します。

```
d_structAddress{
  Street  String
  City    String
  State   String
  Zip     Integer
}
```

複合演算子

動的複合ポートの要素にアクセス、またはこの要素を抽出するには、式で複合演算子を使用できます。

例えば、次のようにドット演算子と添字演算子を使用すると、動的複合ポートの要素にアクセスできます。

```
dynamicStruct.element
dynamicArray[0]
dynamicMap[key]
```

複合演算子の詳細については、[「複合演算子」 \(ページ 174\)](#)を参照してください。

複合関数

動的式で ARRAY および STRUCT 関数を使用すると、動的配列や動的構造を構築できます。動的ポートまたはポートセクタを複合関数の引数として使用します。

動的構造を構築するときは、動的ポートまたはポートセクタのポート数に基づいて、構造の要素数が変化します。

動的構造を配列するときは、動的ポートまたはポートセクタのポート数に基づいて、配列のサイズが変化します。

複合関数の詳細については、「[複合関数](#)」(ページ 176)を参照してください。

動的複合ポートのルールおよびガイドライン

動的複合ポートを使用する場合は、次のルールとガイドラインを考慮します。

- 動的複合ポートはマップレットで使用できません。
- Filter トランスフォーメーション。フィルタ条件を使用して動的複合ポートを比較することはできません。ただし、複合演算子を使用すると、動的複合ポートの要素を抽出して、フィルタ条件内で使用することができます。例えば、次のフィルタ条件を使用できます。

```
dynamicStruct.field == 100
```

最適化されたマッピング

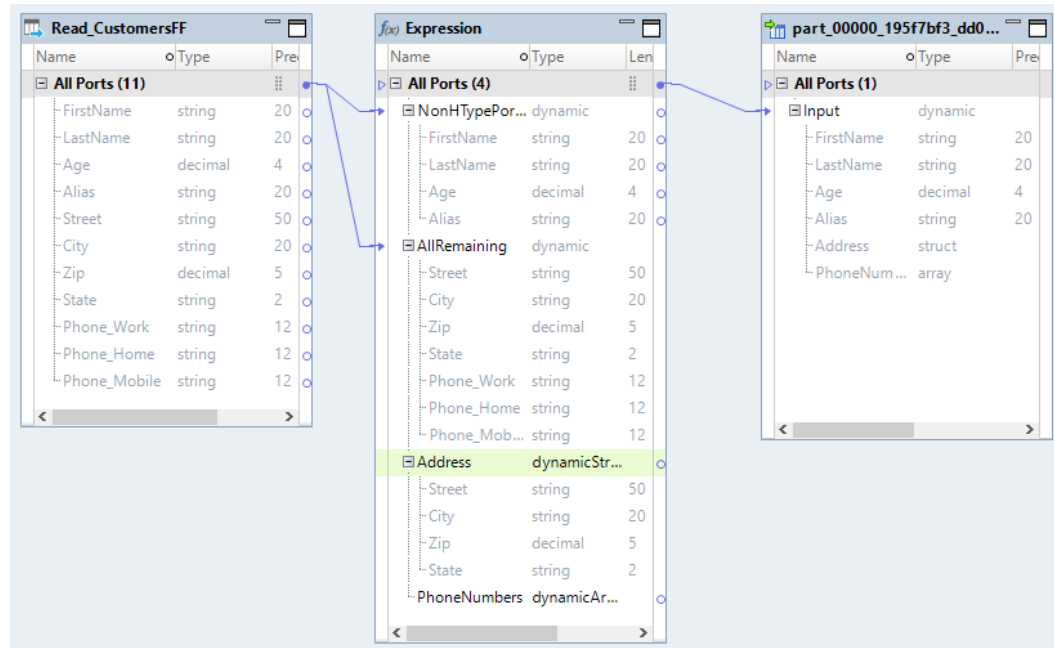
動的マッピングを開発するときに、動的マッピングを実行する前にマッピングをレビューして問題を修正したいことがあります。マッピングをレビューするために、最適化されたマッピングを表示できます。最適化されたマッピングは、コンパイルされたバージョンの動的マッピングです。

最適化されたマッピングを表示するには、マッピングエディタの何もない領域を右クリックして、**[最適化されたマッピングの表示]** をクリックします。

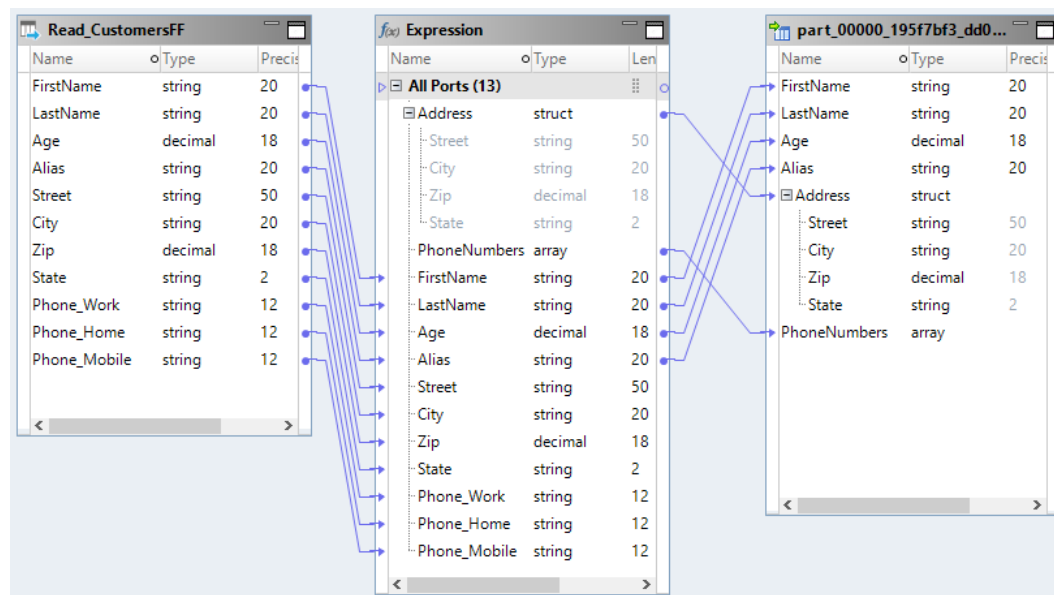
データ統合サービスにより最適化されたマッピングが生成されます。データ統合サービスによりマッピングがコンパイルされ、次のタスクが完了されます。

- 動的ポートを展開して、生成されたポートを静的ポートに変換する。
- 動的複合ポートを複合ポートに変換する。
- 動的式を解決する。
- パラメータ値を解決し、デフォルト値を設定する。
- 静的ポートをリンクする。
- ランタイムリンクを解決してポートを接続する。

次の図に、動的ポートを含む動的マッピング、動的複合ポート、およびマッピングフローリンクを示します。



次の図に、最適化されたマッピングを示します。この最適化されたマッピングでは、生成されたポートが静的ポートに変換され、動的複合ポートが複合ポートに変換され、ポートリンクが解決されます。



第 12 章

インテリジェント構造モデル

この章では、以下の項目について説明します。

- [インテリジェント構造モデルの概要, 208 ページ](#)
- [インテリジェント構造検出プロセス, 209 ページ](#)
- [ユースケース, 209 ページ](#)
- [マッピングでのインテリジェント構造モデルの使用, 210 ページ](#)
- [インテリジェント構造モデルのルールとガイドライン, 211 ページ](#)
- [作業を開始する前に, 211 ページ](#)
- [インテリジェント構造モデルを使用してデータを処理するためのマッピングの開発および実行方法, 212 ページ](#)
- [Informatica Intelligent Cloud Services アカウントの作成, 214 ページ](#)
- [インテリジェント構造モデルの作成, 215 ページ](#)
- [インテリジェント構造モデルのエクスポート, 215 ページ](#)
- [データ損失のチェック, 215 ページ](#)

インテリジェント構造モデルの概要

CLAIRE™ インテリジェント構造検出を使用して、Spark エンジンで実行されるマッピングの半構造化データと構造化データを解析できます。

構造がほとんどまたはまったくない、複雑で長いファイルは、わずかな解析でも認識が困難です。CLAIRE™ インテリジェント構造検出は、非構造化データの構造を自動的に検出できます。

CLAIRE™は、機械学習アルゴリズムを使用して、半構造化データファイルや非構造化データファイルのデータを解読し、データの基となる構造のモデルを作成します。インテリジェント構造モデルや、ファイルで検出されたデータのパターン、繰り返し、リレーション、およびフィールドタイプのモデルを、Informatica Intelligent Cloud Services で生成できます。

モデルを使用するには、データ統合からエクスポートし、Big Data Management のマッピング内のデータオブジェクトに関連付けます。Spark エンジンでマッピングを実行してデータを処理できます。マッピングはインテリジェント構造モデルを使用して、モデルに表されている構造に基づいて、入力ファイルのデータを抽出して解析します。

インテリジェント構造検出プロセス

CLAIRE™インテリジェント構造モデルをインテリジェント構造検出で作成できます。インテリジェント構造検出は、データ統合のサービスです。

サンプルファイルを提供すると、インテリジェント構造検出はその情報の基礎となる構造を判断し、その構造のモデルを作成します。インテリジェント構造モデルを作成したら、それを表示、編集、および調整できます。例えば、構造要素を除外する、または組み合わせることを選択できます。繰り返しグループを正規化できます。

モデルの調整が完了したら、それをエクスポートして、Big Data Management マッピング内のデータオブジェクトに関連付けることができます。

次の図は、Intelligent Structure Discovery が、根底にあるデータパターンを解読し、そのデータパターンのモデルを作成するプロセスを示しています。



Microsoft Excel、Microsoft Word のテーブル、PDF フォーム、および CSV ファイルの半構造化データや、非構造化テキストファイルのモデルを作成できます。また、XML ファイルや JSON ファイルなどの構造化データのモデルも作成できます。

ログファイル、クリックストリーム、顧客の Web アクセス、エラーテキストファイル、その他業界標準に準拠しないインターネットデータ、センサーデータ、デバイスデータなど、ファイルの構造がきわめて複雑で、検出に時間とコストがかかる場合でも、データをすばやくモデル化できます。

ユースケース

あなたは、保険会社の運用グループで働いています。あなたが所属するチームでは、運用上の問題やセキュリティの問題を特定する目的で、Web ログを処理しようとしています。

バックエンドシステムは、会社内のさまざまな支店で行われたシステムアクセスやセキュリティのアクティビティに関するデータを収集します。収集されたデータは、企業のデータセンターに保存されて、Amazon S3 ストレージでホストされます。

チームは、過去数週間に発生したほとんどのエラーとシステムダウンタイムの原因となった運用上の問題のタイプを理解しようと努めています。その後は監査のためにデータを保存する計画です。

データアナリストがデータで作業を開始できるようにするには、Amazon S3 の入力バケットのデータを解析して、アクション可能なデータを生成する必要があります。しかし、分析モデルの作成を目的としてデータを取捨選択するために必要な時間とリソースを費やすことはできません。Web ログからアクション可能なデータが作成されたことを確認するために、さまざまなマッピングやパラメータセットの開発が必要になる場合があります。

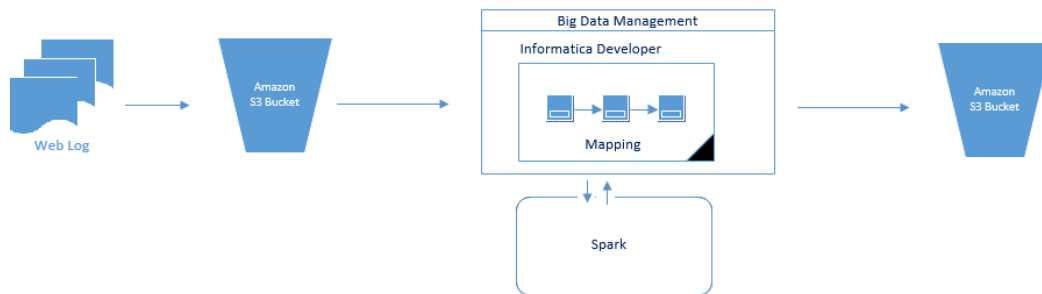
個別のトランスフォーメーションを手動で作成する代わりに、チームは、自動生成されたインテリジェント構造モデルを使用して、関連するデータセットを判別することができます。インテリジェント構造モデルをイン

テリジェント構造検出で作成します。これは、機械学習アルゴリズムを使用して構造化データファイルや非構造化データファイルのデータを解読してデータの基となる構造を発見するデータ統合のアプリケーションです。

Intelligent Structure Discovery は、入力ファイルデータ構造を表すインテリジェント構造モデルを作成します。インテリジェント構造モデルを使用してアクション可能なデータセットを出力するデータオブジェクトを使用して、マッピングを作成します。

マッピングは、Amazon S3 入力バケットのデータを取得した後、インテリジェント構造モデルを使用してデータを処理して準備します。その後、そのデータを Amazon S3 出力バケットに書き込むことができます。

次の図は、Amazon S3 入力バケットからデータを取得し、データを解析および準備し、その後データを Amazon S3 出力バケットに書き込むプロセスを示します。アナリストは、処理後のデータを使用してセキュリティの問題に対処し、運用管理を改善できます。



マッピングでのインテリジェント構造モデルの使用

マッピングでインテリジェント構造モデルを使用するには、モデルをデータオブジェクトと関連付けます。そうすることで、データオブジェクトは、モデルの作成に使用されたファイルと類似のファイル进行处理するために使用できます。

データオブジェクトに基づき、読み取りトランスフォーメーションをマッピングに追加できます。データ品質のチェックや構造化データのリレーショナル形式への変換など、データをさらに処理する場合、関連するダウンストリームトランスフォーメーションをマッピングに追加します。

マッピングの実行時に、読み取りトランスフォーメーションは、1 つまたは複数の入力ファイルを読み取り、データをフィールド、配列、構造に解析します。

モデルと入力に応じて、データオブジェクト出力には、プリミティブデータ型、複合データ型、ネストされたデータ型が含まれる場合があります。トランスフォーメーションでのこれらのデータ型の操作の詳細については、[「復号データ型」](#) ([ページ 158](#))を参照してください。

インテリジェント構造モデルのルールとガイドライン

インテリジェント構造モデルを使用する場合は、次のルールとガイドラインを考慮します。

- データオブジェクト用に選択するモデルは、データオブジェクトに期待する入力ファイルの構造とできるだけ一致させる必要があります。入力ファイルがモデルと一致していない、またはモデルと部分的にしか一致していない場合、大量の未定義データとデータ消失が発生する可能性があります。そのため、モデルを作成するときに、期待する入力ファイルを表すファイルを選択することが重要です。
- インテリジェント構造検出でインテリジェント構造モデルを作成するときに、異なる要素に重複する名前を使用しないでください。
- インテリジェント構造モデルをデータオブジェクトプロパティのカラムプロジェクションに追加する前に、インテリジェント構造モデルが有効であることを確認します。
- マッピングでインテリジェント構造モデルと一緒に使用できるのは、読み取りトランスフォーメーションのみです。マッピングが失敗するため、インテリジェント構造モデルと一緒に書き込みトランスフォーメーションを作成または使用しないでください。
- インテリジェント構造モデルを作成するときに、現在の Big Data Management バージョンに対応する Data Integration バージョンを選択してください。
- インテリジェント構造モデルを使用したデータオブジェクトは、サポートされる Hadoop 分割サイズの 256 MB より小さいサイズの PDF フォーム、Microsoft Word テーブル、および XML ファイルを解析します。
- インテリジェント構造モデルでは、PDF フォームフィールド内のデータを解析することはできますが、フィールドの外部にあるデータを解析することはできません。
- インテリジェント構造モデルは、Microsoft Word テーブル内のデータを解析できます。その他のデータについては未確認です。

作業を開始する前に

Big Data Management で使用するインテリジェント構造モデルを作成する前に、特定の要件を満たす必要があります。

次の要件を必ず満たします。

- アクティブな Informatica Intelligent Cloud Services アカウントを持っていることを確認します。
- Informatica Intelligent Cloud Services で、Developer tool のマッピングで使用するモデルを作成しエクスポートするための独立した組織を設定します。
- Informatica Intelligent Cloud Services 管理者が、インテリジェント構造モデルを作成しエクスポートするための正しいユーザー定義ロールを設定済みであることを確認します。
- インテリジェント構造モデルの作成に使用する Informatica Intelligent Cloud Services のライセンスバージョンが現在の Big Data Management バージョンに対応している必要があります。

登録とロールの詳細については、[Informatica Intelligent Cloud Services Administrator help](#) を参照してください。

インテリジェント構造モデルを使用してデータを処理するためのマッピングの開発および実行方法

インテリジェント構造モデルを組み込んだデータオブジェクトを持つマッピングを作成することで、データを解析できます。このマッピングは、Spark エンジンで実行し、データを処理します。

注: マッピングを開発するためのタスクとタスクを実行する順序は、マッピングシナリオに応じて異なります。

次に、インテリジェント構造モデルで処理でき、データをターゲットに書き込める任意のタイプのファイル内のデータを読み取って処理するためのマッピングを開発して実行する高度なタスクの概要を示します。

データ統合、Informatica Intelligent Cloud Services のサービスで、インテリジェント構造モデルを作成します。

代表的なファイルを使用して、インテリジェント構造モデルを作成します。モデルを .amodel ファイルにエクスポートします。ファイルをローカルに保存すると、それを関連するファイルストレージシステムにコピーできます。

詳細については、Data Integration ヘルプの「[Intelligent structure models](#)」を参照してください。

Big Data Management で、接続を作成します。

関連システムに保存されたファイル内のデータにアクセスするための接続を作成します。インテリジェント構造を組み込むデータオブジェクトを持たせる次のタイプの接続を作成します。

- Hadoop 分散ファイルシステム
- Amazon S3
- Microsoft Azure Blob
- Microsoft Azure Data Lake Store

ソースデータを読み取って解析するために、インテリジェント構造モデルを持つデータオブジェクトを作成します。

1. ソースとして保存されるファイルを表すための、インテリジェント構造モデルを持つデータオブジェクトを作成します。インテリジェント構造モデルを持つ、次のタイプのデータオブジェクトを作成できます。
 - 複合ファイル
 - Amazon S3
 - Microsoft Azure Blob
 - Microsoft Azure Data Lake Store
2. データオブジェクトのプロパティを設定します。
3. データオブジェクト読み取り操作で、プロジェクト階層データの列を複合データ型として設定します。[列プロジェクト] プロパティの [複合データ型としてのプロジェクト列] を有効にします。

注: インテリジェント構造モデルをデータオブジェクトとしてインポートするには、関連する Informatica Intelligent Cloud Services ライセンスが必要です。詳細については、「[「作業を開始する前に」 \(ページ 211\)](#)」を参照してください。

データをターゲットに書き込むためのデータオブジェクトを作成します。

1. データをターゲットストレージに書き込むためのデータオブジェクトを作成します。
2. データオブジェクトのプロパティを設定します。

警告: インテリジェント構造モデルをデータオブジェクト書き込み操作に関連付けしないでください。マッピングでインテリジェント構造モデルに関連付けられている書き込み操作を使用した場合、マッピングは無効になります。

マッピングを作成し、マッピングオブジェクトを追加します。

1. マッピングを作成します。
2. インテリジェント構造モデルを持つデータオブジェクトに基づいて、読み取りトランスフォーメーションを追加します。
3. マッピングロジックに基づいて、Spark エンジンでサポートされるその他のトランスフォーメーションを追加します。マッピングロジックに基づいて、ポートをリンクしトランスフォーメーションプロパティを設定します。
4. データをターゲットストレージまたは出力に渡すデータオブジェクトに基づいて、書き込みトランスフォーメーションを追加します。マッピングロジックに基づいて、ポートをリンクしトランスフォーメーションプロパティを設定します。

Spark エンジンで実行するようにマッピングを設定します。

以下のマッピングランタイムプロパティを設定します。

1. Hadoop を検証環境として選択し、Spark をエンジンとして選択します。
2. Hadoop を実行環境として選択し、Hadoop を接続として選択します。

Spark エンジンでマッピングを検証して実行します。

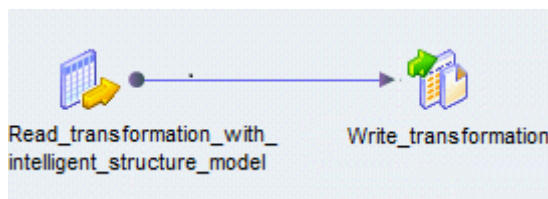
1. マッピングを検証し、エラーがあれば解決します。
2. 必要に応じて、Spark エンジン実行プランを表示し、ロジックをデバッグします。
3. マッピングを実行します。

マッピングの例

あなたの組織は、顧客 ID、品目コード、品目の数量などの注文書の詳細を分析する必要があります。注文書の詳細は、HDFS の Microsoft Excel スプレッドシート内に保存されています。このデータは、ストレージ用にテキストファイルに変更される必要があります。さらに、非公開の顧客データは削除する必要があります。インテリジェント構造を持つデータオブジェクトを使用することで、HDFS 内のファイルからすべての注文レコードを読み取るマッピングを作成します。このマッピングは、データを解析し、そのデータをストレージターゲットに書き込む必要があります。

監査のために抽出データを使用できます。

次の図に、マッピングの例を示します。



次のオブジェクトを HDFS マッピング内で使用できます。

HDFS 入力

入力オブジェクト `Read_transformation_with_intelligent_structure_model` は、HDFS 内に保存された Microsoft Excel ファイルを処理した読み取りトランスフォーメーションであり、フィールド出力を作成します。

Amazon S3 出力

出力オブジェクト Write_transformation は、Amazon S3 バケットを表す読み取りトランスフォーメーションです。

マッピングを実行すると、データ統合サービスはバイナリストリーム内のファイルを読み取り、それを読み取りトランスフォーメーションに渡します。読み取りトランスフォーメーションは、インテリジェント構造に従って関連データを抽出し、データを書き込みトランスフォーメーションに渡します。書き込みトランスフォーメーションは、データをストレージターゲットに書き込みます。

Hadoop ランタイム環境で実行するためのマッピングを設定できます。

マッピングを設定するには、次のタスクを実行します。

1. Hadoop クラスタからのファイルを読み取るための HDFS 接続を作成します。
2. 複合ファイルデータオブジェクト読み取り操作を作成します。次のパラメータを指定します。
 - データオブジェクト内のリソースとしてのインテリジェント構造。インテリジェント構造は、機密データを渡さないように設定されています。
 - HDFS ファイルの場所。
 - 読み取りデータオブジェクト操作内の入力ファイルフォルダの場所。
3. 複合ファイルデータオブジェクト読み取り操作をマッピングにドラッグアンドドロップします。
4. Amazon S3 接続を作成します。
5. Amazon S3 データオブジェクトの書き込みトランスフォーメーションを作成し、そのトランスフォーメーションをマッピングに追加します。

Informatica Intelligent Cloud Services アカウントの作成

Informatica Intelligent Cloud Services アカウントをまだ持っていない場合、作成します。

1. アカウントを作成するには、Informatica Intelligent Cloud Services ログインページ (<https://dm-us.informaticacloud.com/identity-service/home>) にアクセスします。
2. Informatica Intelligent Cloud Services ログインページで、**[アカウントがない場合]** をクリックします。
3. 設定ページで、連絡先の詳細と電子メールアドレスを入力します。電子メールアドレスをユーザー名として使用することを選択できます。

注: アクセスの詳細と、アカウントを確認するためのリンクが記載された登録電子メールを受け取ります。アカウントは、48 時間以内に確認するか、再登録する必要があります。
4. 登録電子メール内で、アカウント確認アクセスリンクをクリックします。

Informatica Intelligent Cloud Services パスワード設定ウィンドウが表示されます。
5. パスワード設定ウィンドウで、自分の Informatica Intelligent Cloud Services パスワードを定義し、**[ログイン]** をクリックします。

Informatica Intelligent Cloud Services サービスピッカーが表示されます。これで、データ統合にアクセスできるようになります。

インテリジェント構造モデルの作成

データ統合でインテリジェント構造モデルを作成します。簡略化されたサンプルファイルを使用してモデルを生成することをお勧めします。サンプルファイルには、解析対象のファイルに類似した代表的なデータ構造が必要です。

インテリジェント構造モデルの作成と改良の詳細については、「Informatica Intelligent Cloud Services マッピング」を参照してください。

1. Informatica Intelligent Cloud Services サービスピッカーで、**[データ統合]** を選択します。
2. **[新規]** > **[コンポーネント]** > **[インテリジェント構造]** をクリックします。
3. **[インテリジェント構造]** ページで、**[新規]** をクリックします。
4. **[新規インテリジェント構造]** ページで、名前と説明を入力します。インテリジェント構造モデルの名前を入力する必要があります。
5. インテリジェント構造モデルを作成するには、サンプルファイルを参照して **[構造の検出]** をクリックします。
インテリジェント構造検出によって、データのインテリジェント構造モデルが作成され、表示されます。
6. インテリジェント構造モデルを絞り込み、出力データの構造をカスタマイズするには、データノードを選択し、ノードの結合、除外、フラット化、または縮小を行います。
7. インテリジェント構造モデルを保存するには、**[OK]** をクリックします。

インテリジェント構造モデルのエクスポート

モデルを作成または編集した後、モデルをローカルドライブにエクスポートできます。

1. データ統合で **[参照]** ページを選択します。
2. **[エクスプローラ]** ページで、インテリジェント構造モデルを含むプロジェクトとフォルダに移動します。
[エクスプローラ] ページには、フォルダ内のすべてのアセットが表示されます。
3. 関連するインテリジェント構造モデルを含む行をクリックして選択します。**[アクション]** メニューで、**[編集]** を選択します。**[インテリジェント構造の詳細]** パネルで、**[編集]** をクリックします。
インテリジェント構造モデルは、別のページに表示されます。
4. **[インテリジェント構造]** ページで、**[ビジュアルモデル]** タブの右上隅にあるアイコンメニューを見つけ、**[モデルのエクスポート]** アイコンをクリックします。
インテリジェント構造検出によって、.amodel ファイルがローカルドライブにダウンロードされます。

データ損失のチェック

インテリジェント構造モデルを使用したマッピングで、入力ファイルの処理時にデータを損失したかどうかを検証するには、Spark 履歴ログを確認します。

- ▶ このログを取得するには、次のコマンドを Spark エンジンホストマシン上で実行します。
`yarn logs -applicationId<application ID>`

第 13 章

ステートフルコンピューティング

この章では、以下の項目について説明します。

- [ステートフルコンピューティングの概要, 216](#) ページ
- [ウィンドウ化構成, 217](#) ページ
- [ウィンドウ関数, 220](#) ページ
- [ウィンドウ化例, 225](#) ページ

ステートフルコンピューティングの概要

式トランスフォーメーションでウィンドウ関数を使用して、Spark エンジン上でステートフルコンピューティングを実行します。

ステートフルコンピューティングは、一部の状態を取得して、新しい状態と共に値を返す機能です。

ステートフルコンピューティングを実行するために、ウィンドウ関数を使用できます。ウィンドウ関数は、大規模なデータセットから小さいサブセットを取得して、処理と分析を行います。

ウィンドウ関数は、行のグループに対して動作し、各入力行の戻り値を計算します。ウィンドウ関数のこの特徴により、ウィンドウ関数なしでは簡潔に表現することが難しいデータ処理タスクを簡単に表現できるようになります。

ウィンドウ関数を使用して、次のタスクを実行します。

- アップストリームまたはダウンストリームの行からデータを取得します。
- 行のグループに基づいて累積合計を計算します。
- 行のグループに基づいて累積平均を計算します。

式トランスフォーメーションにウィンドウ関数を定義する前に、ウィンドウ化プロパティを構成してウィンドウを記述する必要があります。ウィンドウ化プロパティには、フレーム仕様、パーティションキー、およびオーダーキーが含まれます。フレーム仕様では、現在の行の全体的な計算に含まれる行を指定します。パーティションキーは、同じパーティション内にある行を決定します。オーダーキーは、パーティション内の行の順序を決定します。

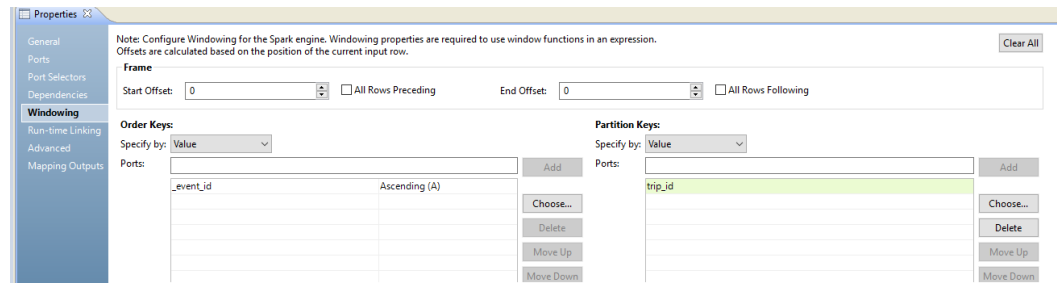
ウィンドウ化プロパティを構成した後、式トランスフォーメーションでウィンドウ関数を定義します。Spark は、ウィンドウ関数の LEAD と LAG をサポートしています。式トランスフォーメーションでは、集計関数をウィンドウ関数として使用することもできます。

ウィンドウ化構成

式トランスフォーメーションにウィンドウ関数を含める場合は、その関数に関連付けられているウィンドウ化プロパティを構成します。ウィンドウ化プロパティは、特定の入力行に関連付けられたパーティション化、順序付け、およびフレーム境界を定義します。

[ウィンドウ化] タブでウィンドウ化のトランスフォーメーションを構成します。

次の図は、[ウィンドウ化] タブを示しています。



[ウィンドウ化] タブで、次のプロパティグループを構成します。

フレーム

現在の行の位置からの物理オフセットに基づいて、現在の入力行のフレームに含まれる行を定義します。

集計関数をウィンドウ関数として使用する場合は、フレームを構成します。ウィンドウ関数の LEAD と LAG は、個々の行を参照し、フレーム仕様を無視します。

パーティションキー

入力行を別々のパーティションに分割します。パーティションキーを定義しない場合は、すべての行が単一パーティションに属します。

オーダーキー

パーティション内の行の順序を決定します。選択するポートによって、パーティション内の行の位置が決まります。オーダーキーは昇順または降順にできます。オーダーキーを定義しない場合、行には特定の順序がありません。

フレーム

フレームは、現在の行に対する相対的な位置に基づいて、現在の入力行の計算に含まれる行を決定します。

LEAD または LAG の代わりに集計関数を使用する場合は、ウィンドウフレームを指定する必要があります。LEAD と LAG は、個々の行を参照し、フレーム仕様を無視します。

開始オフセットと終了オフセットは、現在の入力行の前後に表示される行の数を表します。オフセット「0」は、現在の入力行を表します。例えば開始オフセットが-3 で終了オフセットが 0 の場合は、現在の入力行と現在の行より前の 3 つの行を含むフレームを表します。

次の図は、開始オフセットが-1 で終了オフセットが 1 のフレームを示しています。

Type	Category	Revenue	
Action	Video game	1000	
Arcade	Video game	1000	← 1 PRECEDING
Sports	Video game	2000	
Adventure	Video game	3000	← 1 FOLLOWING
Strategy	Video game	4000	

すべての入力行に対して、関数はフレーム内の行に対して集計操作を実行します。前述のフレームに対して SUM のような集計式を構成する場合、式はフレーム内の値の合計を計算し、入力行に対して値 6000 を返します。

現在の入力行を含まないフレームを指定することもできます。例えば開始オフセットが 10 で終了オフセットが 15 の場合は、現在の入力行より後の 10 番目の行から 15 番目の行までの合計 6 行が含まれるフレームを表します。

注: 開始オフセットは、終了オフセット以下でなければなりません。

【先行のすべての行】 および **【後続のすべての行】** というオフセットは、パーティションの最初の行とパーティションの最後の行を表します。例えば、開始オフセットが **【先行のすべての行】** で終了オフセットが -1 の場合、フレームには現在の行より前の 1 行と、それより前のすべての行が含まれます。

次の図は、開始オフセットが -0 で終了オフセットが **【後続のすべての行】** のフレームを示しています。

Genre	Recordings	Revenue	
Jazz	233	5000	
Gospel	214	1000	
Country	145	2000	All Rows Following ↓
Ethnic	154	9000	
Pop	317	4000	
Rock	237	2100	
Classical	221	3200	
EDM	153	950	
Hip Hop	839	2300	
Punk	415	7650	

パーティションキーおよびオーダーキー

パーティションキーとオーダーキーを構成して、行のグループを形成し、各パーティション内で行の順序またはシーケンスを定義します。

次のキーを使用して、ウィンドウ内の行のグループ化と順序の指定を行います。

パーティションキー

すべての入力にわたって計算を実行するのではなく、パーティションの境界を定義するようにパーティションキーを構成します。ウィンドウ関数は、現在の行と同じパーティション内に入る行にわたって動作します。

パーティションキーは、値またはパラメータによって指定できます。ポート名を使用するには、**【値】** を選択します。ソートキーリストパラメータを使用するには、**【パラメータ】** を選択します。ソート基準となるポートのリストを含んだソートキーリストパラメータを作成します。パーティションキーを指定しない場合は、すべてのデータが同じパーティションに含まれます。

オーダーキー

オーダーキーを使用して、パーティション内の行の順序を決定します。オーダーキーは、パーティション内の特定の行の位置を定義します。

オーダーキーは、値またはパラメータによって指定できます。ポート名を使用するには、**【値】** を選択します。ソートキーリストパラメータを使用するには、**【パラメータ】** を選択します。ソート基準となるポートのリストを含んだソートキーリストパラメータを作成します。また、データを昇順または降順に並べ替えることを選択する必要があります。オーダーキーを指定しない場合、パーティション内の行は特定の順序で配置されません。

例

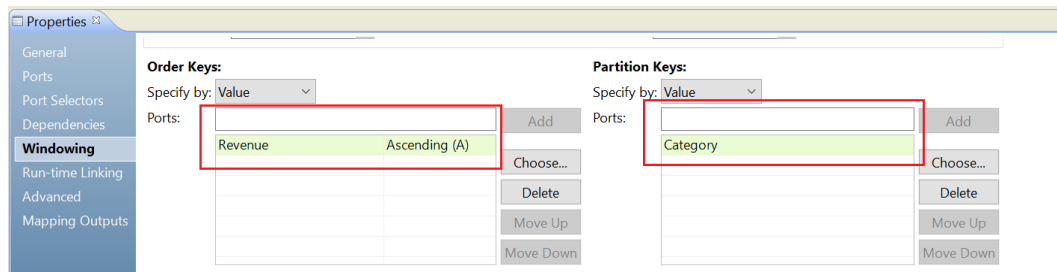
あなたはコーヒーと紅茶の店のオーナーです。売り上げが 1 番目と 2 番目のコーヒーと紅茶を計算したいと考えています。

以下の表に、製品、対応する製品カテゴリ、および各製品の収益を示しています。

Product	Category	Revenue
Espresso	Coffee	600
Black	Tea	550
Cappuccino	Coffee	500
Americano	Coffee	600
Oolong	Tea	250
Macchiato	Coffee	300
Green	Tea	450
White	Tea	650

カテゴリ別にデータを分割し、収益の降順でデータを並べます。

以下の図に、[ウィンドウ化] タブで設定するプロパティを示します。



以下の表は、カテゴリに従って 2 つのパーティションにグループ化されたデータを示しています。各パーティション内では、収益は降順に編成されます。

Product	Category	Revenue
Espresso	Coffee	600
Americano	Coffee	600
Cappuccino	Coffee	500
Macchiato	Coffee	300
White	Tea	650
Black	Tea	550
Green	Tea	450
Oolong	Tea	250

パーティション化仕様と順序付け仕様に基づいて、売り上げのよいコーヒー 2 つはエスプレッソとアメリカーノ、売り上げのよいお茶の 2 つは白茶と紅茶であると判断しました。

ウィンドウ化構成のルールとガイドライン

ウィンドウ化のトランスフォーメーションを構成するときには特定のガイドラインが適用されます。

ウィンドウ関数のウィンドウ化プロパティを定義する場合、次のルールとガイドラインを検討します。

- フレームを構成する場合、開始オフセットは終了オフセット以下でなければなりません。そうしないと、フレームは無効になります。
- 集計関数をウィンドウ関数として使用する場合は、フレーム仕様を構成します。LEAD と LAG は、オフセット値に基づいて動作し、フレーム仕様を無視します。
- 複合ポートはパーティションキーまたはオーダーキーとして使用できません。
- ランタイムエラーを避けるため、一意のポート名をパーティションキーおよびオーダーキーに割り当てます。
- パーティションキーとオーダーキーは、動的ポートと、同じ動的ポートの 1 つ以上の生成されたポートの両方は使用できません。動的ポートまたは生成されたポートのいずれかを選択する必要があります。

ウィンドウ関数

ウィンドウ関数は、行のグループに基づいてテーブルの各入力行の戻り値を計算します。

ウィンドウ関数は、現在の行に関連するテーブル行のセット全体にわたって計算を実行します。このタイプの計算は、集計関数を使用して実行することもできます。ただし、通常の集計関数とは異なり、ウィンドウ関数は行を単一の出力行にグループ化しません。行は一意の ID を保持します。

式トランスフォーメーションに、LEAD および LAG の分析ウィンドウ関数を定義できます。LEAD と LAG は、自己結合の必要なく、テーブル内の複数行へのアクセス権を付与します。

LEAD

LEAD 関数は、現在より先の行のデータを返します。

LEAD では、以下の構文を使用します。

LEAD (Column name, Offset, Default)

LEAD は、現在の行より後の、オフセットの行数分だけ離れた位置にある値を返します。LEAD 関数を使用して、現在の行の値と、後の行の値を比較します。LEAD 関数では、次の引数を使用します。

- カラム名。返すことになる後の行の値を格納しているカラムの名前です。
- オフセット。取得することになるデータを格納している、現在の行より後にある行の数です。例えば、オフセットが"1"の場合は1つ後の行にアクセスし、オフセットが"3"の場合は現在の行より3つ後の行にアクセスします。
- デフォルト。オフセットがパーティションの範囲の外側だった場合に返されるデフォルト値です。指定しないと、デフォルトは NULL になります。

LEAD 関数の詳細については、『Informatica トランスフォーメーション言語リファレンス』を参照してください。

LAG

LAG 関数は、前の行のデータを返します。

LAG では、以下の構文を使用します。

LAG (Column name, Offset, Default)

LAG は、現在の行より前の、オフセットの行数分だけ離れた位置にある値を返します。LAG 関数を使用して、現在の行の値と、前の行の値を比較します。LAG 関数では、次の引数を使用します。

- カラム名。返すことになる前の行の値を格納しているカラムの名前です。
- オフセット。取得することになるデータを格納している、現在の行より前にある行の数です。例えば、オフセットが"1"の場合は1つ前の行にアクセスし、オフセットが"3"の場合は現在の行より3つ前の行にアクセスします。
- デフォルト。オフセットがパーティションの範囲の外側だった場合に返されるデフォルト値です。指定しないと、デフォルトは NULL になります。

LAG 関数の詳細については、『Informatica トランスフォーメーション言語リファレンス』を参照してください。

ウィンドウ関数としての集計関数

ウィンドウ関数として、LEAD と LAG に加えて集計関数も使用できます。SUM や AVG などの集計関数をウィンドウ関数として使用するときは、ステートフル関数である MOVINGSUM、MOVINGAVG、CUME と類似する実行計算を実行できます。ウィンドウ関数には、特定の終了オフセットを設定できるため、ステートフル関数よりも柔軟です。

集計関数をウィンドウ関数として使用するには、ウィンドウ化プロパティ内にフレームを定義する必要があります。フレームは、計算のスコープを制限するために定義できます。集計関数は、フレームを超えて計算を実行し、各行に1つの値を生成します。

例

あなたは過去2年間、さまざまな数量の木材を販売してきた材木営業担当者であるとして。販売数量の現在までの累計を計算します。

次の表に、各販売 ID、日付、販売数量の一覧を示します。

Sale_ID	Date	Quantity
30001	2016-08-02	10
10001	2016-12-24	10
10005	2016-12-24	30
40001	2017-01-09	40
10006	2017-01-18	10
20001	2017-02-12	20

SUM 関数によって、すべての値が合計され、1 つの出力値が返されます。各行の累計を得るため、関数の境界に対してフレームを定義できます。

次の図に、[ウィンドウイング] タブで指定するフレームを示します。

Note: Configure Windowing for the Spark engine. Windowing properties are required to use window functions in an expression. Offsets are calculated based on the position of the current input row.

Frame

Start Offset: 0 ☐ All Rows Preceding End Offset: 0 ☐ All Rows Following

Order Keys:

Specify by: Value

Ports:

Date	Ascending (A)

Partition Keys:

Specify by: Value

Ports:

次のウィンドウ化プロパティを構成します。

- 開始オフセット: 前のすべての行
- 終了オフセット: 0
- オーダーキー: 昇順の日付
- パーティションキー: 指定なし

次の集計関数を定義します。

SUM (Quantity)

SUM は、現在の行の数量を、現在の行の前のすべての行の数量に加算します。この関数は、各行の累計を返します。

次の表に、各日付の累計を示します。

Sale_ID	Date	Quantity	Total
30001	2016-08-02	10	10
10001	2016-12-24	10	20
10005	2016-12-24	30	50
40001	2017-01-09	40	90

Sale_ID	Date	Quantity	Total
10006	2017-01-18	10	100
20001	2017-02-12	20	120

集計オフセット

集計関数は、1つのパーティション内の値セットに対して計算を実行します。フレームオフセットがパーティション外部である場合、集計関数はこのフレームを無視します。

フレームのオフセットがパーティションまたはテーブルの内部でない場合、集計関数はパーティション内を計算します。この関数は、NULL またはデフォルト値を返しません。

例えば、テーブルを販売者 ID 別、オーダーを数量別のパーティションに分けるとします。開始オフセットを-3に設定し、終了オフセットを 4 に設定します。

次の図に、現在の入力行のパーティションとフレームを示します。

Seller ID	Quantity
1	10
1	10
1	30
2	20
2	30
3	10
3	15
3	20
3	30
4	10
4	40

Current input row →

Frame

-3

4

このフレームには、合計 8 行が含まれますが、計算はパーティション内に留まります。このフレームを使用して AVG 関数を定義した場合、この関数はパーティション内の数量の平均を計算し、18.75 を返します。

ネストされた集計関数

ウィンドウ関数のネストされた集計関数は、パーティションごとに個別に計算を実行します。ネストされた集計関数は、ウィンドウ関数内にある場合と、アグリゲータトランスフォーメーションにある場合とは異なる動作をします。

アグリゲータトランスフォーメーション内にあるネストされた集計関数は、すべての行でグローバルに計算を実行します。式トランスフォーメーション内にあるネストされた集計関数は、パーティションごとに個別に計算を実行します。

例として、アグリゲータトランスフォーメーションと式トランスフォーメーションに、次のネストされた集計関数を設定します。

```
MEDIAN ( COUNT ( P1 ) )
```

P2 を、アグリゲータトランスフォーメーションのポートごとのグループとして定義します。

関数は、次のデータセットの数のメジアンを取得します。

P1	P2
10	1
7	1
12	1
11	2
13	2
8	2
10	2

RETURN VALUE: 3.5

ネストされた集計関数を式トランスフォーメーションに含めて、そのトランスフォーメーションをウィンドウ化用に設定した場合、関数は、計算をパーティションごとに個別に実行します。

データを P2 でパーティション化し、前後すべての行のフレームを指定します。ウィンドウ関数は、次の計算を実行します。

1. COUNT (P1)により、各行に 1 つの値を生成します。COUNT は、パーティションの行のうち、NULL 以外の値を持つ行の数を返します。
2. その値の MEDIAN は、COUNT によって生成された値のウィンドウのメジアンを生成します。

ウィンドウ関数は、次の出力を生成します。

P1	P2	Output
10	1	3
7	1	3
12	1	3
11	2	4
13	2	4
8	2	4
10	2	4

複数のウィンドウ関数を使用して、集計関数をネストできます。以下に例を示します。

LAG (LEAD(MAX(FIRST (p1)))

注: ウィンドウ関数 LEAD と LAG は任意の数だけネストできますが、1 つの集計関数内に複数の集計関数をネストさせることはできません。

ウィンドウ関数のルールとガイドライン

Spark エンジンでウィンドウ関数を使用するときに、特定のガイドラインが該当します。

トランスフォーメーションにウィンドウ関数を定義する場合は、次のルールとガイドラインを考慮します。

- ウィンドウ関数のオフセット引数として、整数の定数を指定します。
- 入力値と同じデータ型のデフォルト引数を指定します。
- 複合データ型を含むデフォルト引数または SYSTIMESTAMP 引数は指定できません。
- LEAD ウィンドウ関数と LAG ウィンドウ関数を使用するには、ウィンドウ化のプロパティでパーティションとオーダーキーを設定する必要があります。
- 式トランスフォーメーションのウィンドウ関数として集計関数を使用するには、ウィンドウ化のプロパティでフレーム仕様を設定する必要があります。

ウィンドウ化例

このセクションの例は、LEAD、LAG およびその他の集計関数を、式トランスフォーメーションでウィンドウ関数として使用する方法を示します。

財務プランの例

あなたは自分の担当顧客 2 社の財務プランに関する情報を持つ銀行家です。各プランに、開始日が関連付けられています。

各顧客のために、次のプランの有効化日付に基づいて、現在のプランの有効期限を知りたいとします。前のプランは新しいプランの開始時に終了するため、前のプランの終了日は、次のプランの開始日マイナス 1 日です。

次の表に、顧客コード、関連プランコード、各プランの開始日を示します。

CustomerCode	PlanCode	StartDate
C1	00001	2014-10-01
C2	00002	2014-10-01
C2	00002	2014-11-01
C1	00004	2014-10-25
C1	00001	2014-09-01
C1	00003	2014-10-10

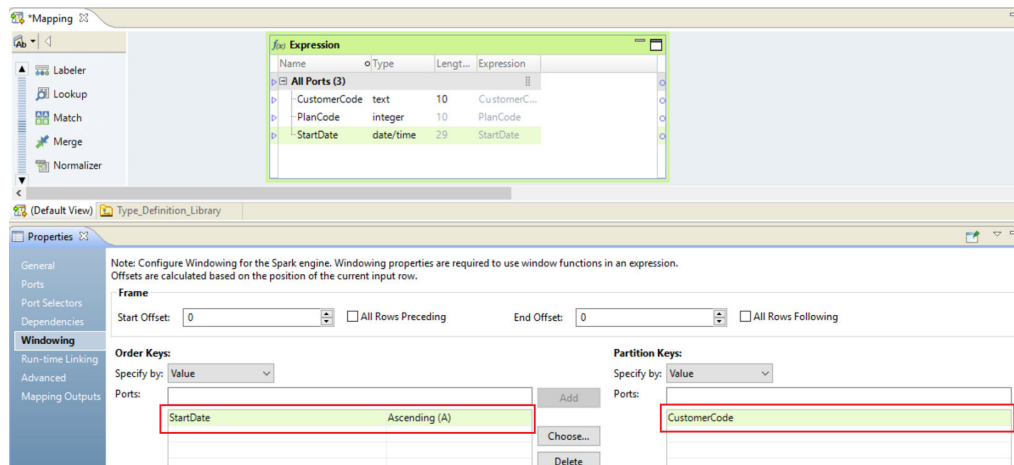
パーティションキーおよびオーダーキーの定義

データを顧客コードでパーティション化し、昇順の開始日で順序付けます。

次のウィンドウ化プロパティを構成します。

プロパティ	説明
オーダーキー	StartDate 昇順。開始日の昇順で、データを時系列に配置します。
パーティションキー	CustomerCode。行を顧客コード別にグループ化し、計算が個々の顧客に基づくようにします。
フレーム	指定されていません。ウィンドウ関数は、オフセット引数に基づいて行にアクセスし、フレーム仕様を無視します。

次の図に、[ウィンドウイング] タブで設定するウィンドウ化プロパティを示します。



次の表に、顧客コードでグループ化された開始日順のデータを示します。

CustomerCode	PlanCode	StartDate
C1	00001	2014-09-01
C1	00002	2014-10-01
C1	00003	2014-10-10
C1	00004	2014-10-25
C2	00001	2014-10-01
C2	00002	2014-11-01

各顧客の開始日は昇順で配置されているため、日付は時系列順です。

ウィンドウ関数の定義

入力ごとに次の行にアクセスする LEAD 関数を定義します。

式トランスフォーメーションの [ポート] タブで、次の関数を定義します。

LEAD (StartDate, 1, '01-Jan-2100')

説明:

- StartDate は関数の処理対象のターゲットカラムを示します。
- 1 がオフセットです。この値は、直後の行にアクセスします。
- 01-Jan-2100 がデフォルト値です。この式は、戻り値がパーティションの境界外である場合、「01-Jan-2100」を返します。

ADD_TO_DATE 関数の定義

アクセスした日付から 1 日を除算する ADD_TO_DATE 関数を使用します。

式トランスフォーメーションの [ポート] タブで、次の式を定義します。

```
ADD_TO_DATE ( LEAD ( StartDate, 1, '01-Jan-2100' ), 'DD', -1, )
```

次のプランの開始日から 1 日を除算することで、現在のプランの終了日がわかります。

次の表に、各プランの終了日を示します。

CustomerCode	PlanCode	StartDate	EndDate
C1	00001	2014-09-01	2014-09-30
C1	00002	2014-10-01	2014-10-09
C1	00003	2014-10-10	2014-10-24
C1	00004	2014-10-25	2099-12-31*
C2	00001	2014-10-01	2014-10-31
C2	00002	2014-11-01	2099-12-31*

*LEAD 関数は、これらのプランがまだ終了していないために、デフォルト値を返しました。これらの行はパーティション外であるため、ADD_TO_DATE 関数は 01-Jan-2100 から 1 日を除算し、2099-12-31 を返しました。

GPS ping の例

あなたの組織はトリップ ID、イベント ID、およびタイムスタンプを含む車両からの GPS ping を受信します。各 ping 間の時間差を計算し、前の行との時間差が 60 秒未満である場合、行にスキップとのフラグを付けようとしています。

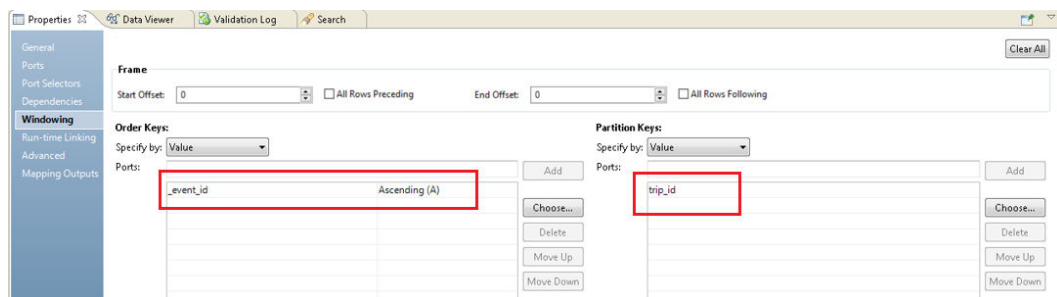
イベントを時系列で順序付け、イベントをトリップ別にパーティション化します。前の行のイベント時刻にアクセスするウィンドウ関数を定義し、ADD_TO_DATE 関数を使用して、2 つのイベント間の時間差を計算します。

ウィンドウ化プロパティ

[ウィンドウイング] タブで、次のウィンドウイングプロパティを定義します。

プロパティ	説明
オーダーキー	_event_id 昇順。イベント ID の昇順で、データを時系列に配置します。
パーティションキー	trip_id。行をトリップ ID 別にグループ化し、計算が同じトリップからのイベントに基づくようにします。
フレーム	指定されていません。ウィンドウ関数は、オフセット引数に基づいて行にアクセスし、フレーム仕様を無視します。

次の図に、式トランスフォーメーションで設定するウィンドウ化プロパティを示します。



ウィンドウ関数

前の行からイベント時刻を取得する次の LAG 関数を定義します。

LAG (_event_time, 1, NULL)

説明:

- _event_time は、前の行からの返される値を含むカラム名です。
- 1 がオフセットです。この値は、現在の行の直前の行を表します。
- NULL がデフォルト値です。この関数は、戻り値がパーティションの境界外である場合 NULL を返します。

2 つの日付間の時間を計算する次の DATE_DIFF 関数を定義します。

DATE_DIFF (_event_time, LAG (_event_time, 1, NULL), 'ss')

DATE_DIFF が 60 秒未満の場合、または _event_time が NULL の場合、行にスキップとのフラグを付けます。

IIF (DATE_DIFF < 60 or ISNULL (_event_time), 'Skip', 'Valid')

次の図に、トランスフォーメーションプロパティで定義する式を示します。

	Name	oType	Configuration	Precisi...	Scale	Input	Output	Variable	Expression
1	trip_id	integer	N/A	10	0	✓	✓		
2	_event_id	integer	N/A	10	0	✓	✓		
3	_event_time	date/time	N/A	29	9	✓	✓		
4	date_diff	integer	N/A	10	0		✓		if(lag(trip_id,1,0)=0,999,DATE_DIFF(_event_time,lag(_event_time,1,null),'ss'))
5	o_datediff	integer	N/A	10	0		✓		date_diff
6	flag	string	N/A	10	0		✓		if(date_diff<60 or isnull(_event_time),'Skip','Valid')

アウトプット

トランスフォーメーションは、次の出力を生成します。

Trip ID	Event ID	Event Time	Time Difference	Flag
101	1	2017-05-03 12:00:00	NULL*	Skip
101	2	2017-05-03 12:00:34	34	Skip
101	3	2017-05-03 12:02:00	86	Valid
101	4	2017-05-03 12:02:23	23	Skip
102	1	2017-05-03 12:00:00	NULL*	Skip
102	2	2017-05-03 12:01:56	116	Valid
102	3	2017-05-03 12:02:00	4	Skip
102	4	2017-05-03 13:00:00	3480	Valid
103	1	2017-05-03 12:00:00	NULL*	Skip
103	2	2017-05-03 12:00:12	12	Skip
103	3	2017-05-03 12:01:12	60	Valid

*これらの行の前の行は、パーティションの境界外であるため、トランスフォーメーションは NULL 値を生成しました。

ウィンドウ関数としての集計関数例

人事グループのための業務で、各従業員の給与をその従業員の部門の平均給与と比較しようとしています。

次の表に、部門名、従業員 ID 番号、従業員の給与の一覧を示します。

Department	Employee	Salary
Development	11	5200
Development	7	4200
Development	9	4500
Development	8	6000
Development	10	5200
Personnel	5	3500
Personnel	2	3900
Sales	3	4800
Sales	1	5000

Department	Employee	Salary
Sales	4	4800

すべての従業員を計算に含めるための、バインドなしフレームを設定し、各従業員の給与とその従業員の部門内の平均給与との間の差異を計算する集計関数を定義します。

ウィンドウ化プロパティ

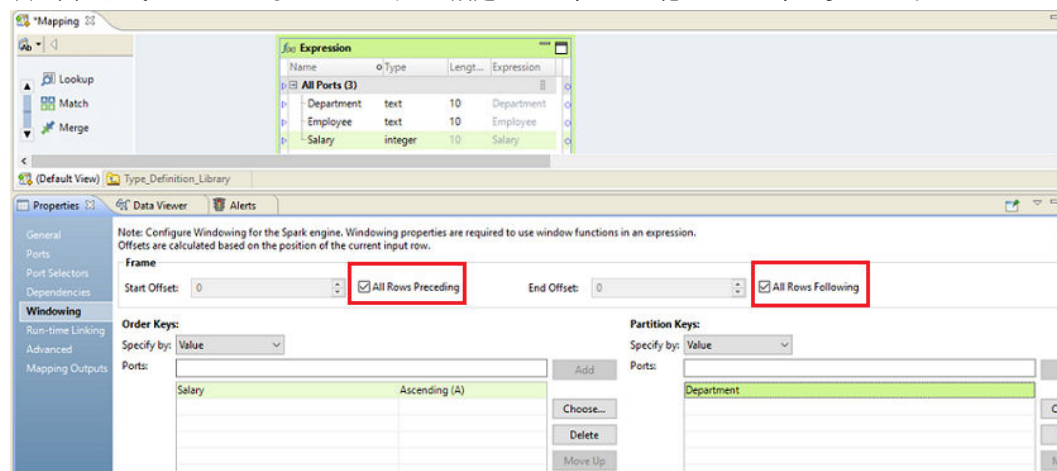
[ウィンドウイング] タブで、次のウィンドウイングプロパティを定義します。

プロパティ	説明
オーダーキー	昇順の給与。データを給与の昇順に配置します。
パーティションキー	部門。行を部門別にグループ化します。
開始オフセット	前のすべての行
終了オフセット	後のすべての行

バインドなしフレームを使用すると、集計関数に計算内のすべてのパーティション行が含まれます。

例えば、現在の行が3行目とします。3行目は「Development」パーティション内にあるため、このフレームには、「Development」パーティション内の3行目前後のすべての行に加えて、3行目が含まれます。

次の図に、式トランスフォーメーションで設定するウィンドウ化プロパティを示します。



ウィンドウ関数

ウィンドウ化用のトランスフォーメーションを設定すると、集計関数はウィンドウ関数として動作します。

各従業員の給与とその従業員の部門内の平均給与との間の差異を計算する集計関数を定義します。

$\text{Salary} - \text{AVG}(\text{Salary}) = \text{Salary\_Diff}$

アウトプット

トランスフォーメーションは、次の給与の差異を生成します。

Department	Employee	Salary	Salary_Diff
Development	11	5200	-820
Development	7	4200	-520
Development	9	4500	180
Development	8	6000	180
Development	10	5200	980
Personnel	5	3500	200
Personnel	2	3900	200
Sales	3	4800	-66
Sales	1	5000	-66
Sales	4	4800	134

どの従業員が自部門の平均給与よりも少なく、または多く稼いでいるかを特定できます。この情報に基づき、その他のトランスフォーメーションを追加し、データをより詳しく調べることができます。例えば、ランクトランスフォーメーションを追加し、各従業員の自部門内での数値的なランクを生成できます。

付録 A

接続

この付録では、以下の項目について説明します。

- [接続, 233 ページ](#)
- [クラウドプロビジョニング設定, 233 ページ](#)
- [Amazon Redshift 接続のプロパティ, 239 ページ](#)
- [Amazon S3 接続のプロパティ, 241 ページ](#)
- [, 242 ページ](#)
- [, 244 ページ](#)
- [Google Analytics 接続のプロパティ, 245 ページ](#)
- [Google BigQuery 接続のプロパティ, 246 ページ](#)
- [Google Cloud Spanner 接続のプロパティ, 247 ページ](#)
- [Google Cloud Storage 接続のプロパティ, 248 ページ](#)
- [Hadoop 接続プロパティ, 249 ページ](#)
- [HDFS 接続プロパティ, 254 ページ](#)
- [HBase 接続プロパティ, 256 ページ](#)
- [MapR-DB の HBase 接続のプロパティ, 256 ページ](#)
- [Hive 接続のプロパティ, 257 ページ](#)
- [JDBC 接続のプロパティ, 260 ページ](#)
- [Kafka 接続のプロパティ, 265 ページ](#)
- [Microsoft Azure Blob ストレージ接続のプロパティ, 266 ページ](#)
- [Microsoft Azure Cosmos DB SQL API 接続のプロパティ, 267 ページ](#)
- [Microsoft Azure Data Lake Store 接続プロパティ, 267 ページ](#)
- [Microsoft Azure SQL Data Warehouse 接続プロパティ, 268 ページ](#)
- [Snowflake 接続プロパティ, 269 ページ](#)
- [ソースまたはターゲットにアクセスするための接続の作成, 271 ページ](#)
- [Hadoop 接続の作成, 271 ページ](#)
- [Hadoop 接続プロパティの設定, 272 ページ](#)

接続

非ネイティブ環境、Hadoop、Databricks にアクセスするための接続を作成します。HBase、HDFS、あるいは Hadoop 環境の Hive ソースまたはターゲットにアクセスする場合は、これらの接続も作成する必要があります。Developer tool、Administrator ツール、および infacmd を使用すると、接続を作成できます。

以下のタイプの接続を作成できます。

Hadoop 接続

Hadoop 環境でマッピングを実行できるように Hadoop 接続を作成します。

HBase 接続

HBase 接続を作成して HBase にアクセスします。HBase 接続は NoSQL 接続です。

HDFS 接続

Hadoop クラスタの HDFS ファイルシステムとの間でデータを読み込み、書き込むための HDFS 接続を作成します。

Hive 接続

ソースまたはターゲットとして Hive に接続できるように Hive 接続を作成します。ネイティブ環境または Hadoop 環境でマッピングを有効にすると、ソースとして Hive にアクセスできます。Blaze エンジンでマッピングを実行すると、ターゲットとして Hive にアクセスできます。

JDBC 接続

JDBC 接続を作成し、Sqoop を介してリレーショナルデータのインポートとエクスポートができるようにこの接続で Sqoop プロパティを設定します。

Databricks 接続

Databricks 環境でマッピングを実行できるように Databricks 接続を作成します。

注: ソーシャルメディア web サイトや Teradata など、他のソースまたはターゲットへの接続の作成に関する詳細については、対応する PowerExchange アダプタユーザーガイドを参照してください。

クラウドプロビジョニング設定

クラウドプロビジョニング設定では、ワークフローがマッピングタスクを実行するために使用するクラスタの作成タスクとクラスタ接続間のリレーションを確立します。クラスタの作成タスクには、クラウドプロビジョニング設定への参照を含める必要があります。代わりに、クラウドプロビジョニング設定は、クラスタワークフローで使用するために作成するクラスタ接続にポイントします。

入力するプロパティは、クラスタの作成で選択する Hadoop ディストリビューションによって異なります。次のいずれかの接続タイプを選択します。

- AWS クラウドプロビジョニング。Amazon Web サービスで Amazon EMR クラスタに接続します。
- Azure クラウドプロビジョニング。Azure プラットフォームの HDInsight クラスタに接続します。
- Databricks クラウドプロビジョニング Azure Databricks プラットフォームで Databricks クラスタに接続します。

AWS クラウドプロビジョニング設定のプロパティ

AWS クラウドプロビジョニング設定のプロパティでは、データ統合サービスが AWS クラウドプラットフォームのリソースに接続および作成できます。

全般プロパティ

次の表では、クラウドプロビジョニング設定の全般プロパティについて説明します。

プロパティ	説明
名前	クラウドプロビジョニング設定の名前。
ID	クラウドプロビジョニング設定の ID。デフォルト：クラウドプロビジョニング設定名と同じ。
説明。	オプション。クラウドプロビジョニング設定の説明。
AWS アクセスキー ID	オプション。AWS アクセスキーの ID。AWS はこの ID を使用して、AWS サービス API に対する REST または HTTP クエリプロトコル要求を制御します。 値を指定しない場合、Informatica はデフォルトクレデンシャルプロバイダチェーンをフォローしようとします。
AWS シークレットアクセスキー	AWS アクセスキーの秘密のコンポーネント。 AWS アクセスキー ID を指定する場合に必要です。
領域	クラスタを作成する領域。VPC が実行中の領域を指定する必要があります。g garden AWS の領域値を使用します。許容可能な値のリストについては、AWS のドキュメントを参照してください。 注: クラスタを作成可能な領域は、Informatica ドメインがインストールされた領域と分けることができます。

権限

次の表では、クラウドプロビジョニング設定の権限プロパティについて説明します。

プロパティ	説明
EMR ロール	作成する EMR クラスタのサービスロールの名前。クラスタの作成、S3 リソースのアクセス、およびクラスタ上のジョブの実行には、ロールに十分な権限を設定する必要があります。 AWS 管理者がこのロールを作成する場合は、「EMR」ロールを選択します。これには、デフォルトの AmazonElasticMapReduceRole ポリシーが含まれます。このポリシー内のサービスを編集することができます。
EC2 インスタンスプロファイル	クラスタで実行するプロセスの権限を制御する EC2 インスタンスプロファイルロールの名前。 AWS 管理者がこのロールを作成する場合は、「EC2 の EMR ロール」ロールを選択します。デフォルトでは、S3 アクセスが含まれます。
ロールの自動スケーリング	EMR クラスタの自動スケーリングを設定する場合に必要です。 このロールは、AWS 管理者が VPC のクラスタで自動スケーリングを設定する場合に作成されます。 デフォルト：このフィールドを空白のままにした場合、AWS 管理者が AWS コンソールでクラスタを作成した際に自動スケーリングロールを「ロールなしで続行」に設定するのと同じです。

EC2 設定

次の表では、クラウドプロビジョニング設定の EC2 設定プロパティについて説明します。

プロパティ	説明
EC2 キーペア	EMR クラスタマスタノードとの通信を有効にする EC2 キーペア。 オプション。このクレデンシャルにより、クラスタにログインできます。クラスタを永続的にするには、このプロパティを設定します。
EC2 サブネット	クラスタを作成するための VPC のサブネットの ID。 クラスタが実行される EC2 インスタンスのサブネット ID を使用します。
マスタセキュリティグループ	オプション。クラスタマスタノードのセキュリティグループの ID。クラスタノードへの送受信トラフィックを制御する仮想ファイアウォールとして機能します。 AWS 管理者が VPC のクラスタを作成および設定する場合、セキュリティグループが作成されます。AWS コンソールでは、このプロパティは ElasticMapReduce-master に相当します。 既存のセキュリティグループを使用するか、AWS 管理者が一時クラスタ専用のセキュリティグループを作成できます。 値を指定しない場合、クラスタが VPC にデフォルトのセキュリティグループを適用します。
追加のマスタセキュリティグループ	オプション。クラスタマスタノードに追加する追加セキュリティグループの ID。セキュリティグループ ID のカンマ区切りのリストを使用します。

プロパティ	説明
コアおよびタスクセキュリティグループ	オプション。クラスタコアとタスクノードのセキュリティグループの ID。AWS 管理者が AWS コンソールのクラスタを作成および設定する場合、プロパティは ElasticMapReduce-slave セキュリティグループに相当します。 値を指定しない場合、クラスタが VPC にデフォルトのセキュリティグループを適用します。
追加コアおよびタスクセキュリティグループ	オプション。クラスタコアおよびタスクノードに追加する追加セキュリティグループの ID。セキュリティグループ ID のカンマ区切りのリストを使用します。
サービスアクセスセキュリティグループ	サービスアクセス用の EMR 管理セキュリティグループ。プライベートサブネットで EMR クラスタをプロビジョニングする場合に必要です。

Azure クラウドプロビジョニング設定のプロパティ

Azure クラウドプロビジョニング設定のプロパティでは、データ統合サービスが Azure クラウドプラットフォームのリソースに接続したり、リソースを作成したりできます。

認証の詳細

次の表に、設定する認証プロパティを示します。

プロパティ	説明
名前	クラウドプロビジョニング設定の名前。
ID	クラウドプロビジョニング設定の ID。デフォルト：クラウドプロビジョニング設定名と同じ。
説明	オプション。クラウドプロビジョニング設定の説明。
サブスクリプション ID	クラスタ作成プロセスで使用する Azure アカウントの ID。
テナント ID	Azure Active Directory と関連付けられている GUID 文字列。
クライアント ID	サービスプリンシパルと関連付けられているアプリケーション ID と同じ GUID 文字列。サービスプリンシパルには、サブスクリプション ID プロパティで特定したサブスクリプションでリソースを作成できる権限を持つロールを割り当てる必要があります。
クライアントシークレット	クライアント ID に関連付けられているキーを提供するオクテット文字列。

ストレージアカウントの詳細

次のいずれかのストレージタイプへのアクセスを設定します。

- Azure Data Lake Storage (ADLS) [Azure documentation](#) を参照してください。
- Azure ストレージアカウント。general または blob ストレージとも呼ばれます。 [Azure documentation](#) を参照してください。

次の表に、HDInsight クラスタを使用して Azure Data Lake Storage (ADLS) を設定する際に必要な情報を示します。

プロパティ	説明
Azure Data Lake ストア名	アクセスする ADLS ストレージの名前。作成する ADLS ストレージとクラスタは同じ領域に格納する必要があります。
Data Lake サービスプリンシパルのクライアント ID	ADLS ストレージにプログラムでアクセスできるクレデンシャル。Informatica ドメインが ADLS と通信できるようになり、HDInsight クラスタでコマンドとマッピングを実行します。 サービスプリンシパルは、次の要件を満たす Azure ユーザーです。 - ADLS ストレージに必要なディレクトリにアクセスできる権限。 - ADLS ストレージの証明書ベースの認証。 - ADLS ストレージのキーベースの認証。
Data Lake サービスプリンシパルの証明書の内容	サービスプリンシパルで使用する公開証明書の Base64 でエンコードされたテキスト。 クラウドプロビジョニング設定を作成する場合は、このプロパティを空白のままにします。クラウドプロビジョニング設定を保存したら、Informatica ドメインがインストールされている VM にログインし、 <code>infacmd ccps updateADLSCertificate</code> を実行してこのプロパティを入力します。
Data Lake サービスプリンシパルの証明書のパスワード	サービスプリンシパルのプライベートキー。このプライベートキーをサービスプリンシパル証明書と関連付ける必要があります。
Data Lake サービスプリンシパルのクライアントシークレット	サービスプリンシパルに関連付けられているキーを提供するオクテット文字列。
Data Lake サービスプリンシパルの OAUTH トークンエンドポイント	OAUTH トークンベースの認証のエンドポイント。

次の表に、HDInsight クラスタを使用して、blob ストレージとも呼ばれる Azure General Storage を設定する際に必要な情報を示します。

プロパティ	説明
Azure ストレージアカウント名	アクセスするストレージアカウントの名前。Azure web コンソールのストレージアカウントノードから値を取得します。作成するストレージとクラスタは同じ領域に格納する必要があります。
Azure ストレージアカウントキー	ストレージアカウントへのアクセスを認証するキー。Azure web コンソールから値を取得するには、ストレージアカウント、アクセスキーを選択します。コンソールにはアカウントキーが表示されます。

クラスタデプロイメントの詳細

以下の表に、設定するクラスタデプロイメントプロパティを示します。

プロパティ	説明
リソースグループ	クラスタを作成するリソースグループ。リソースグループは、Azure リソースの論理セットです。
仮想ネットワークのリソースグループ	オプション。仮想ネットワークが属するリソースグループ。 リソースグループを指定しない場合、データ統合サービスは、仮想ネットワークがクラスタと同じリソースグループのメンバーであると仮定します。
仮想ネットワーク	クラスタを作成する仮想ネットワークまたは <code>vnet</code> の名前。[仮想ネットワークのリソースグループ] プロパティで指定したリソースグループに配置されている <code>vnet</code> を指定します。 <code>vnet</code> は、クラスタを作成する領域と同じ領域内に配置する必要があります。
サブネット名	クラスタを作成するサブネット。サブネットは、以前のプロパティで指定した <code>vnet</code> に含まれる必要があります。 各 <code>vnet</code> に 1 つ以上のサブネットを指定できます。Azure 管理者は既存のサブネットを選択するか、クラスタ用に 1 個のサブネットを作成できます。

外部 Hive メタストアの詳細

このプロパティを指定すると、クラスタがクラスタの外部にある Hive メタストアデータベースに接続できるようになります。

MySQL または Amazon RDS のような外部リレーショナルデータベースを Hive メタストアデータベースとして使用できます。外部データベースは、作成するクラスタと同じクラウドプラットフォーム上に配置する必要があります。

このダイアログボックスで既存の外部データベースを指定していない場合、クラスタは自身の独自データベースをクラスタで作成します。クラスタが終了すると、このデータベースは終了します。

以下の表に、設定する Hive メタストアデータベースのプロパティを示します。

プロパティ	説明
データベース名	Hive メタストアデータベースの名前。
データベースサーバー名	データベースが格納されているサーバー。 注: Azure web コンソールのデータベースサーバー名には、サフィックス <code>database.windows.net</code> が共通して含まれます。例えば、 <code>server123xyz.database.windows.net</code> です。サフィックスが設定されていないデータベースサーバー名を指定できます。 Informatica はサフィックスを自動的に追加します。例えば、 <code>server123xyz</code> を指定できます。
データベースユーザー名	データベースへのアクセスに使用できるドメインのアカウントのユーザー名。
データベースパスワード	ユーザーアカウントのパスワードです。

Databricks クラウドプロビジョニング設定のプロパティ

Databricks クラウドプロビジョニング設定のプロパティでは、データ統合サービスが Databricks クラウドプラットフォームのリソースに接続したり、リソースを作成したりできます。

次の表は、Databricks クラウドプロビジョニング設定のプロパティを示しています。

プロパティ	説明
名前	クラウドプロビジョニング設定の名前。
ID	Databricks クラスタのクラスタ ID。
説明	クラウドプロビジョニング設定の説明（省略可能）。
Databricks Domain	Databricks デプロイメントのドメイン名。
Databricks トークン ID	Databricks 内で作成される、認証に必要なトークン ID。 注: トークンに有効期限がある場合は、トークンの有効期限が切れる前に、Databricks 管理者から新しいトークンを取得していることを確認してください。

Amazon Redshift 接続のプロパティ

Amazon Redshift 接続をセットアップする際には、接続プロパティを設定する必要があります。

次の表に、Amazon Redshift 接続プロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および次の特殊文字は使用できません。~`!\$%^&*()-+={[}] \:;'"<, > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。
説明	接続の説明。説明は、4,000 文字を超えることはできません。
場所	接続を作成するドメイン。
タイプ	接続タイプ。データベースで Amazon Redshift を選択します。

[詳細] タブには、Amazon Redshift 接続の接続属性が含まれています。以下の表に、接続属性を示します。

プロパティ	説明
ユーザー名	Amazon Redshift アカウントのユーザー名。
パスワード	Amazon Redshift アカウントのパスワード。

プロパティ	説明
スキーマ	オプション。Amazon Redshift スキーマ名。複数のスキーマを使用する場合は、スキーマ名を指定しないでください。データオブジェクトウィザードには、Amazon Redshift オブジェクトで利用できるすべてのユーザー定義スキーマが表示されます。 デフォルトは public です。
AWS アクセスキー ID	Amazon S3 バケットアクセスキー ID。 注: AWS Identity and Access Management (IAM) 認証を使用しない場合は必須です。
AWS シークレットアクセスキー	Amazon S3 バケットシークレットアクセスキー ID。 注: AWS Identity and Access Management (IAM) 認証を使用しない場合は必須です。
マスタ対称キー	オプション。クライアントサイド暗号化を有効にする場合に、256 ビットの AES 暗号化キーを Base64 形式で指定します。暗号化キーは、サードパーティ製ツールを使用して生成できます。 この値を指定する場合は、詳細ターゲットプロパティで、暗号化タイプとしてクライアントサイド暗号化を指定してください。
顧客マスタキー ID	オプション。AWS Key Management Service (AWS KMS) によって生成された顧客マスタキー ID またはエイリアス名を指定します。Amazon S3 バケットが存在するリージョンに対応する顧客マスタキーを生成する必要があります。次のいずれかの値を指定できます。 顧客が生成した顧客マスタキー クライアントサイドまたはサーバーサイドの暗号化を有効にします。 デフォルトの顧客マスタキー クライアントサイドまたはサーバーサイドの暗号化を有効にします。アカウントの管理者ユーザーのみがデフォルトの顧客マスタキー ID を使用してクライアントサイド暗号化を有効にできます。 注: 顧客マスタキー ID は、ネイティブ環境または Spark エンジンでマッピングを実行するときに使用できます。
クラスタノードタイプ	Amazon Redshift クラスタのノードタイプ。 次のオプションを選択することができます。 - ds1.xlarge - ds1.8xlarge - dc1.large - dc1.8xlarge - ds2.xlarge - ds2.8xlarge クラスタを構成するノードの詳細については、Amazon Redshift のドキュメントを参照してください。
クラスタ内ノード数	Amazon Redshift クラスタのノード数。 クラスタを構成するノードの詳細については、Amazon Redshift のドキュメントを参照してください。
JDBC URL	Amazon Redshift 接続 URL。

注: そうしないと、現在のバージョンでマッピングを実行すると、マッピングが失敗します。

Amazon S3 接続のプロパティ

Amazon S3 接続をセットアップする際には、接続プロパティを設定する必要があります。

次の表に、Amazon S3 接続のプロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および次の特殊文字は使用できません。~`!\$%^&*()-+={[]] \\:;'"<, > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。
説明	オプション。接続の説明。説明は、4,000 文字を超えることはできません。
場所	接続を作成するドメイン。
タイプ	Amazon S3 接続タイプ。
アクセスキー	Amazon アカウントリソースにアクセスするためのアクセスキー ID。 注: AWS Identity and Access Management (IAM) 認証を使用しない場合は必須です。
秘密鍵	Amazon アカウントリソースにアクセスするための秘密アクセス鍵。秘密鍵はアクセスキーに関連付けられており、アカウントを一意に識別します。 注: AWS Identity and Access Management (IAM) 認証を使用しない場合は必須です。
フォルダパス	Amazon S3 オブジェクトへの完全なパス。パスには、バケット名と任意のフォルダ名を含める必要があります。 フォルダパスの末尾にスラッシュを使用しないでください。例: <バケット名>/<フォルダ名>。
マスタ対称キー	オプション。クライアントサイド暗号化を有効にする場合に、256 ビットの AES 暗号化キーを Base64 形式で指定します。

プロパティ	説明
顧客マスター ID	オプション。AWS Key Management Service (AWS KMS) によって生成された顧客マスターキー ID またはエイリアス名を指定します。Amazon S3 バケットが存在するリージョンの顧客マスターキーを生成する必要があります。 次のいずれかの値を指定できます。 顧客が生成した顧客マスターキー クライアントサイドまたはサーバーサイドの暗号化を有効にします。 デフォルトの顧客マスターキー クライアントサイドまたはサーバーサイドの暗号化を有効にします。アカウントの管理者ユーザーのみがデフォルトの顧客マスターキー ID を使用してクライアントサイド暗号化を有効にできます。 注: ネイティブ環境または Spark エンジンでマッピングを実行する場合に適用されます。
リージョン名	アクセスするバケットが存在する AWS リージョンを選択します。 次のいずれかのリージョンを選択します。 - アジアパシフィック (ムンバイ) - アジアパシフィック (ソウル) - アジアパシフィック (シンガポール) - アジアパシフィック (シドニー) - アジアパシフィック (東京) - AWS GovCloud (米国) - カナダ (中部) - 中国 (北京) - - 欧州 (アイルランド) - 欧州 (フランクフルト) - EU (ロンドン) - EU (パリ) - 南米 (サンパウロ) - 米国東部 (オハイオ) - 米国東部 (バージニア北部) - 米国西部 (北カリフォルニア) - 米国西部 (オレゴン) デフォルトは [米国東部 (バージニア北部)] です。

Box 接続を設定するときは、接続プロパティを設定する必要があります。

注: 接続プロパティの順序は、表示するツールによって異なる場合があります。

次の表に、Tableau 接続のプロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および以下の特殊文字は使用できません。 ~ ` ! \$ % ^ & * () - + = { [] } \ : ; " ' < , > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。 ID では大文字と小文字が区別されません。ID は 255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。 デフォルト値は接続名です。
説明	オプション。接続の説明。説明は、4,000 文字を超えることはできません。
場所	接続を作成するドメイン。
タイプ	接続タイプ。
ホスト名	Greenplum サーバーのホスト名または IP アドレス。
ポート	デフォルトは 9042 です。
ユーザ名	サーバーにアクセスするためのユーザー名です。
パスワード	
デフォルト キースペ ース	
SQL 識別子 文字	データベースが、SQL クエリで区切り識別子を囲むのに使用する文字のタイプ。使用できる文字は、データベースタイプによって異なります。 データベースで通常識別子が使用される場合、[なし] を選択します。データ統合サービスで SQL クエリを生成する場合は、識別子を区切り文字で囲みません。 データベースで区切り識別子が使用される場合、文字を選択します。データ統合サービスで SQL クエリが生成される際、区切り識別子はこの文字で囲まれます。
追加接続プ ロパティ	以下の形式で、1 つ以上の JDBC 接続パラメータを入力します。 <param1>=<value>&<param2>=<value>&<param3>=<value>.... - - - - - - -
SSL モード	
SSL トラス トストアバ ス	

プロパティ	説明
SSL トラストストアパスワード	
SSL キーストアパス	
SSL キーストアパスワード	

Hadoop 接続は、クラスタ型の接続です。Hadoop 接続は、管理者ツールまたは Developer tool で作成および管理できます。infacmd を使用して Hadoop 接続を作成できます。データ統合サービスと Hadoop クラスタ間の通信を有効にするには、Hadoop 接続のプロパティを設定します。

以下の表に、Hadoop 接続の一般的な接続プロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および次の特殊文字は使用できません。~`!\$%^&*()-+={[]] \\:;'"<, > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。
説明	オプション。接続の説明。説明は、4,000 文字を超えることはできません。
接続タイプ	
クラスタ設定	Databricks 環境に関連付けられているクラスタ設定の名前。 クラウドプロビジョニング設定を設定しない場合に必要です。
クラウドプロビジョニング設定	クラスタ設定を設定しない場合に必要です。

プロパティ	説明
一時領域ディレクトリ	
詳細プロパティ	Hadoop 環境に固有な詳細プロパティのリスト。 下位レベルで値を設定することにより、上位レベルで構成されたプロパティをオーバーライドできます。データ統合サービスは、次の優先順位に基づいてプロパティのオーバーライドを処理します。 1. 2. データ統合サービスのカスタムプロパティ 注: Informatica は、サードパーティが提供するドキュメント、Informatica が提供するドキュメント、または Informatica のグローバルカスタマサポートに問い合わせることなく、これらのプロパティ値を変更することはお勧めしません。プロパティに関する知識がないまま値を変更すると、パフォーマンスの低下やその他の予期しない結果が発生する可能性があります。

詳細プロパティ

[Spark 設定] セクションの **【詳細プロパティ】** で、次のプロパティを設定します。

`infaspark.json.parser.mode`

破損した JSON レコードの処理方法をパーサーに指定します。次のいずれかのモードに値を設定できます。

- DROPMALFORMED。パーサーは破損したすべてのレコードを無視します。デフォルトのモードです。
- PERMISSIVE。パーサーは、破損したレコードの標準ではないフィールドを NULL として受け取ります。
- FAILFAST。パーサーが破損したレコードを検出し、Spark アプリケーションがダウンした場合、パーサーは例外を生成します。

`infaspark.json.parser.multiLine`

JSON ファイルの複数行のレコードをパーサーが読み取れるかどうかを指定します。この値は、true または false に設定できます。デフォルトは false です。Spark バージョン 2.2.x 以上を使用する非ネイティブのディストリビューションにのみ適用します。

`infaspark.flatfile.writer.nullValue`

Databricks Spark エンジンがターゲットに書き込むときに、NULL 値は空の文字列（「」）に変換されます。例: 12, AB,"",23p09udj。

Databricks Spark エンジンでは文字列カラムに空の文字列を書き込むことができますが、空の文字列を非文字列カラムに書き込もうとすると、タイプの不一致でマッピングが失敗します。

Databricks Spark エンジンで空の文字列を NULL 値に戻してターゲットに書き込めるようにするには、Databricks Spark 接続に次の詳細プロパティを設定します。

`infaspark.flatfile.writer.nullValue=true`

Google Analytics 接続のプロパティ

Salesforce Analytics 接続をセットアップする際には、接続プロパティを設定する必要があります。

注: 接続プロパティの順序は、表示するツールによって異なる場合があります。

次の表に、Google Analytics 接続のプロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および次の特殊文字は使用できません。~`!\$%^&*()-+={[] \:;'"<,>./?
ID	データ統合サービスが接続の識別に使用する文字列。 ID では大文字と小文字が区別されません。ID は 255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。 デフォルト値は接続名です。
説明	オプション。接続の説明。説明は、4,000 文字を超えることはできません。
場所	接続を作成するドメイン。
タイプ	接続タイプ。
サービスアカウント ID	サービスアカウントの作成後にダウンロードする JSON ファイル内にある client_email 値を指定します。
サービスアカウントキー	サービスアカウントの作成後にダウンロードする JSON ファイル内にある private_key 値を指定します。
APIVersion	【Core Reporting API v3】を選択します。 注:

Google BigQuery 接続のプロパティ

Salesforce Analytics 接続をセットアップする際には、接続プロパティを設定する必要があります。

注: 接続プロパティの順序は、表示するツールによって異なる場合があります。

次の表に、Google BigQuery 接続のプロパティを示します。

プロパティ	説明
サービスアカウント ID	サービスアカウントの作成後にダウンロードする JSON ファイル内にある client_email 値を指定します。
サービスアカウントキー	サービスアカウントの作成後にダウンロードする JSON ファイル内にある private_key 値を指定します。

プロパティ	説明
接続モード	Google BigQuery との間でのデータの読み書きに使用するモード。 次のいずれかの接続モードを選択します。 - 簡易。レコードデータ型フィールド内の各フィールドを、マッピング内の個別のフィールドとしてフラット化します。 - 混合。レコードデータ型のフィールドを含む Google BigQuery テーブル内のすべての最上位のフィールドを表示します。Google BigQuery コネクタは、最上位のレコードデータ型のフィールドを、マッピング内の文字列データ型の単一のフィールドとして表示します。 - 複合。Google BigQuery テーブル内のすべての列を、マッピング内の文字列データ型の単一のフィールドとして表示します。 デフォルトは「簡易」です。
スキーマ定義のファイルパス	JSON ファイル名は、Google BigQuery テーブル名と同じです。 JSON ファイルは、Google Cloud Storage 内の指定したパスからローカルマシンにダウンロードできます。
プロジェクト ID	サービスアカウントの作成後にダウンロードする JSON ファイル内にある project_id 値を指定します。 同じサービスアカウントを使用して複数のプロジェクトを作成した場合、接続先のデータセットが含まれるプロジェクトの ID を入力します。
ストレージパス	このプロパティは、大量のデータを読み書きするときに適用されます。 バケット名、またはバケット名とフォルダ名のいずれかを入力できます。 例えば、gs://<bucket_name>または gs://<bucket_name>/<folder_name>を入力します。

Google Cloud Spanner 接続のプロパティ

Google Cloud Spanner 接続を作成する際には、接続プロパティを設定する必要があります。

注: 接続プロパティの順序は、表示するツールによって異なる場合があります。

次の表に、Google Cloud Spanner 接続のプロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および次の特殊文字は使用できません。~`!\$%^&*()-+={} \`';"<, > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。 ID では大文字と小文字が区別されません。ID は 255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。 デフォルト値は接続名です。
説明	オプション。接続の説明。説明は、4,000 文字を超えることはできません。
場所	接続を作成するドメイン。
タイプ	接続タイプ。

プロパティ	説明
プロジェクト ID	サービスアカウントの作成後にダウンロードする JSON ファイル内にある <code>project_id</code> 値を指定します。 同じサービスアカウントを使用して複数のプロジェクトを作成した場合、接続先のバケットが含まれるプロジェクトの ID を入力します。
サービスアカウント ID	サービスアカウントの作成後にダウンロードする JSON ファイル内にある <code>client_email</code> 値を指定します。
サービスアカウントキー	サービスアカウントの作成後にダウンロードする JSON ファイル内にある <code>private_key</code> 値を指定します。
インスタンス ID	Google Cloud Spanner 内で作成したインスタンスの名前。

Google Cloud Storage 接続のプロパティ

Google Cloud Storage 接続を作成する際には、接続プロパティを設定する必要があります。

注: 接続プロパティの順序は、表示するツールによって異なる場合があります。

次の表に、Google Cloud Storage 接続のプロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および次の特殊文字は使用できません。~`!\$%^&*()-+={ }\:;'"<, > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。 ID では大文字と小文字が区別されません。ID は 255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。 デフォルト値は接続名です。
説明	オプション。接続の説明。説明は、4,000 文字を超えることはできません。
場所	接続を作成するドメイン。
タイプ	接続タイプ。
プロジェクト ID	サービスアカウントの作成後にダウンロードする JSON ファイル内にある <code>project_id</code> 値を指定します。 同じサービスアカウントを使用して複数のプロジェクトを作成した場合、接続先のバケットが含まれるプロジェクトの ID を入力します。

プロパティ	説明
サービスアカウント ID	サービスアカウントの作成後にダウンロードする JSON ファイル内にある <code>client_email</code> 値を指定します。
サービスアカウントキー	サービスアカウントの作成後にダウンロードする JSON ファイル内にある <code>private_key</code> 値を指定します。

Hadoop 接続プロパティ

Hadoop 接続を使用して、Hadoop クラスタで実行するマッピングを設定します。Hadoop 接続は、クラスタ型の接続です。Hadoop 接続は、管理者ツールまたは Developer tool で作成および管理できます。infacmd を使用して Hadoop 接続を作成できます。Hadoop 接続プロパティは、特に明記されている場合を除き、大文字と小文字が区別されます。

Hadoop クラスタプロパティ

データ統合サービスと Hadoop クラスタ間の通信を有効にするには、Hadoop 接続のプロパティを設定します。

以下の表に、Hadoop 接続の一般的な接続プロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および以下の特殊文字は使用できません。 ~ ` ! \$ % ^ & * () - + = { [] } \ : ; " ' < , > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。
説明	接続の説明。接続を識別するために使用できる文字列を入力します。説明は、4,000 文字を超えることはできません。
クラスタ設定	Hadoop 環境に関連付けられているクラスタ設定の名前。 クラウドプロビジョニング設定を行わない場合に必要です。
クラウドプロビジョニング設定	Amazon AWS や Microsoft Azure などのクラウドプラットフォームに関連付けられているクラウドプロビジョニング設定の名前。 クラスタ設定を行わない場合に必要です。

プロパティ	説明
クラスタ環境変数*	Hadoop クラスタが使用する環境変数。 例えば、変数 ORACLE_HOME は、Oracle データベースクライアントソフトウェアがインストールされているディレクトリを表します。 Hadoop 環境のランタイムプロパティは、データ統合サービス、Hadoop 接続、およびマッピングで構成できます。下位レベルで値を設定することにより、上位レベルで構成されたプロパティをオーバーライドできます。例えば、データ統合サービスのカスタムプロパティでプロパティを構成する場合は、Hadoop 接続またはマッピングでオーバーライドできます。データ統合サービスは、次の優先順位に基づいてプロパティのオーバーライドを処理します。 1. infacmd ms runMapping で -cp オプションを指定して設定された、マッピングのカスタムプロパティ 2. Hadoop 環境のマッピングのランタイムプロパティ 3. ランタイムエンジン用の Hadoop 接続の詳細プロパティ 4. Hadoop 接続の詳細な全般プロパティ、環境変数、およびクラスパス 5. データ統合サービスのカスタムプロパティ
クラスタのライブラリパス*	クラスタ上の共有ライブラリのパス。 \$DEFAULT_CLUSTER_LIBRARY_PATH 変数には、デフォルトディレクトリの一覧が含まれています。
クラスタのクラスパス*	Hadoop jar ファイルおよび必要なライブラリにアクセスするためのクラスパス。 \$DEFAULT_CLUSTER_CLASSPATH 変数には、デフォルトの jar ファイルおよびライブラリへのパスの一覧が含まれています。 Hadoop 環境のランタイムプロパティは、データ統合サービス、Hadoop 接続、およびマッピングで構成できます。下位レベルで値を設定することにより、上位レベルで構成されたプロパティをオーバーライドできます。例えば、データ統合サービスのカスタムプロパティでプロパティを構成する場合は、Hadoop 接続またはマッピングでオーバーライドできます。データ統合サービスは、次の優先順位に基づいてプロパティのオーバーライドを処理します。 1. infacmd ms runMapping で -cp オプションを指定して設定された、マッピングのカスタムプロパティ 2. Hadoop 環境のマッピングのランタイムプロパティ 3. ランタイムエンジン用の Hadoop 接続の詳細プロパティ 4. Hadoop 接続の詳細な全般プロパティ、環境変数、およびクラスパス 5. データ統合サービスのカスタムプロパティ
クラスタの実行可能ファイルパス*	クラスタ上の実行可能ファイルのパス。 \$DEFAULT_CLUSTER_EXEC_PATH 変数には、デフォルトの実行可能ファイルへのパスの一覧が含まれています。
* Informatica は、サードパーティが提供するドキュメント、Informatica が提供するドキュメント、または Informatica のグローバルカスタマサポートに問い合わせることなく、これらのプロパティ値を変更することはお勧めしません。プロパティに関する知識がないまま値を変更すると、パフォーマンスの低下やその他の予期しない結果が発生する可能性があります。	

共通プロパティ

以下の表に、Hadoop 接続に対して設定する共通の接続プロパティを示します。

プロパティ	説明
偽装ユーザー名	Hadoop クラスターで Kerberos 認証を使用する場合に必要です。Hadoop 偽装ユーザー Hadoop 環境でマッピングを実行するためにデータ統合サービスが偽装するユーザー名。 データ統合サービスは、構成されているユーザーに基づいてマッピングを実行します。データ統合サービスがマッピングの実行に使用するユーザーを決定するには、次の順序を参照してください。 1. オペレーティングシステムプロファイルユーザー。プロファイルユーザーが構成されている場合、マッピングはオペレーティングシステムプロファイルユーザーで実行されます。オペレーティングシステムプロファイルユーザーが存在しない場合、マッピングは Hadoop 偽装ユーザーで実行されます。 2. Hadoop 偽装ユーザーこのマッピングは、オペレーティングシステムプロファイルユーザーが構成されていない場合に、Hadoop 偽装ユーザーで実行されます。Hadoop 偽装ユーザーが構成されていない場合、データ統合サービスは、データ統合サービスユーザーでマッピングを実行します。 3. Informatica サービスユーザー。オペレーティングシステムのプロファイルユーザーと Hadoop 偽装ユーザーが設定されていない場合、マッピングの実行には Informatica デモンを起動するオペレーティングユーザーが使用されます。
一時テーブル圧縮コーデック	圧縮コーデッククラス名の Hadoop 圧縮ライブラリ。 注: Spark エンジンは、一時テーブルの圧縮設定をサポートしていません。Spark エンジンでマッピングを実行すると、Spark エンジンは一時テーブルを非圧縮ファイル形式で格納します。
コーデッククラス名	データ圧縮を有効にし、一時ステージングテーブルのパフォーマンスを改善するコーデッククラス名。
Hive ステージングデータベース名	Hive ステージングテーブルの名前空間。指定されたデータベース名を持たないテーブルには、名前として default を使用してください。 名前空間を設定しない場合、データ統合サービスは Hive ターゲット接続の Hive データベース名を使用してステージングテーブルを作成します。 データ統合サービスは、Hadoop 接続で設定された値を使用します。
詳細プロパティ	Hadoop 環境に固有な詳細プロパティのリスト。プロパティは、Blaze エンジン、エンジン、および Hive エンジンに共通です。詳細プロパティには、デフォルトプロパティのリストが含まれます。 Hadoop 環境のランタイムプロパティは、データ統合サービス、Hadoop 接続、およびマッピングで構成できます。下位レベルで値を設定することにより、上位レベルで構成されたプロパティをオーバーライドできます。例えば、データ統合サービスのカスタムプロパティでプロパティを構成する場合は、Hadoop 接続またはマッピングでオーバーライドできます。データ統合サービスは、次の優先順位に基づいてプロパティのオーバーライドを処理します。 1. infacmd ms runMapping で -cp オプションを指定して設定された、マッピングのカスタムプロパティ 2. Hadoop 環境のマッピングのランタイムプロパティ 3. ランタイムエンジン用の Hadoop 接続の詳細プロパティ 4. Hadoop 接続の詳細な全般プロパティ、環境変数、およびクラスパス 5. データ統合サービスのカスタムプロパティ 注: Informatica は、サードパーティが提供するドキュメント、Informatica が提供するドキュメント、または Informatica のグローバルカスタマサポートに問い合わせることなく、これらのプロパティ値を変更することはお勧めしません。プロパティに関する知識がないまま値を変更すると、パフォーマンスの低下やその他の予期しない結果が発生する可能性があります。

拒否ディレクトリのプロパティ

以下の表に、Hadoop 拒否ディレクトリに設定する接続プロパティを示します。

プロパティ	説明
Hadoop への拒否ファイルの書き込み	Blaze エンジンを使用してマッピングを実行する場合は、このチェックボックスを選択して、拒否ファイルを移動する場所を指定します。選択すると、拒否ファイルはプロパティにリストされている HDFS の場所（拒否ファイルディレクトリ）に移動されます。 デフォルトでは、データ統合サービスは、RejectDir システムパラメータに基づいて拒否ファイルを格納します。
拒否ファイルディレクトリ	マッピングを実行するときの HDFS 上の Hadoop マッピングファイルのディレクトリ。

Hive プッシュダウン設定

注: バージョン 10.2.2 から、Informatica では Hive エンジンのサポートを終了しました。

Blaze 設定

以下の表に、Blaze エンジンに対して設定する接続プロパティを示します。

プロパティ	説明
Blaze ステージングディレクトリ	Blaze エンジンが一時ファイルを保存するために使用するディレクトリの HDFS ファイルパス。ディレクトリが存在することを確認してください。YARN ユーザー、Blaze エンジンユーザー、およびマッピング偽装ユーザーには、このディレクトリへの書き込み権限が必要です。 デフォルトは/blaze/workdir です。このプロパティをオフにすると、ステージングファイルは Hadoop ステージングディレクトリ/tmp/blaze_<user name>に書き込まれます。
Blaze ユーザー名	Blaze サービスと Blaze サービスログの所有者。 Hadoop クラスターで Kerberos 認証を使用する場合、既定のユーザーはデータ統合サービスの SPN ユーザーです。Hadoop クラスターで Kerberos 認証を使用せず、Blaze ユーザーが構成されていない場合、既定のユーザーはデータ統合サービスユーザーです。
最小ポート	Blaze エンジンのポート番号範囲の最小値。デフォルトは 12300 です。
最大ポート	Blaze エンジンのポート番号範囲の最大値。デフォルトは 12600 です。
YARN キュー名	クラスタ上の利用可能なリソースを指定する Blaze エンジンが使用する YARN スケジューラのキュー名。
Blaze ジョブ監視アドレス	Blaze ジョブ監視のホスト名およびポート番号。 次の形式を使用します。 <hostname>:<port> ここで - <hostname>は、Blaze ジョブ監視サーバーのホスト名または IP アドレスです。 - <port>は、Blaze ジョブ監視がリモートプロシージャコール (RPC) をリスンするポートです。 例えば、次のように入力します。myhostname:9080

プロパティ	説明
Blaze YARN ノードラベル	Blaze エンジンが実行される Hadoop クラスタ上のノードを決定するノードラベル。ノードラベルを指定しない場合、Blaze エンジンはデフォルトのパーティションのノードで実行されます。 Hadoop クラスタがノードラベルの論理演算子をサポートしている場合は、ノードラベルの一覧を指定できます。ノードラベルを一覧表示するには、演算子&& (AND)、 (OR)、および! (NOT) を使用します。
詳細プロパティ	Blaze エンジンに固有な詳細プロパティのリスト。詳細プロパティには、デフォルトプロパティのリストが含まれます。 Hadoop 環境のランタイムプロパティは、データ統合サービス、Hadoop 接続、およびマッピングで構成できます。下位レベルで値を設定することにより、上位レベルで構成されたプロパティをオーバーライドできます。例えば、データ統合サービスのカスタムプロパティでプロパティを構成する場合は、Hadoop 接続またはマッピングでオーバーライドできます。データ統合サービスは、次の優先順位に基づいてプロパティのオーバーライドを処理します。 1. infacmd ms runMapping で-cp オプションを指定して設定された、マッピングのカスタムプロパティ 2. Hadoop 環境のマッピングのランタイムプロパティ 3. ランタイムエンジン用の Hadoop 接続の詳細プロパティ 4. Hadoop 接続の詳細な全般プロパティ、環境変数、およびクラスパス 5. データ統合サービスのカスタムプロパティ 注: Informatica は、サードパーティが提供するドキュメント、Informatica が提供するドキュメント、または Informatica のグローバルカスタマサポートに問い合わせることなく、これらのプロパティ値を変更することはお勧めしません。プロパティに関する知識がないまま値を変更すると、パフォーマンスの低下やその他の予期しない結果が発生する可能性があります。

Spark 設定

以下の表に、Spark エンジンに対して設定する接続プロパティを示します。

プロパティ	説明
Spark ステージングディレクトリ	Spark エンジンがジョブ実行用の一時ファイルを保存するために使用するディレクトリの HDFS ファイルパス。YARN ユーザー、データ統合サービスユーザー、およびマッピング偽装ユーザーがこのディレクトリに対する書き込みアクセス許可を持っている必要があります。
Spark イベントログディレクトリ	オプション。Spark エンジンがイベントをログ記録するために使用するディレクトリの HDFS ファイルパス。

プロパティ	説明
YARN キュー名	クラスタで使用可能なリソースを指定する Spark エンジンによって使用される YARN スケジューラキュー名。この名前は、大文字と小文字が区別されます。
詳細プロパティ	Spark エンジンに固有な詳細プロパティのリスト。詳細プロパティには、デフォルトプロパティのリストが含まれます。 Hadoop 環境のランタイムプロパティは、データ統合サービス、Hadoop 接続、およびマッピングで構成できます。下位レベルで値を設定することにより、上位レベルで構成されたプロパティをオーバーライドできます。例えば、データ統合サービスのカスタムプロパティでプロパティを構成する場合は、Hadoop 接続またはマッピングでオーバーライドできます。データ統合サービスは、次の優先順位に基づいてプロパティのオーバーライドを処理します。 1. infacmd ms runMapping で-cp オプションを指定して設定された、マッピングのカスタムプロパティ 2. Hadoop 環境のマッピングのランタイムプロパティ 3. ランタイムエンジン用の Hadoop 接続の詳細プロパティ 4. Hadoop 接続の詳細な全般プロパティ、環境変数、およびクラスパス 5. データ統合サービスのカスタムプロパティ 注: Informatica は、サードパーティが提供するドキュメント、Informatica が提供するドキュメント、または Informatica のグローバルカスタマサポートに問い合わせることなく、これらのプロパティ値を変更することはお勧めしません。プロパティに関する知識がないまま値を変更すると、パフォーマンスの低下やその他の予期しない結果が発生する可能性があります。

HDFS 接続プロパティ

Hadoop ファイルシステム (HDFS) 接続を使用して、Hadoop クラスタ内のデータにアクセスします。HDFS 接続は、ファイルシステムタイプの接続です。HDFS 接続は、Administrator ツール、Analyst ツール、または Developer ツールで作成および管理できます。HDFS 接続プロパティは、特に明記されている場合を除き、大文字と小文字が区別されます。

注: 接続プロパティの順序は、表示するツールによって異なる場合があります。

以下の表に、HDFS 接続のプロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。名前は 128 文字以内で指定し、空白および以下の特殊文字は使用できません。 ~ ` ! \$ % ^ & * () - + = { [] } \ : ; " ' < , > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。ID の大文字と小文字は区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成した後で変更することはできません。デフォルト値は接続名です。
説明	接続の説明。説明は、765 文字を超えることはできません。
場所	接続を作成するドメイン。Analyst ツールでは使用できません。
タイプ	接続タイプ。デフォルトは Hadoop ファイルシステムです。

プロパティ	説明
ユーザー名	HDFS にアクセスするためのユーザー名。
NameNode の URI	ストレージシステムにアクセスするための URI。 fs.defaultFS の値は、クラスタ設定の core-site.xml 設定セットで確認できます。 注: クラスタ設定をインポートするときに接続を作成する際は、NameNode URI プロパティがデフォルトで設定され、クラスタ設定を更新するたびに更新されます。このプロパティを手動で設定するか、値を上書きした場合、更新操作ではこのプロパティは更新されません。

複数のストレージタイプへのアクセス

さまざまな種類のストレージに接続するには、接続パラメータの NameNode URI プロパティを使用します。次の表は、ストレージの種類とストレージの種類別の NameNode URI 形式を示しています。

記憶領域	NameNode URI 形式
HDFS	hdfs://<namenode>:<port> ここで、 - <namenode>は、NameNode のホスト名または IP アドレスです。 - <port>は、NameNode がリモートプロシージャコール (RPC) をリスンするポートです。 NameNode 高可用性の場合の hdfs://<nameservice>。
MapR-FS	maprfs:///
HDInsight 内の WASB	wasb://<container_name>@<account_name>.blob.core.windows.net/<path> ここで、 - <container_name>は、特定の Azure Storage Blob コンテナを示します。 注: <container_name>は省略可能です。 - <account_name>は、Azure Storage Blob オブジェクトを示します。 例: wasb://infabdmoffering1storage.blob.core.windows.net/infabdmoffering1cluster/mr-history
HDInsight 内の ADLS	adl://home

Azure HDInsight クラスタからクラスタ構成を作成する場合、クラスタ構成ではプライマリストレージとして ADLS または WASB のいずれかを使用します。ADLS または WASB をセカンダリストレージとして指定したクラスタ設定は作成できません。HDFS 接続の NameNode URI プロパティを編集して、ローカルの HDFS の場所に接続することができます。

HBase 接続プロパティ

HBase 接続を使用して HBase にアクセスします。HBase 接続は NoSQL 接続です。HBase 接続は、Administrator ツールまたは Developer ツールで作成および管理できます。Hbase 接続プロパティは、特に明記されている場合を除き、大文字と小文字が区別されます。

以下の表に、HBase 接続のプロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および以下の特殊文字は使用できません。 ~ ` ! \$ % ^ & * () - + = { [] } \ : ; " ' < , > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。
説明	接続の説明。説明は、4,000 文字を超えることはできません。
場所	接続を作成するドメイン。
タイプ	接続タイプ。HBase を選択します。
データベースタイプ	接続先のデータベースの種類。 HBase テーブルの接続を作成するには、[HBase] を選択します。

MapR-DB の HBase 接続のプロパティ

HBase 接続を使用して、MapR テーブルに接続します。HBase 接続は NoSQL 接続です。HBase 接続は、Administrator ツールまたは Developer ツールで作成および管理できます。Hbase 接続プロパティは、特に明記されている場合を除き、大文字と小文字が区別されます。

次の表は、MapR-DB の HBase 接続プロパティを示しています。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および以下の特殊文字は使用できません。 ~ ` ! \$ % ^ & * () - + = { [] } \ : ; " ' < , > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。
説明	接続の説明。説明は、4,000 文字を超えることはできません。

プロパティ	説明
場所	接続を作成するドメイン。
タイプ	接続タイプ。[HBase] を選択します。
データベースタイプ	接続先のデータベースの種類。 MapR-DB テーブルの接続を作成するには、MapR-DB を選択します。
クラスタ設定	Hadoop 環境に関連付けられているクラスタ設定の名前。
MapR-DB のデータベースパス	接続先の MapR-DB テーブルが格納されているデータベースのパス。有効な MapR クラスタパスを入力してください。 MapR-DB の HBase データオブジェクトを作成する場合は、[データベースパス] フィールドで指定した MapR-DB パスに存在するテーブルだけを参照できます。指定したパスのサブディレクトリで利用できるテーブルにはアクセスできません。 例えば、パスを /user/customers/ として指定すると、customers ディレクトリ内のテーブルにアクセスできます。ただし、customers ディレクトリに regions という名前のサブディレクトリが含まれている場合、次のディレクトリ内にあるテーブルにはアクセスできません。 /user/customers/regions

Hive 接続のプロパティ

Hive 接続を利用して、Hive データにアクセスします。Hive 接続は、データベースタイプの接続です。Hive 接続は、Administrator ツール、Analyst ツール、または Developer tool で作成および管理できます。Hive 接続プロパティは、特に明記されている場合を除き、大文字と小文字が区別されます。

注: 接続プロパティの順序は、表示するツールによって異なる場合があります。

以下の表に、Hive 接続のプロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および以下の特殊文字は使用できません。 ~ ` ! \$ % ^ & * () - + = { [] } \ : ; " ' < , > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。
説明	接続の説明。説明は、4000 文字を超えることはできません。
場所	接続を作成するドメイン。Analyst ツールでは使用できません。
タイプ	接続タイプ。Hive を選択します。

プロパティ	説明
LDAP ユーザー名	Hadoop クラスタでマッピングを実行するためにデータ統合サービスが偽装するユーザーの LDAP ユーザー名。ユーザー名は、ネイティブ環境の [メタデータ接続文字列] または [データアクセス接続文字列] に指定する JDBC 接続文字列によって決まります。 Hadoop クラスタが Kerberos 認証を使用する場合、JDBC 接続文字列のプリンシパル名とこのユーザー名が同じである必要があります。Hadoop クラスタが Kerberos 認証を使用しない場合、ユーザー名は JDBC ドライバの動作によって異なります。Hive JDBC ドライバを使用すると、さまざまな方法でユーザー名を指定することができ、そのユーザー名を JDBC URL の一部にすることができます。 Hadoop クラスタが Kerberos 認証を使用しない場合、ユーザー名は JDBC ドライバの動作によって決まります。 ユーザー名を指定しないと、Hadoop クラスタは次の基準に基づいてジョブを認証します。 - Hadoop クラスタが Kerberos 認証を使用しない。データ統合サービスを実行するマシンのオペレーティングシステムのプロファイルユーザー名に基づいてジョブを認証します。 - Hadoop クラスタが Kerberos 認証を使用する。ジョブの認証は、データ統合サービスの SPN に基づいて行います。LDAP ユーザー名は無視されます。
パスワード	LDAP ユーザー名のパスワード。
環境 SQL	Hadoop 環境を設定する SQL コマンド。ネイティブ環境タイプでは、データ統合サービスは Hive メタストアへの接続を作成するたびに環境 SQL を実行します。Hadoop クラスタでのプロファイルの実行に Hive 接続を使用すると、データ統合サービスは各 Hive セッションの最初に環境 SQL を実行します。 どちらの接続モードでも、環境 SQL の使用には以下のルールとガイドラインが適用されます。 - 環境 SQL は、Hive クエリを指定するために使用します。 - 環境 SQL を使用して Hive ユーザー定義関数のクラスパスを設定し、続いて環境 SQL または PreSQL を使用して Hive ユーザー定義関数を指定します。データオブジェクトプロパティで PreSQL を使用してクラスパスを指定することはできません。Hive ユーザー定義関数を使用する場合は、.jar ファイルを次のディレクトリにコピーする必要があります。 <pre><Informatica installation directory>/services/shared/hadoop/<Hadoop distribution name>/extras/hive-auxjars</pre> - 環境 SQL は、PreSQL コマンド内またはカスタムクエリ内で使用する Hadoop パラメータまたは Hive パラメータを定義するために使用できます。 - 環境 SQL プロパティに複数の値を使用する場合は、値の間にスペースを含めないようにしてください。
SQL 識別子文字	特殊文字と SQL の予約済み SQL キーワード (WHERE など) の識別に使用される文字のタイプ。データ統合サービスは特殊文字と予約済み SQL キーワードを選択した文字で囲みます。データ統合サービスはこの文字を [大文字小文字が混在した識別子をサポート] プロパティにも使用します。

ソースまたはターゲットとして Hive にアクセスするためのプロパティ

以下の表に、ソースまたはターゲットとして Hive にアクセスするために設定する接続プロパティを示します。

プロパティ	説明
JDBC ドライバクラス名	Hive JDBC ドライバクラスの名前。このオプションを空白のままにすると、ディストリビューションに同梱されているデフォルトの Apache Hive JDBC ドライバが使用されます。デフォルトの Apache Hive JDBC ドライバが要件に合わない場合は、ドライバクラス名を指定することで、サードパーティの Hive JDBC ドライバを使用して Apache Hive JDBC ドライバをオーバーライドできます。
メタデータ接続文字列	Hadoop サーバーからメタデータにアクセスするために使用される JDBC 接続 URI。 PowerExchange for Hive を使用して、HiveServer サービスまたは HiveServer2 サービスと通信できます。HiveServer に接続するには、次の形式で接続文字列を指定します。 <pre>jdbc:hive2://<hostname>:<port>/<db></pre> ここで - <hostname>は、HiveServer2 が実行されているマシンの名前または IP アドレスです。 - <port>は、HiveServer2 がリスンするポート番号です。 - <db>は、接続先のデータベース名です。データベース名を指定しないと、データ統合サービスはデフォルトのデータベースの詳細を使用します。 HiveServer 2 に接続するには、その特定の Hadoop ディストリビューション用に Apache Hive が実装する接続文字列形式を使用します。Apache Hive 接続文字列形式の詳細については、Apache Hive のドキュメントを参照してください。 ユーザーの偽装については、hive.server2.proxy.user=<xyz> を JDBC 接続 URI に追加する必要があります。ユーザーの偽装を設定しない場合は、現在のユーザーの資格情報を使用して HiveServer2 に接続します。 Hadoop クラスタが SSL または TLS 認証を使用する場合は、ssl=true を JDBC 接続 URI に追加する必要があります。例: jdbc:hive2://<hostname>:<port>/<db>;ssl=true SSL または TLS 認証に自己署名証明書を使用する場合は、クライアントコンピュータとデータ統合サービスコンピュータで証明書ファイルが使用可能であることを確認してください。詳細については、『<i>Informatica Big Data Management Integration ガイド</i>』を参照してください。
Hive JDBC サーバーをバイパス	JDBC ドライバモード。埋め込み JDBC ドライバモードを使用するには、チェックボックスを選択します。 JDBC 埋め込みモードを使用するには、以下のタスクを実行します。 - Hive クライアントと Informatica サービスが同一のマシンにインストールされていることを確認します。 - Hadoop クラスタでマッピングを実行するように Hive 接続プロパティを設定します。 非埋め込みモードを選択する場合は、データアクセス接続文字列を設定する必要があります。 JDBC 埋め込みモードを使用することをお勧めします。
厳密に定義された認証	Hive ソースで厳密に定義された認証に従うオプションを選択すると、マッピングでは次のことに従います。 - 行レベルとカラムレベルの制限。Sentry または Ranger のセキュリティモードが有効になっている Hadoop クラスタに適用されます。 - データマスキングルール。Dynamic Data Masking によって機密データを含むカラムに設定されたマスキングルールに適用されます。 このオプションを選択しない場合、Blaze および Spark エンジンでは制限とマスキングルールを無視し、制限されたデータまたは機密データが結果に含まれます。

プロパティ	説明
データアクセス接続文字列	Hadoop データストアからデータにアクセスするための接続文字列。HiveServer に接続するには、次の形式で非埋め込み JDBC モードの接続文字列を指定します。 <pre>jdbc:hive2://<hostname>:<port>/<db></pre> ここで - <hostname>は、HiveServer2 が実行されているマシンの名前または IP アドレスです。 - <port>は、HiveServer2 がリスンするポート番号です。 - <db>は、接続先のデータベースです。データベース名を指定しないと、データ統合サービスはデフォルトのデータベースの詳細を使用します。 HiveServer 2 に接続するには、特定の Hadoop ディストリビューション用に Apache Hive が実装する接続文字列形式を使用します。Apache Hive 接続文字列形式の詳細については、Apache Hive のドキュメントを参照してください。 ユーザーの偽装については、hive.server2.proxy.user=<xyz>を JDBC 接続 URI に追加する必要があります。ユーザーの偽装を設定しない場合は、現在のユーザーの資格情報を使用して HiveServer2 に接続します。 Hadoop クラスタが SSL または TLS 認証を使用する場合は、ssl=true を JDBC 接続 URI に追加する必要があります。例: jdbc:hive2://<hostname>:<port>/<db>;ssl=true SSL または TLS 認証に自己署名証明書を使用する場合は、クライアントコンピュータとデータ統合サービスコンピュータで証明書ファイルが使用可能であることを確認してください。詳細については、『<i>Informatica Big Data Management Integration Guide</i>』を参照してください。
HDFS 上の Hive ステージングディレクトリ	Hive ステージングテーブル用の HDFS ディレクトリ。Hadoop 偽装ユーザーとマッピング偽装ユーザーに実行権限を付与する必要があります。 このオプションは適用可能で、ネイティブ環境の Hive ターゲットにデータを書き込むときに必要です。
Hive ステージングデータベース名	Hive ステージングテーブルの名前空間。指定されたデータベース名を持たないテーブルには、名前として default を使用してください。 このオプションは、Hive ターゲットにデータを書き込むためにネイティブ環境でマッピングを実行する際に適用できます。 Blaze または Spark エンジンでマッピングを実行する場合、Hive 接続で Hive ステージングデータベース名を設定する必要はありません。データ統合サービスは、Hadoop 接続で設定された値を使用します。

JDBC 接続のプロパティ

JDBC 接続を使用してデータベースのテーブルにアクセスすることができます。JDBC 接続は、Administrator ツール、Developer ツール、または Analyst ツールで作成および管理できます。

注: 接続プロパティの順序は、表示するツールによって異なる場合があります。

以下の表に、JDBC 接続のプロパティを示します。

プロパティ	説明
データベースタイプ	データベースのタイプ。
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。名前は 128 文字以内で指定し、空白および以下の特殊文字は使用できません。 ~ ` ! \$ % ^ & * () - + = { [] } \ : ; " ' < , > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。
説明	接続の説明。説明は、765 文字を超えることはできません。
ユーザー名	データベースユーザー名。
パスワード	データベースユーザー名のパスワード。
JDBC ドライバクラス名	JDBC ドライバクラスの名前。 以下の表に、該当するデータベースタイプに入力できる、ドライバクラス名を示します。 - Oracle 用 DataDirect JDBC ドライバクラス名: <code>com.informatica.jdbc.oracle.OracleDriver</code> - IBM DB2 用 DataDirect JDBC ドライバクラス名: <code>com.informatica.jdbc.db2.DB2Driver</code> - Microsoft SQL Server 用 DataDirect JDBC ドライバクラス名: <code>com.informatica.jdbc.sqlserver.SQLServerDriver</code> - Sybase ASE 用 DataDirect JDBC ドライバクラス名: <code>com.informatica.jdbc.sybase.SybaseDriver</code> - Informix 用 DataDirect JDBC ドライバクラス名: <code>com.informatica.jdbc.informix.InformixDriver</code> - MySQL 用 DataDirect JDBC ドライバクラス名: <code>com.informatica.jdbc.mysql.MySQLDriver</code> 特定のデータベースで使用するドライバクラスの詳細については、ベンダ提供のドキュメントを参照してください。

プロパティ	説明
接続文字列	データベースへの接続に使用する接続文字列。以下の接続文字列を使用します。 <code>jdbc:<subprotocol>:<subname></code> 次のリストに、該当するデータベースの種類で入力できる接続文字列の例を示します。 - DataDirect Oracle JDBC ドライバの接続文字列: <code>jdbc:informatica:oracle://<host>:<port>;SID=<value></code> - Oracle JDBC ドライバの接続文字列: <code>jdbc:oracle:thin:@//<host>:<port>:<SID></code> - DataDirect IBM DB2 JDBC ドライバの接続文字列: <code>jdbc:informatica:db2://<host>:<port>;DatabaseName=<value></code> - IBM DB2 JDBC ドライバの接続文字列: <code>jdbc:db2://<host>:<port>/<database_name></code> - DataDirect Microsoft SQL Server JDBC ドライバの接続文字列: <code>jdbc:informatica:sqlserver://<host>;DatabaseName=<value></code> - SQL Server JDBC ドライバの接続文字列: <code>jdbc:sqlserver://<host>;DatabaseName=<value></code> - Netezza JDBC ドライバの接続文字列: <code>jdbc:netezza://<host>:<port>/<database_name></code> - Pivotal Greenplum ドライバの接続文字列: <code>jdbc:pivotal:greenplum://<host>:<port>/<database_name>=<value></code> - Postgres Greenplum ドライバの接続文字列: <code>jdbc:postgresql://<host>:<port>/<database_name></code> - Teradata JDBC ドライバの接続文字列: <code>jdbc:teradata://<host>/<database_name>=<value>,tmode=<value>,charset=<value></code> 特定のドライバで使用する接続文字列の詳細については、ベンダ提供のマニュアルを参照してください。
環境 SQL	オプション。データベースに接続するとき、SQL コマンドを入力してデータベースの環境設定を行います。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。 注: Sqoop を有効にすると、Sqoop はこのプロパティを無視します。
トランザクション SQL	オプション。データベースに接続するとき、SQL コマンドを入力してデータベースの環境設定を行います。データ統合サービスにより、トランザクション環境 SQL が各トランザクションの開始時に実行されます。 注: Sqoop を有効にすると、Sqoop はこのプロパティを無視します。
SQL 識別子文字	データベースが、SQL クエリで区切り識別子を囲むのに使用する文字のタイプ。使用できる文字は、データベースタイプによって異なります。 データベースで通常識別子が使用される場合、[なし] を選択します。データ統合サービスで SQL クエリが生成される際、識別子の周りに区切り文字は配置されません。 データベースで区切り識別子が使用される場合、文字を選択します。データ統合サービスで SQL クエリが生成される際、区切り識別子はこの文字で囲まれます。 注: Sqoop を有効にすると、Sqoop はこのプロパティを無視します。
大文字と小文字が混在する識別子をサポート	データベースで大文字と小文字を区別する識別子が使用される場合、有効にします。有効にした場合、データ統合サービスは、すべての識別子を [SQL 識別子文字] プロパティに対して選択された文字で囲みます。 [SQL 識別子文字] プロパティが [なし] に設定されている場合、[大文字と小文字が混在する識別子をサポート] プロパティは無効になっています。 注: Sqoop を有効にする場合、ターゲットをランタイムに作成または置換する DDL スクリプトを生成して実行すると、Sqoop はこのプロパティを受け入れます。他のすべてのシナリオでは、Sqoop はこのプロパティを無視します。

Sqoop の接続レベルの引数

JDBC 接続では、Sqoop がデータベースへの接続に使用する必要がある引数を定義できます。データ統合サービスは、指定された引数を、JDBC 接続プロパティに基づいて作成するデフォルトコマンドとマージします。指定した引数は、JDBC 接続プロパティよりも優先されます。

同じドライバを使用してメタデータのインポートとマッピングの実行を行い、追加の Sqoop 引数を指定しない場合は、JDBC 接続の【使用する Sqoop のバージョン】リストから【Sqoop v1.x】を選択して、【Sqoop 引数】フィールドは空のままにします。データ統合サービスは、指定された JDBC 接続プロパティに基づいて Sqoop コマンドを作成します。

ただし、ランタイムタスクに別のドライバを使用する場合や、追加のランタイム Sqoop 引数を指定する場合は、【使用する Sqoop のバージョン】リストから【Sqoop v1.x】を選択して、【Sqoop 引数】フィールドに引数を指定します。

JDBC 接続では、次の Sqoop 引数を設定できます。

ドライバ

Sqoop がデータベースに接続するために使用する必要がある JDBC ドライバクラスを定義します。

以下の構文を使用します。

```
--driver <JDBC driver class>
```

例えば、接続するデータベースタイプに応じて、次の構文を使用します。

- Aurora: --driver com.mysql.jdbc.Driver
- Greenplum: --driver org.postgresql.Driver
- IBM DB2: --driver com.ibm.db2.jcc.DB2Driver
- IBM DB2 z/OS: --driver com.ibm.db2.jcc.DB2Driver
- Microsoft SQL Server: --driver com.microsoft.sqlserver.jdbc.SQLServerDriver
- Netezza: --driver org.netezza.Driver
- Oracle: --driver oracle.jdbc.driver.OracleDriver
- Teradata: --driver com.teradata.jdbc.TeraDriver

接続

Sqoop がデータベースに接続するために使用する必要がある JDBC 接続文字列を定義します。JDBC 接続文字列は、ドライバの引数で定義するドライバに基づく必要があります。

以下の構文を使用します。

```
--connect <JDBC connection string>
```

例えば、接続するデータベースタイプに応じて、次の構文を使用します。

- Aurora: --connect "jdbc:mysql://<host_name>:<port>/<schema_name>"
- Greenplum: --connect jdbc:postgresql://<host_name>:<port>/<database_name>
- IBM DB2: --connect jdbc:db2://<host_name>:<port>/<database_name>
- IBM DB2 z/OS: --connect jdbc:db2://<host_name>:<port>/<database_name>
- Microsoft SQL Server: --connect jdbc:sqlserver://<host_name>:<port> or named_instance=<database_name>
- Netezza: --connect "jdbc:netezza://<database_server_name>:<port>/<database_name>;schema=<schema_name>"
- Oracle: --connect jdbc:oracle:thin:@<database_host_name>:<database_port>:<database_SID>

- Teradata: `--connect jdbc:teradata://<host_name>/database=<database_name>`

connection-manager

Sqoop がデータベースに接続するために使用する必要がある接続マネージャのクラス名を定義します。

以下の構文を使用します。

```
--connection-manager <connection manager class name>
```

例えば、汎用 JDBC マネージャのクラス名を使用するには、次の構文を使用します。

```
--connection-manager org.apache.sqoop.manager.GenericJdbcManager
```

直接

Oracle に対してデータの読み書きを行う場合、直接引数を設定して、Sqoop が OraOop を使用できるようにすることができます。OraOop は、ネイティブプロトコルを使用して Oracle データベースに接続する Oracle 用の特殊な Sqoop プラグインです。OraOop を構成すると、パフォーマンスが向上します。

Spark エンジンで Sqoop マッピングを実行するときに、OraOop を設定できます。

以下の構文を使用します。

```
--direct
```

OraOop を使用する場合は、次の構文を使用して複数の引数を指定する必要があります。

```
-D<argument=value> -D<argument=value>
```

注: 複数の引数を指定し、-D と引数の名前と値のペアの間に空白文字を含めた場合、Sqoop は最初の引数のみを考慮し、残りの引数は無視します。

ジョブを特定のキューに送らない場合、Spark エンジンはデフォルトのキューを使用します。

-Dsqoop.connection.factories

Teradata Connector for Hadoop (TDCH) の Sqoop 専用コネクタを使用して Blaze エンジンでマッピングを実行するには、-Dsqoop.connection.factories 引数を設定する必要があります。この引数を使用して、Sqoop が使用する必要がある TDCH 接続のファクトリクラスを定義します。接続ファクトリクラスは、使用する TDCH Sqoop コネクタによって異なります。

- Cloudera Connector Powered by Teradata を使用するには、次のように -Dsqoop.connection.factories 引数を設定します。
`-Dsqoop.connection.factories=com.cloudera.connector.teradata.TeradataManagerFactory`
- Hortonworks Connector for Teradata (Teradata Connector for Hadoop 提供) を使用するには、次のように -Dsqoop.connection.factories 引数を設定します。
`-Dsqoop.connection.factories=org.apache.sqoop.teradata.TeradataManagerFactory`

注: Spark エンジンでマッピングを実行するために -Dsqoop.connection.factories 引数を設定する必要はありません。デフォルトでは、データ統合サービスは Cloudera Connector Powered by Teradata と Hortonworks Connector for Teradata (Teradata Connector for Hadoop 提供) を呼び出します。

--infaoptimize

Spark エンジンでの Sqoop パススルーマッピングのパフォーマンス最適化を無効にするには、この引数を使用します。

Sqoop パススルーマッピングを Spark エンジンで実行すると、データ統合サービスでは次のシナリオでマッピングパフォーマンスを最適化します。

- Sqoop ソースからデータを読み取り、テキスト形式を使用する Hive ターゲットにデータを書き込みます。

- Sqoop ソースからデータを読み取り、Flat、Avro、または Parquet 形式を使用する HDFS ターゲットにデータを書き込みます。

パフォーマンス最適化を無効にするには、`--infaoptimize` 引数を `false` に設定します。例えば、最適化された Sqoop マッピングを実行した後にデータ型の問題が発生する場合に、パフォーマンス最適化を無効にすることができます。

以下の構文を使用します。

```
--infaoptimize false
```

設定できる Sqoop 引数の完全なリストについては、Sqoop のマニュアルを参照してください。

Kafka 接続のプロパティ

Kafka 接続はメッセージング接続です。Kafka 接続を使用して、Kafka にメッセージターゲットとして接続します。Developer tool または infacmd で Kafka 接続を作成および管理できます。

次の表に、Kafka 接続の一般的な接続プロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および以下の特殊文字は使用できません。 ~ ` ! \$ % ^ & * () - + = { [] } \ : ; " ' < , > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。
説明	接続の説明。接続を識別するために使用できる文字列を入力します。説明は、4,000 文字を超えることはできません。
場所	接続を作成するドメイン。
タイプ	接続タイプ。

次の表に、Kafka 接続の Kafka Broker プロパティを示します。

プロパティ	説明
Kafka Broker リスト	Kafka Broker のカンマ区切りのリストであり、Kafka メッセージングブローカーの設定を保持しています。 Kafka Broker を指定するには、次の形式を使用します。 <IP Address>:<port>
ZooKeeper ホストのポートリスト	オプション。Apache ZooKeeper のカンマ区切りのリストであり、Kafka メッセージングブローカーの設定を保持しています。 ZooKeeper を指定するには、次の形式を使用します。 <IP Address>:<port>

プロパティ	説明
再試行タイムアウト	統合サービスがデータを書き込むために Kafka Broker への再接続を試行する秒数。指定した時間にソースまたはターゲットが利用できない場合は、データ損失を回避するために、マッピング実行が停止します。
Kafka Broker のバージョン	Kafka メッセージングブローカーのバージョンを 0.10.1.x-2.0.0 に設定します。

Microsoft Azure Blob ストレージ接続のプロパティ

注: 接続プロパティの順序は、表示するツールによって異なる場合があります。

次の表に、Microsoft Azure Blob ストレージ接続のプロパティを示します。

プロパティ	説明
名前	Microsoft Azure Blob ストレージ接続の名前。
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。
説明	接続の説明。
場所	接続を作成するドメイン。
タイプ	接続タイプ。AzureBlob を選択します。

【接続の詳細】 タブには、Microsoft Azure Blob ストレージ接続の接続属性が含まれています。以下の表に、接続属性を示します。

プロパティ	説明
アカウント名	Microsoft Azure ストレージアカウントの名前。
アカウントキー	Microsoft Azure ストレージアクセスキー。
コンテナ名	ルートコンテナまたは絶対パスを持つサブフォルダ。
エンドポイントサフィックス	Microsoft Azure エンドポイントのタイプ。以下の任意のエンドポイントを選択できます。 - core.windows.net: デフォルト - core.usgovcloudapi.net: 米国合衆国政府の Microsoft Azure エンドポイントを選択する場合 - core.chinacloudapi.cn: 該当なし

Microsoft Azure Cosmos DB SQL API 接続のプロパティ

Cosmos DB 接続を使用して、Cosmos DB データベースに接続します。Cosmos DB 接続の作成時には、メタデータおよびデータアクセスの情報を入力します。

プロパティ	説明
名前	
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。
説明	接続の説明。説明は、765 文字を超えることはできません。
場所	
タイプ	Microsoft Azure Cosmos DB SQL API
Cosmos DB URI	Microsoft Azure Cosmos DB アカウントの URI。
キー	
データベース	JSON ドキュメントとの間での読み書きするコレクションが格納されているデータベース名。

注: Cosmos DB URI とキー値は、Azure ポータルのキー設定から見つかります。詳細については、Azure 管理者に問い合わせてください。

Microsoft Azure Data Lake Store 接続プロパティ

注: 接続プロパティの順序は、表示するツールによって異なる場合があります。

Microsoft SQL Server 接続は、Administrator ツールまたは Developer tool で作成および管理できます。以下の表に、Microsoft Azure Data Lake Store 接続プロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および次の特殊文字は使用できません。~`!\$%^&*()-+={} \;\:'<,>./
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。

プロパティ	説明
説明	接続の説明。説明は、4,000 文字を超えることはできません。
場所	接続を作成するドメイン。
タイプ	接続タイプ。Microsoft Azure Data Lake Store を選択します。

次の表に、メタデータアクセスのプロパティを示します。

プロパティ	説明
ADLS アカウント名	Microsoft Azure Data Lake Store の名前。
ClientID	Active Directory で OAuth 認証を完了するためのアプリケーションの ID。
クライアントシークレット	Active Directory で OAuth 認証を完了するためのクライアント秘密鍵。
ディレクトリ	データの読み取りまたは書き込みに使用する Microsoft Azure Data Lake Store ディレクトリ。デフォルトはルートディレクトリです。
AuthEndpoint	クライアント ID に基づいてアクセスコードが生成され、クライアントシークレットが完了する OAuth 2.0 トークンエンドポイント。

クライアント ID、クライアントシークレット、および認証エンドポイントの作成に関する詳細については、Azure 管理者にお問い合わせいただくか、Microsoft Azure Data Lake Store ドキュメントを参照してください。

Microsoft Azure SQL Data Warehouse 接続プロパティ

注: 接続プロパティの順序は、表示するツールによって異なる場合があります。

Microsoft SQL Server 接続は、Administrator ツールまたは Developer tool で作成および管理できます。次の表に、Microsoft Azure SQL Data Warehouse 接続プロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および次の特殊文字は使用できません。~`!\$%^&*()-+=[] \\:;'"<,>./
ID	データ統合サービスが接続の識別に使用する文字列。ID では大文字と小文字が区別されません。255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。デフォルト値は接続名です。
説明	接続の説明。説明は、4,000 文字を超えることはできません。

プロパティ	説明
場所	接続を作成するドメイン。
タイプ	接続タイプ。Microsoft Azure SQL Data Warehouse を選択します。

次の表に、メタデータアクセスのプロパティを示します。

プロパティ	説明
Azure DW JDBC URL	Microsoft Azure Data Warehouse JDBC 接続文字列。例えば、次の接続文字列を入力できます: jdbc:sqlserver:// <Server>.database.windows.net:1433;database=<Database>。管理者は Microsoft Azure ポータルから URL をダウンロードできます。
Azure DW JDBC ユーザー名	Microsoft Azure SQL Data Warehouse アカウントに接続するためのユーザー名。Microsoft Azure SQL Data Warehouse のデータを読み取り、書き込み、切り結めるための権限が必要です。
Azure DW JDBC パスワード	Microsoft Azure SQL Data Warehouse アカウントに接続するためのパスワード。
Azure DW スキーマ名	Microsoft Azure SQL Data Warehouse 内のスキーマの名前。
Azure Blob アカウント名	ファイルをステージングする Microsoft Azure ストレージアカウントの名前。
Azure Blob アカウントキー	Blob ストレージアカウントへのアクセスを認証するキー。
Blob エンドポイント	Microsoft Azure エンドポイントのタイプ。以下の任意のエンドポイントを選択できます。 - core.windows.net: デフォルト - core.usgovcloudapi.net: 米国合衆国政府の Microsoft Azure エンドポイントを選択する場合 - core.chinacloudapi.cn: 該当なし

Snowflake 接続プロパティ

Snowflake 接続を設定するときに、接続プロパティを設定する必要があります。

注: 接続プロパティの順序は、表示するツールによって異なる場合があります。

次の表に、Snowflake 接続プロパティを示します。

プロパティ	説明
名前	接続の名前。この名前では、大文字と小文字が区別されず、ドメイン内で一意にする必要があります。このプロパティは、接続を作成した後に変更できます。名前は 128 文字以内で指定し、空白および次の特殊文字は使用できません。~`!\$%^&*()-+=[{}] \\:;'"<, > . ? /
ID	データ統合サービスが接続の識別に使用する文字列。 ID では大文字と小文字が区別されません。ID は 255 文字以下で、ドメイン内で一意である必要があります。このプロパティは、接続を作成してから変更できません。 デフォルト値は接続名です。
説明	オプション。接続の説明。説明は、4,000 文字を超えることはできません。
場所	接続を作成するドメイン。
タイプ	接続タイプ。Snowflake を選択します。
ユーザー名	Snowflake アカウントに接続するユーザー名。
パスワード	Snowflake アカウントに接続するためのパスワード。
アカウント	Snowflake アカウントの名前。
ウェアハウス	Snowflake ウェアハウス名。
ロール	ユーザーに割り当てられた Snowflake ロール。
追加の JDBC URL パラメータ	以下の形式で、1 つ以上の JDBC 接続パラメータを入力します。 <param1>=<value>&<param2>=<value>&<param3>=<value>.... 以下に例を示します。 user=jon&warehouse=mywh&db=mydb&schema=public Okta SSO 認証を介して Snowflake にアクセスするには、次の形式で SAML 2.0 プロトコルを実装する Web ベースの IdP を入力します。 authenticator=https://<Your_Okta_Account_Name>.okta.com 注: Microsoft ADFS はサポートされていません。 Okta 認証の設定の詳細については、次の Web サイトを参照してください: https://docs.snowflake.net/manuals/user-guide/admin-security-fed-auth-configure-snowflake.html#configuring-snowflake-to-use-federated-authentication

ソースまたはターゲットにアクセスするための接続の作成

データオブジェクト、プレビューデータ、プロファイルデータをインポートする前に、接続を作成します。

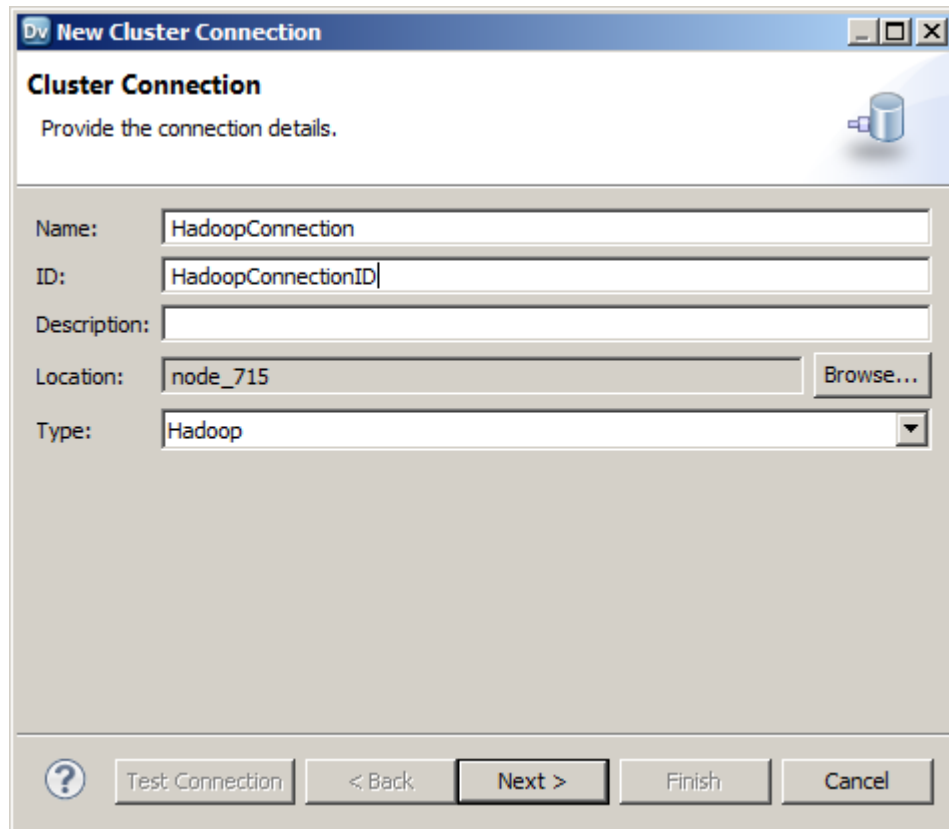
1. Administrator ツール内で、**【管理】** > **【接続】** をクリックします。
2. **【アクション】** > **【新規】** > **【接続】** を選択します。
3. 作成する接続のタイプを選択します。
 - HBase 接続を選択するには、**【NoSQL】** > **【HBase】** を選択します。
 - HDFS 接続を選択するには **【ファイルシステム】** > **【Hadoop ファイルシステム】** を選択します。
 - Hive 接続を選択するには、**【データベース】** > **【Hive】** を選択します。
 - JDBC 接続を選択するには、**【データベース】** > **【JDBC】** を選択します。
4. **【OK】** をクリックします。
5. 接続名、ID、説明（省略可能）を入力します。
6. 接続のプロパティを設定します。Hive 接続の場合、**【ソースまたはターゲットとして、Hive にアクセス】** オプションを選択し、ソースまたはターゲットとして Hive を使用する必要があります。
7. **【テスト接続】** をクリックし、接続を確認します。
8. **【完了】** をクリックします。

Hadoop 接続の作成

Hadoop 環境でマッピングを実行する前に Hadoop 接続を作成します。

1. **【ウィンドウ】** > **【設定】** をクリックします。
2. **【Informatica】** > **【接続】** を選択します。
3. **【使用可能な接続】** リストでドメインを展開します。
4. **【使用可能な接続】** リストで **【クラスタ】** 接続タイプを選択して、**【追加】** をクリックします。
【新しいクラスタ接続】 ダイアログボックスが表示されます。

5. 接続の全般プロパティを入力します。



6. **【次へ】** をクリックします。
7. Hadoop クラスタプロパティ、共通プロパティ、および拒否ディレクトリプロパティを入力します。
8. **【次へ】** をクリックします。
9. **【次へ】** をクリックします。

バージョン 10.2.2 から、Informatica では Hive エンジンのサポートを終了しました。Hive 設定プロパティは入力できません。

10. Blaze エンジンの設定プロパティを入力し、**【次へ】** をクリックします。
11. Spark エンジンの設定プロパティを入力し、**【完了】** をクリックします。

Hadoop 接続プロパティの設定

Hadoop 接続を作成する場合、デフォルト値が、クラスタ環境変数、クラスタパスプロパティ、および詳細プロパティに割り当てられます。このプロパティの値は追加または編集することができます。また、デフォルト値にリセットすることもできます。

使用するクラスタ環境と機能に基づいて、次の Hadoop 接続プロパティを設定できます。

- クラスタ環境変数
- クラスタのライブラリパス
- 共通する詳細プロパティ

- Blaze エンジンの詳細プロパティ
- Spark エンジンの詳細プロパティ

注: Informatica は、サードパーティが提供するドキュメント、Informatica が提供するドキュメント、または Informatica のグローバルカスタマサポートに問い合わせることなく、これらのプロパティ値を変更することはお勧めしません。プロパティに関する知識がないまま値を変更すると、パフォーマンスの低下やその他の予期しない結果が発生する可能性があります。

デフォルト値にリセットするには、プロパティ値を削除します。例えば、編集したクラスタのライブラリパスプロパティを削除すると、値はデフォルト \$DEFAULT_CLUSTER_LIBRARY_PATH にリセットされます。

クラスタ環境変数

【クラスタ環境変数】 プロパティには、クラスタが使用する環境変数が表示されます。各環境変数には名前および値が含まれます。環境変数を追加または編集できます。

テキストボックスのプロパティを編集するには、次の形式で `&`: を使用して名前と値のペアを区別します。

```
<name1>=<value1>[&:<name2>=<value2>...&:<nameN>=<valueN>]
```

【**クラスタ環境変数**】 プロパティで次の環境変数を設定します。

HADOOP_NODE_JDK_HOME

クラスタノードが使用するクラスタサービスおよび JDK バージョンの実行元のディレクトリを表します。Blaze エンジンの Hadoop 環境および Sqoop マッピングで Java トランスフォーメーションを実行する必要があります。デフォルトは `/usr/java/default` です。データ統合サービスで使用する JDK バージョンは、クラスタの JDK バージョンとの互換性が必要です。

<クラスタ JDK ホーム>/jdk<バージョン>に設定します。

例えば、`HADOOP_NODE_JDK_HOME=<cluster JDK home>/jdk<version>`です。

クラスタのライブラリパス

【クラスタのライブラリパス】 プロパティは、クラスタで共有するライブラリのパス変数のリストです。ライブラリパス変数を追加または編集できます。

テキストボックスのプロパティを編集するには、次の形式で `:` を使用して各パス変数を区切ります。

```
<variable1>[:<variable2>...:<variableN>]
```

【**クラスタライブラリパス**】 プロパティでライブラリパス変数を設定します。

共通する詳細プロパティ

共通する詳細プロパティは、Hadoop 環境に固有の詳細プロパティまたはカスタムプロパティのリストです。プロパティは、Blaze エンジンおよび Spark エンジンに共通です。各プロパティには名前および値が含まれます。詳細プロパティを追加または編集できます。

テキストボックスのプロパティを編集するには、次の形式で `&`: を使用して名前と値のペアを区別します。

```
<name1>=<value1>[&:<name2>=<value2>...&:<nameN>=<valueN>]
```

共通するプロパティセクションの【**詳細プロパティ**】で次のプロパティを設定します。

infapdo.java.opts

Java ランタイム環境をカスタマイズする Java オプションのリスト。プロパティにはデフォルト値が含まれます。

MapR 環境のマッピングに統合トランスフォーメーションまたは一致トランスフォーメーションが含まれる場合は、次の値を変更します。

- -Xmx512M。Java 仮想メモリの最大サイズを指定します。デフォルトは 512MB です。700 MB 以上に値を増やします。
例えば、infapdo.java.opts=-Xmx700M です。

Blaze エンジン詳細プロパティ

Blaze の詳細プロパティは、Blaze エンジンに固有の詳細プロパティまたはカスタムプロパティのリストです。各プロパティには名前および値が含まれます。詳細プロパティを追加または編集できます。

テキストボックスのプロパティを編集するには、名前と値の各ペアを&:で区切る次の形式を使用します。

<name1>=<value1>[&:<name2>=<value2>...&:<nameN>=<valueN>]

[Blaze 設定] セクションの **【詳細プロパティ】** で、次のプロパティを設定します。

infagrid.cadi.namespace

データ統合サービスが使用する名前空間。複数の Blaze インスタンスを設定するために必要です。

<一意の名前空間>に設定します。

例: infagrid.cadi.namespace=TestUser1_namespace

infagrid.blaze.console.jsfport

Blaze エンジンコンソールの JSF ポート。他のクラスタプロセスで使用されていないポート番号を使用します。複数の Blaze インスタンスを設定するために必要です。

<一意の JSF ポート値>に設定します。

例: infagrid.blaze.console.jsfport=9090

infagrid.blaze.console.httpport

Blaze エンジンコンソールの HTTP ポート。他のクラスタプロセスで使用されていないポート番号を使用します。複数の Blaze インスタンスを設定するために必要です。

<一意の HTTP ポート値>に設定します。

例: infagrid.blaze.console.httpport=9091

infagrid.node.local.root.log.dir

Blaze サービスログのパス。デフォルトは/tmp/infal/logs/blaze です。複数の Blaze インスタンスを設定するために必要です。

<ローカルの Blaze サービスログディレクトリ>に設定します。

例: infagrid.node.local.root.log.dir=<directory path>

infacal.hadoop.logs.directory

永続的な Blaze ログの HDFS 内のパス。デフォルトは/var/log/hadoop-yarn/apps/informatica です。複数の Blaze インスタンスを設定するために必要です。

<永続ログディレクトリのパス>に設定します。

例: infacal.hadoop.logs.directory=<directory path>

infagrid.node.hadoop.local.root.log.dir

サービスログディレクトリの Hadoop 接続のパス。

<サービスログディレクトリのパス>に設定します。

例: infagrid.node.local.root.log.dir=\$HADOOP_NODE_INFA_HOME/blazeLogs

Spark の詳細プロパティ

Spark の詳細プロパティは、Spark エンジンに固有の詳細プロパティまたはカスタムプロパティのリストです。各プロパティには名前および値が含まれます。詳細プロパティを追加または編集できます。各プロパティには名前および値が含まれます。詳細プロパティを追加または編集できます。

[Spark 設定] セクションの **【詳細プロパティ】** で、次のプロパティを設定します。

テキストボックスのプロパティを編集するには、次の形式で `&:` を使用して名前と値のペアを区別します。

`<name1>=<value1>[&:<name2>=<value2>...&:<nameN>=<valueN>]`

`spark.authenticate`

Hadoop での Spark サービスの認証を有効にします。Spark の暗号化に必要です。

TRUE に設定します。

例: `spark.authenticate=TRUE`

`spark.authenticate.enableSaslEncryption`

SASL 認証が有効になっている場合に、暗号化された通信を有効にします。Spark の暗号化で SASL 認証が使用される場合に必要です。

TRUE に設定します。

例: `spark.authenticate.enableSaslEncryption=TRUE`

`spark.executor.cores`

各実行プログラムプロセスが Spark エンジンでタスクレットを実行するために使用するコアの数を示します。

`spark.executor.cores=1` に設定します。

`spark.executor.instances`

各実行プログラムプロセスが Spark エンジンでタスクレットを実行するために使用するインスタンスの数を示します。

`spark.executor.instances=1` に設定します。

`spark.executor.memory`

各実行プログラムプロセスが Spark エンジンでタスクレットを実行するために使用するメモリの量を示します。

`spark.executor.memory=3G` に設定します。

`infaspark.driver.cluster.mode.extraJavaOptions`

クラスタ内で実行される Spark ドライバ用の追加 Java オプションのリスト。Kerberos 認証を使用する Kafka クラスタとの間の読み書きを行うストリーミングマッピングに必要です。

例えば、以下を設定します。

```
infaspark.driver.cluster.mode.extraJavaOptions=  
-Djava.security.egd=file:/dev/./urandom  
-XX:MaxMetaspaceSize=256M -Djavax.security.auth.useSubjectCredsOnly=true  
-Djava.security.krb5.conf=<path to keytab file>/krb5.conf  
-Djava.security.auth.login.config=<path to jaas config>/kafka_client_jaas.config
```

特定のユーザーのプロパティを設定するには、次のコード行を含めます。

```
infaspark.driver.cluster.mode.extraJavaOptions =  
-Djava.security.egd=file:/dev/./urandom  
-XX:MaxMetaspaceSize=256M -XX:+UseG1GC -XX:MaxGCPauseMillis=500  
-Djava.security.krb5.conf=/etc/krb5.conf
```

infaspark.executor.extraJavaOptions

Spark 実行プログラム用の追加 Java オプションのリスト。Kerberos 認証を使用する Kafka クラスタとの間の読み書きを行うストリーミングマッピングに必要です。

例えば、以下を設定します。

```
infaspark.executor.extraJavaOptions=  
-Djava.security.egd=file:/dev/./urandom  
-XX:MaxMetaspaceSize=256M -Djavax.security.auth.useSubjectCredsOnly=true  
-Djava.security.krb5.conf=/<path to krb5.conf file>/krb5.conf  
-Djava.security.auth.login.config=/<path to jAAS config>/kafka_client_jaas.config
```

特定のユーザーのプロパティを設定するには、次のコード行を含めます。

```
infaspark.executor.extraJavaOptions =  
-Djava.security.egd=file:/dev/./urandom  
-XX:MaxMetaspaceSize=256M -XX:+UseG1GC -XX:MaxGCPauseMillis=500  
-Djava.security.krb5.conf=/etc/krb5.conf
```

infaspark.flatfile.writer.nullValue

Databricks Spark エンジンがターゲットに書き込むときに、NULL 値は空の文字列（「」）に変換されます。例: 12, AB, "", 23p09udj。

Databricks Spark エンジンでは文字列カラムに空の文字列を書き込むことができますが、空の文字列を非文字列カラムに書き込もうとすると、タイプの不一致でマッピングが失敗します。

Databricks Spark エンジンで空の文字列を NULL 値に戻してターゲットに書き込めるようにするには、Databricks Spark 接続に次の詳細プロパティを設定します。

```
infaspark.flatfile.writer.nullValue=true
```

spark.hadoop.validateOutputSpecs

HBase テーブルが存在するかどうかを検証します。Amazon EMR クラスタの HBase ターゲットに書き込みを行うストリーミングマッピングに必要です。値を false に設定します。

infaspark.json.parser.mode

破損した JSON レコードの処理方法をパーサーに指定します。次のいずれかのモードに値を設定できます。

- DROPMALFORMED。パーサーは破損したすべてのレコードを無視します。デフォルトのモードです。
- PERMISSIVE。パーサーは、破損したレコードの標準ではないフィールドを NULL として受け取ります。
- FAILFAST。パーサーが破損したレコードを検出し、Spark アプリケーションがダウンした場合、パーサーは例外を生成します。

infaspark.json.parser.multiLine

JSON ファイルの複数行のレコードをパーサーが読み取れるかどうかを指定します。この値は、true または false に設定できます。デフォルトは false です。Spark バージョン 2.2.x 以上を使用する非ネイティブのディストリビューションにのみ適用します。

infaspark.pythontx.exec

Big Data Management 用の Spark エンジンで Python トランスフォーメーションを実行するために必要です。Hadoop クラスタの作業ノード上の Python 実行可能バイナリの場所。

例えば、以下を設定します。

```
infaspark.pythontx.exec=/usr/bin/python3.4
```

データ統合サービスマシン上で Python のインストールを使用する場合、データ統合サービスマシン上の Informatica インストールディレクトリの Python 実行可能バイナリに値を設定します。

例えば、以下を設定します。

```
infaspark.pythontx.exec=INFA_HOME/services/shared/spark/python/lib/python3.4
```

`infaspark.pythontx.executorEnv.PYTHONHOME`

Big Data Management と Big Data Streaming 用の Spark エンジンで Python トランスフォーメーションを実行するために必要です。Hadoop クラスタの作業ノード上の Python インストールディレクトリの場所。

作業ノードの Python インストールディレクトリが `usr/lib/python` などのディレクトリ内にある場合、次の値にプロパティを設定します。

```
infaspark.pythontx.executorEnv.PYTHONHOME=/usr/lib/python
```

データ統合サービスマシン上で Python のインストールを使用する場合、データ統合サービスマシン上の Python インストールディレクトリの場所を使用します。

例えば、次の値にプロパティを設定します。

```
infaspark.pythontx.executorEnv.PYTHONHOME=
INFA_HOME/services/shared/spark/python/
```

`infaspark.pythontx.executorEnv.LD_PRELOAD`

Big Data Streaming 用の Spark エンジンで Python トランスフォーメーションを実行するために必要です。データ統合サービスマシン上にある、Python インストールフォルダ内の Python 共有ライブラリの場所。

例えば、以下を設定します。

```
infaspark.pythontx.executorEnv.LD_PRELOAD=
INFA_HOME/services/shared/spark/python/lib/libpython3.6m.so
```

`infaspark.pythontx.submit.lib.JEP_HOME`

Big Data Streaming 用の Spark エンジンで Python トランスフォーメーションを実行するために必要です。データ統合サービスマシン上にある、Python インストールフォルダ内の Jep パッケージの場所。

例えば、以下を設定します。

```
infaspark.pythontx.submit.lib.JEP_HOME=
INFA_HOME/services/shared/spark/python/lib/python3.6/site-packages/jep/
```

`spark.shuffle.encryption.enabled`

認証が有効になっている場合に、暗号化された通信を有効にします。Spark の暗号化に必要です。

TRUE に設定します。

例: `spark.shuffle.encryption.enabled=TRUE`

`spark.scheduler.maxRegisteredResourcesWaitingTime`

タスクをスケジュール設定する前に、リソースの登録を待機する時間（ミリ秒）。デフォルトは 30000 です。Spark ジョブの実行を開始する前に、この値を小さくして遅延を減らします。Spark エンジンでのマッピングのパフォーマンスを向上させるために必要です。

15000 に設定します。

例: `spark.scheduler.maxRegisteredResourcesWaitingTime=15000`

spark.scheduler.minRegisteredResourcesRatio

タスクのスケジュール設定が開始される前に取得される登録済みリソースの最小比率。デフォルトは 0.8 です。Spark ジョブの実行を開始する前に、この値を小さくして遅延を減らします。Spark エンジンでのマッピングのパフォーマンスを向上させるために必要です。

0.5 に設定します。

例: spark.scheduler.minRegisteredResourcesRatio=0.5

付録 B

データ型リファレンス

この付録では、以下の項目について説明します。

- [データ型リファレンスの概要, 279 ページ](#)
- [非ネイティブ環境でのトランスフォーメーションデータ型のサポート, 280 ページ](#)
- [複合ファイルデータ型とトランスフォーメーションデータ型, 281 ページ](#)
- [Hive のデータ型とトランスフォーメーションデータ型, 284 ページ](#)
- [Sqoop のデータ型, 286 ページ](#)

データ型リファレンスの概要

Informatica Developer は、非ネイティブ環境で実行するマッピングで次のデータ型を使用します。

ネイティブデータ型

ネイティブデータ型は、物理データオブジェクトとして使用されるソースおよびターゲットに固有です。ネイティブデータ型は、物理データオブジェクトカラムのプロパティに表示されます。

トランスフォーメーションデータ型

トランスフォーメーションデータ型は、トランスフォーメーションに表示されるデータ型のセットです。ANSI SQL-92 汎用データ型に基づく内部データ型で、プラットフォーム間でデータを移動するときデータ統合サービスによって使用されます。トランスフォーメーションデータ型は、マッピング内のすべてのトランスフォーメーションに表示されます。

トランスフォーメーションデータ型には、次のデータ型があります。

- プリミティブデータ型。単一のカラム位置にある 1 つのデータ値を表します。
- 複合データ型。単一のカラム位置にある複数のデータ値を表します。Spark エンジンで実行されるマッピングで複合データ型を使用して、複合ファイルの階層データを処理します。

データ統合サービスは、ソースデータを読み取るときに、ネイティブデータ型に対応するトランスフォーメーションデータ型に変換してから、データのトランスフォームを実行します。データ統合サービスは、ターゲットに書き込むときに、トランスフォーメーションデータ型に対応するネイティブデータ型に変換します。

非ネイティブ環境でのトランスフォーメーションデータ型のサポート

次の表に、非ネイティブランタイムエンジンでの Informatica トランスフォーメーションデータ型のサポートを示します。

トランスフォーメーションデータ型	エンジン
Array	- Spark - Databricks Spark
Bigint	- Blaze - Spark - Databricks Spark
Binary	- Blaze - Spark
Date/Time	- Blaze - Spark - Databricks Spark
Decimal	- Blaze - Spark - Databricks Spark
Double	- Blaze - Spark - Databricks Spark
Integer	- Blaze - Spark - Databricks Spark
Map	- Spark - Databricks Spark
String	- Blaze - Spark - Databricks Spark
Struct	- Spark - Databricks Spark
Text	- Blaze - Spark - Databricks Spark
timestampWithTZ	サポートされていません。

複合ファイルデータ型とトランスフォーメーションデータ型

マッピングで複合データ型を使用して複合ファイル内の階層データを処理できます。

Spark および Databricks Spark エンジンで実行されるマッピングでは次の複合ファイルで複合データ型を使用できます。

- Avro
- JavaScript Object Notation (JSON)
- Parquet

Avro データ型とトランスフォーメーションデータ型

Apache Avro データ型は、データ統合サービスがプラットフォーム間のデータ移動に使用するトランスフォーメーションデータ型に対応しています。

次の表で、Avro データ型とトランスフォーメーションデータ型を比較します。

Avro	トランスフォーメーション	説明
Array	Array	文字数に制限はありません。
Boolean	Integer	-2,147,483,648〜2,147,483,647 精度 10、位取り 0
Byte	Binary	1〜104,857,600 バイト
Double	ダブル	精度 15
固定長	Binary	1〜104,857,600 バイト
Float	Double	精度 15
Int	Integer	-2,147,483,648〜2,147,483,647 精度 10、位取り 0
Long	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 精度 19、位取り 0。
マップ	マップ	文字数に制限はありません。
Record	Struct	文字数に制限はありません。
String	String	1〜104,857,600 文字
Union	["primitive_type complex_type", "null"]または["null", "primitive_type complex_type"]の共有体で表される対応するデータ型。	プリミティブデータ型または複合データ型に依存します。

Avro Union データ型

共有体は、フィールドが複数のデータ型を持つ可能性があることを示します。例えば、共有体はフィールドが string または null であることを示す場合があります。共有体は、データ型を含む JSON 配列として表されません。

Developer tool は、["primitive_type|complex_type", "null"]または["null", "primitive_type|complex_type"]の共有体のみを解釈します。Avro データ型は、対応するトランスフォーメーションデータ型に変換されます。Developer tool は null を無視します。

サポートされていない Avro データ型

Developer tool でサポートされていない Avro のデータ型は以下のとおりです。

- enum
- null

JSON のデータ型とトランスフォーメーションデータ型

JavaScript Object Notation のデータ型は、データ統合サービスがプラットフォーム間のデータ移動に使用するトランスフォーメーションデータ型に対応します。

次の表で、JSON のデータ型とトランスフォーメーションデータ型を比較します。

JSON	トランスフォーメーション	説明
Array	Array	文字数に制限はありません。
Double	ダブル	精度 15
Integer	Integer	-2,147,483,648～2,147,483,647 精度 10、位取り 0
Object	Struct	文字数に制限はありません。
String	String	1～104,857,600 文字

サポートされていない JSON のデータ型

Developer tool でサポートされていない JSON のデータ型は以下のとおりです。

- date/timestamp
- enum
- union

Parquet のデータ型とトランスフォーメーションデータ型

Apache Parquet のデータ型は、データ統合サービスがプラットフォーム間のデータ移動に使用するトランスフォーメーションデータ型に対応しています。

次の表で、Parquet のデータ型とトランスフォーメーションデータ型を比較します。

Parquet	トランスフォーメーション	説明
Binary	Binary	1～104,857,600 バイト
Binary (UTF8)	String	1～104,857,600 文字
Boolean	Integer	-2,147,483,648～2,147,483,647 精度 10、位取り 0
Double	ダブル	精度 15
Fixed Length Byte Array	Decimal	宣言された精度と位取りを持つ 10 進型の値。位取りは、精度以下にする必要があります。
Float	Double	精度 15
group (LIST)	Array	文字数に制限はありません。
Int32	Integer	-2,147,483,648～2,147,483,647 精度 10、位取り 0
Int64	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 精度 19、位取り 0。
Int64 (TIMESTAMP_MILLIS)	Date/Time	西暦 0001 年 1 月 1 日～西暦 9999 年 12 月 31 日 精度 29、位取り 9 (精度はナノ秒まで) 日付値と時刻値が結合されたデータ型。
Int96	Date/Time	西暦 0001 年 1 月 1 日～西暦 9999 年 12 月 31 日 精度 29、位取り 9 (精度はナノ秒まで) 日付値と時刻値が結合されたデータ型。
マップ	マップ	文字数に制限はありません。
Struct	Struct	文字数に制限はありません。
Union	["primitive_type", "null"]または["null", "primitive_type"]の共有体で表される対応のプリミティブデータ型。	プリミティブデータ型に依存します。

Parquet Union データ型

共有体は、フィールドが複数のデータ型を持つ可能性があることを示します。例えば、共有体はフィールドが string または null であることを示す場合があります。共有体は、データ型を含む JSON 配列として表されます。

Developer tool は、["primitive_type", "null"] または ["null", "primitive_type"]. の共有体のみを解釈します。Parquet のデータ型は、対応するトランスフォーメーションデータ型に変換されます。Developer tool は null を無視します。

サポートされていない Parquet のデータ型

Developer tool でサポートされていない Parquet のデータ型は以下のとおりです。

- int96 (TIMESTAMP_MILLIS)
- date
- TIMESTAMP

Hive のデータ型とトランスフォーメーションデータ型

次の表に、データ統合サービスでサポートされる Hive のデータ型、およびそれに対応するトランスフォーメーションのデータ型を示します。

Hive のデータ型	トランスフォーメーションデータ型	範囲と内容
Binary	Binary	1~104,857,600 バイト。Hadoop 環境内の Binary データ型のデータを読み書きできます。バイナリデータ型の変換には、ユーザー定義関数を使用できます。
Tiny Int	Integer	-32,768~32,767
Integer	Integer	-2,147,483,648~2,147,483,647 精度 10、位取り 0
Bigint	Bigint	-9,223,372,036,854,775,808~9,223,372,036,854,775,807 精度 19、位取り 0
Decimal	Decimal	高精度のマッピングが無効な場合、データ統合サービスはすべての Decimal 値を Double 値に変換します。 高精度のマッピングが有効な場合、データ統合サービスは精度が 38 桁を超える Decimal 値を Double 値に変換します。
Double	Double	精度 15
Float	Double	精度 15

Hive のデータ型	トランスフォーメーションデータ型	範囲と内容
String	String	1～104,857,600 文字
Boolean	Integer	1 または 0 Boolean のデフォルトのトランスフォーメーション型は Integer です。これを値が True および False の String データ型に設定することもできます。
配列	String	1～104,857,600 文字
Struct	String	1～104,857,600 文字
マップ	String	1～104,857,600 文字
タイムスタンプ	datetime	タイムスタンプ形式は YYYY-MM-DD HH:MM:SS.ffffff です。精度 29、位取り 9。
日付	datetime	0000-01-01 から 9999-12-31。Hive の日付形式は YYYY-MM-DD です。精度 10、位取り 0。
Char	String	1～255 文字
Varchar	String	1～65355 文字

Hive の複合データ型

配列、マップ、および構造などの Hive の複合データ型は、プリミティブデータ型または複合データ型のコンポジットです。Informatica Developer は、文字列データ型を使用して複合データ型を表し、区切り文字を使用して複合データ型の要素を区切ります。

注: Hive ソースまたは Hive ターゲットに含まれる Hive の複合データ型は、Hadoop クラスタでマッピングを実行する場合はサポートされません。

次の表に、複合データ型を表すために使用するトランスフォーメーション型と区切り文字について説明します。

複合データ型	説明
Array	配列内の要素は文字列データ型である。配列内の要素はカンマで区切る。例えば、fruits の配列は、[apple,banana,orange]として表される。
Map	マップにはキーと値のペアが含まれ、=文字で区切った文字列と整数のペアとして表される。文字列と整数のペアはカンマで区切る。例えば、fruits のマップは、[1=apple,2=banana,3=orange]として表される。
Struct	構造は、:文字で区切られた文字列と整数のペアとして表される。文字列と整数のペアはカンマで区切る。例えば、fruits の構造は、[1,apple]として表される。

Sqoop のデータ型

Sqoop を使用する場合、処理にいくつかのバリエーションが生じます。Sqoop は、データベースベンダーがサポートするデータ型のサブセットをサポートします。

Aurora のデータ型

Informatica は、Sqoop の使用時に次の Aurora のデータ型をサポートしています。

- Binary
- Bit
- Blob（インポート用にのみサポート）
- Char
- Date
- Datetime
- Decimal
- Double
- Enum
- 浮動小数点数型
- 整数型
- Numeric
- Real
- Set
- テキスト
- 時間
- タイムスタンプ
- Varbinary
- Varchar

IBM DB2 および DB2 for z/OS のデータ型

Informatica は、Sqoop の使用時に次の IBM DB2 と DB2 for z/OS のデータ型をサポートしています。

- Bigint
- Blob（インポート用にのみサポート）
- Char
- Clob
- Date
- DBClob
- DecFloat（インポート用にのみサポート）
- Decimal
- Double（DB2 for z/OS 用にのみサポート）
- Float（DB2 用にのみサポート）

Graphic
Integer
LongVargraphic (DB2 用にのみサポート)
Numeric
Real
Smallint
Time
Timestamp
Varchar
Vargraphic
XML (インポート用にのみサポート)

Greenplum のデータ型

Informatica は、Sqoop の使用時に次の Greenplum のデータ型をサポートしています。

Bigint
Bigserial
Bytea
Date
Decimal
Double
Integer
Nchar
Numeric
Nvarchar
Real
Serial
Smallint
テキスト
時間
タイムスタンプ

Microsoft SQL Server データ型

Sqoop を使用するとき、次の Microsoft SQL Server のデータ型がサポートされます。

Bigint
Bit
Char
日時型
10 進型
Float

INT
Money
Numeric
Real
Smalldatetime
Smallint
Smallmoney
テキスト
時間
Tinyint
Varchar

Sqoop の Microsoft SQL Server データ型のルールとガイドライン

Sqoop マッピングに Microsoft SQL Server のデータ型を設定する場合は、次のルールとガイドラインを考慮します。

- ランタイムにターゲットテーブルを作成または置き換えて、Blaze または Spark エンジンでマッピングを実行して Bigint データをエクスポートすると、マッピングは失敗します。
- Sqoop マッピングを実行して time データをエクスポートすると、Sqoop はミリ秒をエクスポートしません。

Netezza のデータ型

Informatica は、Sqoop の使用時に次の Netezza のデータ型をサポートしています。

Bigint
Blob（インポート用にのみサポート）
Byteint
Char
Date
倍精度浮動小数点数型
Float4
Float8
Number
Timestamp
Varchar

Oracle のデータ型

Sqoop を使用するとき、次の Oracle データ型がサポートされます。

Blob（インポート用にのみサポート）
Char
Date
浮動小数点数型

長整数

Nchar (OraOop を設定している場合にサポート)

Nvarchar (OraOop を設定している場合にサポート)

Number (P,S)

Timestamp

Varchar

Varchar2

Sqoop の Oracle データ型のルールとガイドライン

Sqoop マッピングに Oracle のデータ型を設定する場合は、次のルールとガイドラインを考慮します。

- Blaze エンジンで Sqoop マッピングを実行して Oracle 浮動小数点データをエクスポートする場合、Sqoop はデータを切り詰めます。
- Blaze エンジンで Sqoop マッピングを実行し、ナノ秒単位の Oracle タイムスタンプデータをエクスポートすると、Sqoop はターゲットに 3 桁だけ書き込みます。
- OraOop を設定し、Spark エンジンで Sqoop マッピングを実行して Oracle タイムスタンプデータをエクスポートすると、Sqoop はターゲットに 3 桁だけ書き込みます。

Teradata のデータ型

Sqoop を使用するとき、次の Teradata データ型がサポートされます。

Bigint (インポート用にのみサポート)

Blob (インポート用にのみサポート)

Byteint

Char

Clob

Date

Decimal

Double

浮動小数点数型

Integer

Number

Numeric

Real

Smallint

Time

Timestamp

Varchar

注: Sqoop マッピングを実行して Time データ型をエクスポートする場合、Sqoop はミリ秒をエクスポートしません。

Sqoop 用の TDCH 専用コネクタを備えた Teradata のデータ型

Informatica は、Cloudera Connector Powered by Teradata、Hortonworks Connector for Teradata、および MapR Connector for Teradata with Sqoop を使用する場合に、次の Teradata データ型をサポートします。

Bigint

Byte (Hortonworks Connector for Teradata と MapR Connector for Teradata でのみサポート)

Byteint

文字

Date

Decimal

Double Precision/Float/Real

Integer

Number (P,S)

Numeric

Smallint

Time (Cloudera Connector Powered by Teradata と Hortonworks Connector for Teradata でのみサポート)

Timestamp

Varchar

Varbyte (Hortonworks Connector for Teradata と MapR Connector for Teradata でのみサポート)

Vertica のデータ型

Informatica は、Sqoop の使用時に次の Vertica のデータ型をサポートしています。

Boolean (Spark エンジンでのみサポート)

Bigint

Char

Date

Datetime

10 進型

Double precision

Float

Float (n)

Float8

Int

Int8

整数型

長整数型 Varchar

Money

Number

Numeric

Real

Smalldatetime

Smallint

Time

タイムスタンプ

Tinyint

Varchar

付録 C

関数リファレンス

この付録では、以下の項目について説明します。

- [非ネイティブ環境での関数サポート, 292 ページ](#)
- [関数とデータ型の処理, 294 ページ](#)

非ネイティブ環境での関数サポート

ネイティブ環境で有効な Informatica トランスフォーメーション言語の関数の一部は、Hadoop、Databricks などの非ネイティブ環境では制限があるか、サポートされないことがあります。この表に記載されていない関数は、すべてのエンジンで制限なくサポートされます。

重要: 非ネイティブ環境にマッピングをプッシュすると、マッピングを処理するエンジンは、データ統合サービスとは異なる一連のルールを使用します。その結果、マッピングの結果は、エンジンが使用するルールに基づいて異なる可能性があります。

次の表は、各関数と、非ネイティブ環境の各エンジンでの関数のサポートレベルを一覧表示しています。

関数	Blaze エンジン	Spark エンジン	Databricks Spark
ABORT	サポートされていません。	サポートされていません。	サポートされていません。
ANY	サポート	サポート	サポート
ARRAY	サポートされていません。	サポート	サポート
AVG	サポート	サポート	サポート
CAST	サポートされていません。	サポート	サポート
COLLECT_LIST	サポートされていません。	サポート	サポート
COLLECT_MAP	サポートされていません。	サポート	サポート
CONCAT_ARRAY	サポートされていません。	サポート	サポート

関数	Blaze エンジン	Spark エンジン	Databricks Spark
COUNT	サポート	サポート	サポート
CREATE_TIMESTAMP_TZ	サポート	サポートされていません。	サポートされていません。
CUME	サポートされていません。	サポートされていません。	サポートされていません。
DECOMPRESS	サポート	サポート	サポート
ERROR	サポートされていません。	サポートされていません。	サポートされていません。
FIRST	サポートされていません。	サポートされていません。	サポートされていません。
GET_TIMEZONE	サポート	サポートされていません。	サポートされていません。
GET_TIMESTAMP	サポート	サポートされていません。	サポートされていません。
LAST	サポートされていません。	サポートされていません。	サポートされていません。
LAG	サポートされていません。	サポート	サポート
LEAD	サポートされていません。	サポート	サポート
MAP	サポートされていません。	サポート	サポート
MAP_FROM_ARRAYS	サポートされていません。	サポート	サポート
MAP_KEYS	サポートされていません。	サポート	サポート
MAP_VALUES	サポートされていません。	サポート	サポート
MAX (Dates)	サポート	サポート	サポート
MAX (Numbers)	サポート	サポート	サポート
MAX (String)	サポート	サポート	サポート
METAPHONE	サポート	サポート	サポート
MIN (Dates)	サポート	サポート	サポート
MIN (Numbers)	サポート	サポート	サポート

関数	Blaze エンジン	Spark エンジン	Databricks Spark
MIN (String)	サポート	サポート	サポート
MOVINGAVG	サポートされていません。	サポートされていません。	サポートされていません。
MOVINGSUM	サポートされていません。	サポートされていません。	サポートされていません。
PERCENTILE	サポート	サポート	サポート
RESPEC	サポートされていません。	サポート	サポート
SIZE	サポートされていません。	サポート	サポート
STDDEV	サポート	サポート	サポート
STRUCT	サポートされていません。	サポート	サポート
STRUCT_AS	サポートされていません。	サポート	サポート
SUM	サポート	サポート	サポート
SYSTIMESTAMP	サポート	サポート	サポート
TO_DATE	サポート	サポート	サポート
TO_DECIMAL	サポート	サポート	サポート
TO_DECIMAL38	サポート	サポート	サポート
TO_TIMESTAMP_TZ	サポート	サポートされていません。	サポートされていません。
UUID4	サポート	サポート	サポート
UUID_UNPARSE	サポート	サポート	サポート
VARIANCE	サポート	サポート	サポート

詳細については、『Informatica トランスフォーメーション言語リファレンス』を参照してください。

関数とデータ型の処理

Hadoop 環境でマッピングを実行するときに、Hadoop エンジンは、データ統合サービスと異なる方法で Informatica 関数とデータ型を処理する場合があります。また、Informatica 関数とデータ型の処理は、Hadoop 環境の実行時エンジンごとに異なる可能性があります。関数およびデータ型の処理と有効性に違いがあるために、マッピングの結果が異なる可能性があります。

Spark エンジン処理のルールとガイドライン

Spark エンジンでの Informatica の関数の処理には、いくつかの制限とガイドラインがあります。

重要: Hadoop 環境にマッピングをプッシュすると、マッピングを処理するエンジンは、データ統合サービスとは異なる一連のルールを使用します。その結果、マッピングの結果は、エンジンが使用するルールに基づいて異なる可能性があります。このトピックでは、Informatica が内部テストと使用の中で発見したいいくつかの処理の違いについて説明します。Informatica がサードパーティのエンジンのすべてのルールをテストすることはなく、相違点を網羅したリストを提供することはできません。

Spark エンジンでの関数とデータ型の処理の際は、次のルールとガイドラインを考慮します。

- Spark エンジンとデータ統合サービスでは、オーバーフロー値を異なる方法で処理します。Spark エンジンの処理ルールは、データ統合サービスが使用するルールとは異なる可能性があります。その結果、Spark エンジンがオーバーフローを処理するときは、マッピングの結果がネイティブ環境と Hadoop 環境とで異なる可能性があります。Spark に関して次のように処理の相違点があることを考慮します。
 - 式の結果が、ゼロによる除算や負の数値の SQRT などの、数値エラーになった場合は、Spark エンジンは NULL を返します。ネイティブ環境では、同じ式の結果は行エラーになります。
- Spark エンジンとデータ統合サービスでは、データ型の変換を異なる方法で処理します。その結果、Spark エンジンがデータ型変換を実行するときは、マッピングの結果がネイティブ環境と Hadoop 環境とで異なる可能性があります。Spark に関して次のように処理の相違点があることを考慮します。
 - Decimal などの浮動小数点型の計算処理結果は、ネイティブ環境と Hadoop 環境との間で異なる可能性があります。結果の違いは、複数の操作間で大きくなる可能性があります。
 - 高精度モードで実行するマッピングで TO_DECIMAL または TO_DECIMAL38 関数を使用するときは、スケール引数を指定する必要があります。マッピングを低精度モードで実行する場合、Spark エンジンはスケール引数を無視し、double を返します。
 - double または decimal の値の小数桁数が小数位ポートで設定されているスケールを超えると、Spark エンジンは後続の桁を切り捨て、必要な場合は値を丸めます。
 - Hive 2.3 を使用する場合、Spark エンジンは位取りの値を保証します。
例えば、Spark エンジンが Hive 2.3 を使用して 9 桁の位取りの 10 進数の 1.1234567 を処理すると、出力は 1.123456700 になります。Hive 2.3 を使用しない場合、出力は 1.1234567 です。
 - Spark エンジンはナノ秒までの日付を処理できません。ミリ秒までの精度で日時データを返すことができます。
- データオーバーフローが発生した場合、Spark エンジンは NULL を返します。例えば、精度が 3、スケールが 2 のポートで式 TO_DECIMAL(12.34, 2) を使用した場合、戻り値は NULL です。NULL 値はマッピングを介してプロパゲートされます。マッピングはデフォルト値を使用してその値を上書きし、関数 IS_NULL を使用して検出して、ターゲットに NULL 値を書き込みます。
- ストリーミングマッピングで高精度を有効にしている場合でも、Spark エンジンはマッピングを低精度モードで実行します。
- マッピングに decimal データ型の要素を含む複合ポートが含まれている場合、Spark エンジンはマッピングを低精度モードで実行します。
- Hadoop 環境は「/n」の値を NULL 値として扱います。集計関数に空の値または NULL 値が含まれている場合、Hadoop 環境で集計の計算中にこれらの値が含まれます。
- マッピング検証は、ポート名などの変数値を使用して SYSTIMESTAMP を設定した場合に失敗します。この関数は、引数を含めないか、タイムスタンプ値を取得する精度を含めるかのいずれかです。
- マッピング検証は、タイムゾーンデータ型のタイムスタンプが出力ポートに含まれる場合に失敗します。
- アグリゲータトランスフォーメーション内に、単一またはネストされた関数を含めないでください。データ統合サービスは、ネイティブ環境でのマッピングに失敗します。処理を Hadoop 環境にプッシュすることはできませんが、予期しない結果になる可能性があります。Informatica は、集計を実行するには、複数のトランスフォーメーションを作成することをお勧めします。

- ウィンドウ化用に設定されているトランスフォーメーションのデータはプレビューできません。
- Spark METAPHONE 関数は、org.apache.commons.codec.language ライブラリからの音声エンコーダを使用します。Spark エンジンがマッピングを実行するときに、METAPHONE 関数はネイティブ環境での出力とは異なる出力を生成する可能性があります。次の表に、いくつかの例を示します。

文字列	データ統合サービス	Spark エンジン
Might	MFT	MT
HIGH	HF	H

- ISO 標準形式で記述された文字列を処理するために、Spark エンジンで TO_DATE 関数を使用する場合、*T*を日付文字列に追加し、*"T"*をフォーマット文字列に追加する必要があります。次の式に、TO_DATE 関数を使用して ISO 標準形式で記述された文字列 YYYY-MM-DDTHH24:MI:SS を変換する例を示します。
`TO_DATE('2017-11-03*T*12:45:00','YYYY-MM-DD*"T"*HH24:MI:SS')`
 次の表に、この関数での文字列の変換方法を示します。

ISO Standard Format	RETURN VALUE
2017-11-03T12:45:00	Nov 03 2017 12:45:00

- UUID4 関数は、UUID_UNPARSE または ENC_BASE64 内の引数として使用される場合にのみサポートされます。
- UUID_UNPARSE 関数は、引数が UUID4() の場合にのみサポートされます。

索引

A

ADLS

- 複合ファイルソース [61](#), [63](#)
- 複合ファイルターゲット [74](#), [76](#)

Amazon AWS [234](#)

Amazon Redshift 接続

- プロパティ [239](#)

Amazon S3

- 複合ファイルソース [62](#)
- 複合ファイルターゲット [76](#)

array

- 1 次元 [159](#)
- インデックス [159](#)
- 形式 [159](#)
- 次元 [159](#)
- 多次元 [159](#)
- 複合データ型 [159](#)
- 例 [159](#)

Azure

- 設定 [236](#)

B

Big Data

- アプリケーションサービス [16](#)
- リポジトリ [17](#)

Blaze エンジン

- Blaze エンジンのアーキテクチャ [20](#)
- サマリレポート [144](#), [145](#), [147](#)
- セグメントの時間 [144](#)
- タスクレットの実行時間 [145](#)
- タスクレット情報 [147](#)
- マッピングプロパティ [145](#)
- 監視 [141](#), [143](#), [144](#)
- 接続プロパティ [249](#)

Blaze エンジンで実行するプロファイル

- 概要 [125](#)

Blaze ジョブ監視

- ロギング [148](#)

Blaze 実行プラン

- 監視 [138](#)

D

Databricks

- クラウドプロビジョニング設定 [239](#)

Databricks Spark

- 実行プラン [44](#)

Databricks の統合

- 概要 [19](#)

Databricks 環境

- フラットファイルソース [61](#)
- フラットファイルターゲット [74](#)

Databricks 環境 (続く)

- ソース [60](#)
- 複合ファイルターゲット [73](#)

E

Expression トランスフォーメーション

- Blaze エンジン [95](#)
- Databricks Spark エンジン [95](#)
- Spark エンジン [95](#)

H

Hadoop [233](#)

Hadoop でのマッピング実行

- ロギング [137](#)
- 監視 [138](#)

Hadoop で実行するプロファイル

- 監視 [132](#)

Hadoop のユーティリティ

- Sqoop [18](#)

Hadoop 環境

- Hive ターゲット [80](#)
- Sqoop ソースの制限事項 [70](#)
- Sqoop ターゲットの制限事項 [84](#)
- ファイルソース [62](#)
- ファイルターゲット [75](#)
- フラットファイルターゲット [79](#)
- フラットファイルの制限 [65](#)
- リレーショナルソース [66](#)
- ログ [135](#)
- 最適化 [45](#)

Hadoop 接続

- 作成 [271](#)

HBase 接続

- MapR-DB プロパティ [256](#)
- プロパティ [256](#)

HDFS

- 複合ファイルソース [65](#)
- 複合ファイルターゲット [78](#)

HDFS マッピング

- データ抽出の例 [53](#)
- 説明 [52](#)

HDFS 接続

- プロパティ [254](#)
- 作成 [271](#)

Hive

- ターゲットの制限事項 [80](#)

Hive クエリプラン

- 表示、プロファイル [132](#)
- 表示、マッピング [44](#)

Hive ソース

- Blaze エンジン [68](#)
- Informatica マッピングとの連携 [67](#)

Hive ソース (続く)

 postSQL [67](#)

 preSQL [67](#)

Hive ターゲット

 Blaze エンジン [83](#)

 postSQL [81](#)

 preSQL [81](#)

 切り詰め [82](#)

Hive で実行するプロファイル

 概要 [125](#)

Hive プッシュダウン

 接続プロパティ [249](#)

Hive マッピング

 説明 [54](#)

Hive メタストア [121](#)

Hive メタストアとしての外部 RDS [121](#)

Hive 実行プラン

 監視 [138](#)

Hive 接続

 プロパティ [257](#)

 作成 [271](#)

I

Informatica Big Data Management

 概要 [15](#)

Informatica Intelligent Cloud Services アカウント

 作成 [214](#)

Informatica エンジン

 Informatica エンジンの実行プラン [42](#)

J

Java トランスフォーメーション

 Blaze エンジン [96](#)

 Spark エンジン [97](#)

 非ネイティブ環境 [96](#)

JDBC 接続

 プロパティ [260](#)

M

map

 複合データ型 [161](#)

 キーと値のペア [161](#)

 形式 [161](#)

 例 [161](#)

MapR-FS

 複合ファイルソース [65](#)

 複合ファイルターゲット [78](#)

Microsoft Azure [236](#)

Microsoft Azure Data Lake Store 接続

 プロパティ [267](#)

Microsoft Azure SQL Data Warehouse 接続

 プロパティ [268](#)

P

postSQL

 Hive ソース [67](#)

 Hive ターゲット [81](#)

preSQL

 Hive ソース [67](#)

 Hive ターゲット [81](#)

Python トランスフォーメーション

 Spark エンジン [103](#)

 非ネイティブ環境 [103](#)

R

Rank トランスフォーメーション

 Blaze エンジン [103](#)

 Databricks Spark エンジン [104](#)

S

Sequence Generator トランスフォーメーション

 Blaze エンジン [105](#)

 Spark エンジン [105](#)

Snowflake 接続

 プロパティ [269](#)

Sorter トランスフォーメーション

 Databricks Spark エンジン [107](#)

Spark

 実行プラン [43](#)

Spark HDFS ステージングディレクトリ

 Hadoop 接続プロパティ [249](#)

Spark イベントログディレクトリ

 Hadoop 接続プロパティ [249](#)

Spark エンジン

 データ型の処理 [295](#)

 ルールおよびガイドライン [295](#)

 関数の処理 [295](#)

 階層データ [155](#)

 監視 [149](#)

 接続プロパティ [249](#)

Spark デプロイモード

 Hadoop 接続プロパティ [249](#)

Spark 実行パラメータ

 Hadoop 接続プロパティ [249](#)

Sqoop ソース

 Hadoop 環境内 [70](#)

 Vertica からのデータ読み取り [70](#)

Sqoop ターゲット

 Hadoop 環境内 [84](#)

Sqoop のデータ型

 Aurora [286](#)

 Greenplum [287](#)

 IBM DB2 [286](#)

 IBM DB2 for z/OS [286](#)

 Microsoft SQL Server [287](#)

 Netezza [288](#)

 Oracle [288](#)

 TDCH の Sqoop 専用コネクタを備えた Teradata のデータ型 [290](#)

 Teradata [289](#)

 Vertica [290](#)

Sqoop のマッピング引数

 batch [35](#)

 infooptimize [36](#)

 infoownername [36](#)

 m [34](#)

 num-mappers [34](#)

 split-by [35](#)

 verbose [37](#)

 スキーマ [36](#)

Sqoop マッピング

 サポートされるエンジン [34](#)

 概要 [34](#)

Sqoop 引数

 パラメータ化 [38](#)

Sqoop 差分データ抽出
--infa-incremental-key [37](#)
--infa-incremental-type [37](#)
--infa-incremental-value [37](#)
--infa-incremental-value-format [37](#)
構成 [37](#)
Sqoop 接続
サポートされるデータ型 [286](#)
Sqoop 接続の引数
-Dsqaop.connection.factories [263](#)
ドライバ [263](#)
接続 [263](#)
直接 [263](#)
Sqoop 設定
プロファイリング [126](#)
マッピングプロパティ [38](#)
struct
形式 [161](#)
スキーマ [161](#)
名前と型のペア [161](#)
複合データ型 [161](#)
例 [161](#)

T

TDCH 接続ファクトリ
-Dsqaop.connection.factories [263](#)

U

URL の監視
Blaze ジョブと Spark ジョブ [137](#)

Y

YARN
キュー [48](#)
スケジューラ
キャパシティ [48](#)
フェア [48](#)
YARN Web ユーザーインタフェース
説明 [136](#)

あ

アーキテクチャ
Big Data Management [16](#)
Big Data Management と Databricks [22](#)
Hadoop 環境 [18](#)
アグリゲータトランスフォーメーション
Blaze エンジン [89](#)
Databricks Spark エンジン [90](#)
Spark エンジン [90](#)
非ネイティブ環境 [89](#)
アップデートストラテジトランスフォーメーション
Blaze エンジン [108](#)
Spark エンジン [110](#)
非ネイティブ環境 [108](#)
アドレスバリデータトランスフォーメーション
非ネイティブ環境 [88](#)
アマゾン S3 接続
プロパティ [241](#)

い

インテリジェント構造
ユースケース [209](#)
インテリジェント構造マッピング
データ変換の例 [213](#)
インテリジェント構造モデル
エクスポート [215](#)
概要 [208](#)
作成 [215](#)
インテリジェント構造を使用した処理
Spark での実行 [210](#)
マッピング内 [210](#)
インテリジェント構造検出
プロセス [209](#)
ライセンス [211](#)

う

ウィンドウ化プロパティ
順序 [217](#), [218](#)
パーティション [217](#), [218](#)
フレーム [217](#), [223](#)
ルールおよびガイドライン [220](#)
ウィンドウ化
プロパティ [217](#)
概要 [216](#)
例 [225](#), [227](#), [229](#)
ウィンドウ関数
LAG [221](#)
LEAD [221](#)
ルールおよびガイドライン [225](#)
概要 [220](#)
集計関数 [221](#)

え

エフェメラルクラスタ
概要 [112](#)
エンタープライズ検出
Informatica Analyst での実行 [130](#)

か

カラムプロファイルの作成
プロファイル [129](#)

き

キージェネレータトランスフォーメーション
非ネイティブ環境 [99](#)

く

クラウドプラットフォームクラスタ [112](#)
クラウドプロビジョニング設定
Databricks のプロパティ [239](#)
Amazon AWS のプロパティ [234](#)
Microsoft Azure プロパティ [236](#)
クラスタの作成タスク
プロパティ [115](#)
クラスタの削除タスク [123](#)

クラスタワークフロー

Azure ジョブ監視 [124](#)

Blaze エンジンでのマッピングの実行 [120](#)

Hive メタストアデータベースとしての外部 RDS [121](#)

クラウドプロビジョニング接続 [233](#)

クラスタの削除タスク [123](#)

クラスタワークフロー内のマッピング [123](#)

コンポーネント [113](#), [123](#)

サンプル [113](#)

ワークフロータスク [123](#)

概要 [112](#)

監視 [124](#)

実行 [124](#)

クラスタワークフロージョブの監視 [124](#)

グリッド

説明 [56](#)

こ

コンポーネントアーキテクチャ

クライアントおよびツール [16](#)

さ

サードパーティツール

Hadoop クラスタ [18](#)

し

シーケンスジェネレータトランスフォーメーション

非ネイティブ環境 [105](#)

ジョイナトランスフォーメーション

Blaze エンジン [98](#)

Databricks Spark エンジン [99](#)

Spark エンジン [99](#)

非ネイティブ環境 [98](#)

す

ステートフルコンピューティング

概要 [216](#)

そ

ソーシャルメディアマッピング

説明 [55](#)

ソース

非ネイティブ環境で [59](#)

Databricks 環境 [60](#)

ソースファイル名

フラットファイルのソースの場合 [66](#)

ソータートランスフォーメーション

Blaze エンジン [106](#)

Spark エンジン [106](#)

非ネイティブ環境 [105](#)

た

ターゲット

Hadoop マッピング内のフラットファイル [79](#)

タイプ設定

マップ [169](#)

タイプ設定 (続く)

構造 [169](#)

構造ポート [173](#)

配列 [169](#)

て

ディシジョントランスフォーメーション

非ネイティブ環境 [95](#)

データオブジェクト

インテリジェント構造を使用したデータの処理 [210](#)

データオブジェクトプロファイル

エンタープライズ検出 [128](#)

単一のプロファイルの作成 [127](#)

データプロセッサトランスフォーメーション

Blaze エンジン [94](#)

非ネイティブ環境 [94](#)

データマスキングトランスフォーメーション

非ネイティブ環境 [92](#)

データ型

Hadoop 環境での処理 [294](#)

Hive [284](#)

Hive の複合データ型 [285](#)

サポート [280](#)

ネストされたデータ型 [158](#)

プリミティブデータ型 [158](#)

複合データ型 [158](#)

複合ファイル [281](#)

と

ドット演算子

構造要素の抽出 [175](#)

トランスフォーメーション

非ネイティブ環境で [86](#)

ね

ネイティブ環境

パーティション化 [56](#)

マッピング [52](#)

高可用性 [57](#)

最適化 [56](#)

ネストされたデータ型定義

例 [167](#)

ネストされた構造

生成 [184](#)

の

ノードラベル適用

Blaze [49](#)

ノーマライザトランスフォーメーション

非ネイティブ環境 [102](#)

は

パーサトランスフォーメーション

非ネイティブ環境 [102](#)

パーティション化

最適化 [57](#)

説明 [56](#)

ひ

ビッグデータ
 ビッグデータ処理 [23](#)
ビッグデータ処理
 データの収集 [23](#)

ふ

ファイルソース
 Hadoop 環境 [62](#)
ファイルターゲット
 Hadoop 環境 [75](#)
フィルタトランスフォーメーション
 Blaze エンジン [96](#)
 非ネイティブ環境 [96](#)
フラットファイルソース
 Databricks 環境で [61](#)
 Hadoop 環境内 [65](#)
フラットファイルターゲット
 Databricks 環境で [74](#)
プロパティ [242](#), [245](#)–[248](#)
プロファイル
 カラムプロファイルの作成 [129](#)

ま

マージトランスフォーメーション
 非ネイティブ環境 [102](#)
マッピング
 インテリジェント構造 [210](#)
マッピングの設定
 非ネイティブ [41](#)
マッピングの例
 Hive [54](#)
 Twitter [55](#)
マッピング実行プラン
 概要 [41](#)
マップ
 生成 [176](#)
 要素へのアクセス [174](#)
マップポート
 タイプ設定 [172](#)
マップ関数
 COLLECT_MAP [176](#)
 MAP [176](#)
 MAP_FROM_ARRAYS [176](#)
 MAP_KEYS [176](#)
 MAP_VALUES [176](#)
 SIZE [176](#)

ゆ

ユーティリティ
 Hadoop クラスタ [18](#)

ら

ラベラトランスフォーメーション
 非ネイティブ環境 [99](#)
ランクトランスフォーメーション
 Spark エンジン [104](#)

ランクトランスフォーメーション (続く)
 非ネイティブ環境 [103](#)
ランタイムプロパティ
 非ネイティブマッピング [28](#)

り

リレーショナルから階層型 [181](#)

る

ルータートランスフォーメーション
 非ネイティブ環境 [104](#)
ルールおよびガイドライン
 Spark エンジン [295](#)
 ウィンドウ化プロパティ [220](#)
 ウィンドウ関数 [225](#)
ルックアップトランスフォーメーション
 Blaze エンジン [100](#)
 Databricks Spark エンジン [101](#)
 Spark エンジン [100](#)
 非ネイティブ環境 [100](#)

れ

例
 array [159](#)
 map [161](#)
 struct [161](#)
 ウィンドウ化 [225](#), [227](#)
 ドット演算子 [174](#)
 ネストされたデータ型定義 [167](#)
 パーティションキーおよびオーダーキー [218](#)
 集計ウィンドウ関数 [221](#)
 添字演算子 [174](#)
 複合データ型定義 [165](#)
 複合演算子 [174](#)

ろ

ロギング
 Hadoop でのマッピング実行 [148](#)
 Spark エンジン [153](#)
ログ
 Blaze エンジン [147](#)
 Hadoop 環境 [135](#)
 Spark エンジン [153](#)
ログ URL
 YARN Web ユーザーインタフェース [153](#)

わ

ワークフロー
 非ネイティブマッピング [40](#)