Informatica®

Informatica® Mass Ingestion
April 2024

一括取り込みデータベース

Informatica Mass Ingestion 一括取り込みデータベース
April 2024

発行日: 2024-05-20

目次

序文 7

Informatica のリソース 7

- Informatica マニュアル 7
- Informatica Intelligent Cloud Services Web サイト 7
- Informatica Intelligent Cloud Services コミュニティ 7
- Informatica Intelligent Cloud Services マーケットプレース 8
- データ統合のコネクタのドキュメント 8
- Informatica ナレッジベース 8
- Informatica Intelligent Cloud Services Trust Center 8
- Informatica グローバルカスタマサポート 8

第 1 章 : 一括取り込みデータベース 9

使用例 9

サポートされているソースタイプとターゲットタイプ 10

一括取り込みデータベースアーキテクチャ 12

一括取り込みデータベースのシステム要件 14

一般的な制限とガイドライン 14

一括取り込みデータベースソース - 準備と使用 15

- DB2 for i ソース 15
- DB2 for LUW ソース 18
- DB2 for z/OS ソース 19
- Microsoft SQL Server、RDS for SQL Server、Azure SQL Database、Azure Managed Instance ソース 27
- MongoDB ソース 34
- MySQL ソース 35
- Netezza ソース 37
- Oracle ソース 38
- PostgreSQL ソース 58
- SAP HANA および SAP HANA Cloud ソース 62
- Teradata ソース 65

一括取り込みデータベースターゲット - 準備と使用 66

- Amazon Redshift ターゲット 66
- Amazon S3、フラットファイル、Google Cloud Storage、Microsoft Azure Data Lake Storage、Microsoft Fabric OneLake、および Oracle Cloud Object Storage ターゲット 66
- Databricks Delta ターゲット 68
- Google BigQuery ターゲット 70
- Kafka ターゲットと Kafka 対応 Azure Event Hubs ターゲット 73
- Microsoft Azure Synapse Analytics ターゲット 76
- Microsoft SQL Server および Azure SQL データベースのターゲット 77
- Oracle ターゲット 78

PostgreSQL ターゲット. 80
Snowflake ターゲット. 81
Amazon S3、Google Cloud Storage、および Azure Data Lake Storage Gen2 ターゲット上の CDC ファイルのデフォルトのディレクトリ構造. 84
Amazon S3、Google Cloud Storage、フラットファイル、Microsoft Fabric OneLake、および ADLS Gen2 ターゲット上の出力ファイルのカスタムディレクトリ構造. 87
サポートされている Avro データ型. 96
スキーマドリフト処理. 96
すべてのソーステーブル DML 変更の監査履歴のターゲットテーブルへの適用 98
ターゲットに対する削除を論理削除として適用する機能. 100
データベース統合タスクの設定. 100
始める前に. 100
基本的なタスク情報の定義. 101
ソースの設定. 102
ターゲットの設定. 115
スケジュールとランタイムオプションの設定. 157
データベース取り込みタスクのデプロイ. 162
データベース取り込みジョブの実行. 162
デプロイ済みデータベース統合ジョブの実行. 163
スケジュールに基づいた初期ロードジョブの実行. 163
データベース取り込みジョブの管理. 163
データベース統合ジョブの停止. 163
データベース統合ジョブの強制終了. 164
データベース統合ジョブの再開. 164
データベース取り込みジョブの再開時のスキーマドリフトオプションのオーバーライド. 165
データベース取り込みジョブの再デプロイ. 166
データベース統合ジョブのデプロイ解除. 167
ソースオブジェクトとターゲットオブジェクトの再同期. 168
増分変更データ処理のための再開とリカバリ. 169
デフォルトデータ型のマッピング. 169
DB2 for i ソースと Amazon Redshift ターゲット. 169
DB2 for i ソースと Databricks Delta ターゲット. 171
DB2 for i ソースと Google BigQuery ターゲット. 172
DB2 for i ソースと Microsoft Azure Synapse Analytics ターゲット. 178
Db2 for i ソースと Microsoft SQL Server ターゲット. 180
DB2 for i ソースと Oracle ターゲット. 181
DB2 for i ソースと PostgreSQL ターゲット. 183
DB2 for i ソースと Snowflake ターゲット. 184
DB2 for LUW ソースと Amazon Redshift ターゲット. 185
DB2 for LUW ソースと Databricks Delta ターゲット. 187
DB2 for LUW ソースと Google BigQuery ターゲット. 188
DB2 for LUW ソースと Microsoft Azure Synapse Analytics ターゲット. 190
Db2 for LUW ソースと Microsoft SQL Server ターゲット. 192

DB2 for LUW ソースと Oracle ターゲット. 193
DB2 for LUW ソースと Snowflake ターゲット. 194
DB2 for z/OS ソースと Amazon Redshift ターゲット. 196
DB2 for z/OS ソースと Databricks Delta ターゲット. 197
DB2 for z/OS ソースと Google BigQuery ターゲット. 198
DB2 for z/OS ソースと Microsoft Azure Synapse Analytics ターゲット. 201
DB2 for z/OS ソースと Oracle ターゲット. 202
DB2 for z/OS ソースと Snowflake ターゲット. 203
Microsoft SQL Server または Azure SQL Database ソースと Amazon Redshift ターゲット. . . 205
Microsoft SQL Server ソースと Databricks Delta ターゲット. 206
Microsoft SQL Server ソースと Google BigQuery ターゲット. 208
Microsoft SQL Server ソースと Parquet 出力形式を使用するターゲット. 213
Microsoft SQL Server または Azure SQL Database ソースと Microsoft Azure Synapse Analytics ターゲット. 214
Microsoft SQL Server ソースと Microsoft SQL Server ターゲット. 216
Microsoft SQL Server ソースと Oracle ターゲット. 218
Microsoft SQL Server または Azure SQL Database ソースと Snowflake ターゲット. 220
MongoDB ソースと Amazon Redshift ターゲット. 221
MongoDB ソースと Databricks Delta ターゲット. 222
MongoDB ソースと Google BigQuery ターゲット. 222
MongoDB ソースと Microsoft Azure Synapse Analytics ターゲット. 222
MongoDB ソースと Microsoft SQL Server ターゲット. 223
MongoDB ソースと Oracle ターゲット. 223
MongoDB ソースと Snowflake ターゲット. 223
MySQL ソースと Amazon Redshift ターゲット. 224
MySQL ソースと Databricks Delta ターゲット. 226
MySQL ソースと Google BigQuery ターゲット. 228
MySQL ソースと Microsoft Azure Synapse Analytics ターゲット. 232
MySQL ソースと Microsoft SQL Server ターゲット. 234
MySQL ソースと Oracle ターゲット. 237
MySQL ソースと Snowflake ターゲット. 239
Netezza ソースと Amazon Redshift ターゲット. 241
Netezza ソースと Databricks Delta ターゲット. 242
Netezza ソースと Google BigQuery ターゲット. 243
Netezza ソースと Microsoft Azure Synapse Analytics ターゲット. 245
Netezza ソースと Microsoft SQL Server ターゲット. 246
Netezza ソースと Oracle ターゲット. 247
Netezza ソースと Snowflake ターゲット. 249
Oracle ソースと Amazon Redshift ターゲット. 250
Oracle ソースと Databricks Delta ターゲット. 253
Oracle ソースと Google BigQuery ターゲット. 254
Oracle ソースと Microsoft Azure Synapse Analytics ターゲット. 258
Oracle ソースと Microsoft SQL Server ターゲット. 259

Oracle ソースと Oracle ターゲット. 261
Oracle ソースと PostgreSQL ターゲット. 262
Oracle ソースと Snowflake ターゲット. 264
PostgreSQL ソースと Amazon Redshift ターゲット. 265
PostgreSQL ソースと Databricks Delta ターゲット. 268
PostgreSQL ソースと Google BigQuery ターゲット. 270
PostgreSQL ソースと Microsoft Azure Synapse Analytics ターゲット. 274
PostgreSQL ソースと Oracle ターゲット. 277
PostgreSQL ソースと Snowflake ターゲット. 279
SAP HANA ソースと Amazon Redshift ターゲット. 282
SAP HANA ソースと Databricks Delta ターゲット. 284
SAP HANA ソースと Google BigQuery ターゲット. 285
SAP HANA ソースと Microsoft Azure Synapse Analytics ターゲット. 288
SAP HANA ソースと Microsoft SQL Server ターゲット. 290
SAP HANA ソースと Oracle ターゲット. 291
SAP HANA ソースと Snowflake ターゲット. 293
Teradata ソースと Amazon Redshift ターゲット. 295
Teradata ソースと Databricks Delta ターゲット. 297
Teradata ソースと Google BigQuery ターゲット. 299
Teradata ソースと Microsoft Azure Synapse Analytics ターゲット. 304
Teradata ソースと Oracle ターゲット. 306
Teradata ソースと Snowflake ターゲット. 309

索引. 312

序文

*一括取り込み一括取り込みデータベース*データベース取り込みデータベース統合一括取り込み」をお読みください。

このパブリケーションでは、サポートされているソースとターゲット、ユースケース、ソースの準備、制限事項と使用に関する考慮事項、タスクのデプロイ、データベース取り込みジョブの実行と管理についても取り上げています。

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 マニュアル

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

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

Informatica Intelligent Cloud Services Web サイト

Informatica Intelligent Cloud Services Web サイト（http://www.informatica.com/cloud）にアクセスできます。このサイトには、Informatica Cloud 統合サービスに関する情報が含まれます。

Informatica Intelligent Cloud Services コミュニティ

Informatica Intelligent Cloud Services コミュニティを使用して、技術的な問題について議論し、解決します。また、技術的なヒント、マニュアルの更新情報、FAQ（よくある質問）への答えを得ることもできます。

次の Informatica Intelligent Cloud Services コミュニティにアクセスします。

https://network.informatica.com/community/informatica-network/products/cloud-integration

開発者は、次の Cloud 開発者コミュニティで詳細情報を確認したり、ヒントを共有したりできます。

https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers

Informatica Intelligent Cloud Services マーケットプレース

Informatica マーケットプレイスにアクセスすると、データ統合コネクタ、テンプレート、およびマップレットを試用したり購入したりできます。

https://marketplace.informatica.com/

データ統合のコネクタのドキュメント

データ統合のコネクタのドキュメントには、マニュアルポータルからアクセスできます。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 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 Intelligent Cloud Services Trust Center

Informatica Intelligent Cloud Services Trust Center は、Informatica のセキュリティポリシーおよびリアルタイムでのシステムの可用性について情報を提供します。

Trust Center（https://www.informatica.com/trust-center.html）にアクセスします。

Informatica Intelligent Cloud Services Trust Center にサブスクライブして、アップグレード、メンテナンス、およびインシデントの通知を受信します。Informatica Intelligent Cloud Services Status ページには、すべての Informatica Cloud 製品の実稼働ステータスが表示されます。メンテナンスの更新はすべてこのページに送信され、停止中は最新の情報が表示されます。更新と停止の通知がされるようにするには、Informatica Intelligent Cloud Services の 1 つのコンポーネントまたはすべてのコンポーネントについて更新の受信をサブスクライブします。すべてのコンポーネントにサブスクライブするのが、更新を逃さないようにするための最良の方法です。

サブスクライブするには、Informatica Intelligent Cloud Services Status ページで **[サブスクライブして更新]** をクリックします。電子メール、SMS テキストメッセージ、Webhook、RSS フィード、またはこの 4 つの任意に組み合わせとして送信される通知を受信するという選択ができます。

Informatica グローバルカスタマサポート

グローバルサポートセンターには、Informatica Network または電話でお問い合わせください。

Informatica Network でオンラインサポートリソースを検索するには、Informatica Intelligent Cloud Services のヘルプメニューで **[サポートにお問い合わせください]** をクリックして、**Cloud Support** ページに移動します。**Cloud Support** ページには、システムステータス情報とコミュニティディスカッションが記載されています。追加のリソースを検索する場合や電子メールで Informatica グローバルカスタマサポートに問い合わせる場合は、Informatica Network にログインし、**[サポートが必要な場合]** をクリックしてください。

Informatica グローバルカスタマサポートの電話番号は、Informatica の Web サイト https://www.informatica.com/services-and-training/support-services/contact-us.html に掲載されています。

第 1 章

一括取り込みデータベース

一括取り込みデータベースは、一般的なリレーショナルデータベースから大規模なデータを取り込み、クラウドベースのターゲットやビッグデータを処理できるターゲットなど、複数のタイプのターゲットにデータをプロパゲートできます。一括取り込みデータベースは、Informatica Intelligent Cloud Services 一括取り込みサービスの個別にライセンスされた機能です。

一括取り込みサービスは、データベース統合タスクの設定とデプロイ、および取り込みジョブの実行と監視を行うための使いやすいインタフェースを提供します。ジョブは、取り込みタスクの実行可能インスタンスです。

一括取り込みデータベースは、次のタイプのロード操作を実行できます。

- *初期ロード*。ある時点で読み取られたソースデータをターゲットにロードします。データがロードされると、取り込みジョブは終了します。このロードタイプを使用して、増分変更が後で送信されるターゲットをマテリアライズしたり、オンプレミスデータベースシステムからクラウドベースのシステムに移行したり、データレイクまたはデータウェアハウスにデータを追加したりできます。
- *増分ロード*。データの変更を継続的に、またはジョブが停止または終了するまでプロパゲートします。ジョブは、最後に実行されてから、または特定の開始点から発生した変更をプロパゲートします。このロードタイプを使用して、個別のレポートシステムと分析システムのデータを最新の状態に保つことができるため、最新のデータに基づいてビジネスまたは組織の意思決定を行うことができます。このロードタイプを使用して、ビッグデータ処理のためにデータウェアハウスとクラウドデータレイクに最新の変更をフィードすることもできます。
- *初期ロードと増分ロード*。ターゲットへのポイントインタイムデータの初期ロードを実行してから、同じソーステーブルに対して継続的に行われた増分データ変更のプロパゲートに自動的に切り替わります。

各ロードタイプでサポートされるソースとターゲットの詳細については、「一括取り込みデータベースのソースタイプとターゲットタイプ」を参照してください。

データベース統合タスクは、名前の一致に基づいて、ソースのテーブルとフィールドをターゲットのテーブルとフィールドに自動的にマッピングします。ルールを定義して、ターゲットテーブル名をカスタマイズできます。

使用例

一括取り込みデータベースは、複数のビジネス上の問題を解決するために使用できます。

一括取り込みデータベースは次のシナリオで使用できます。

- **オフラインレポート**。データベースのパフォーマンスの低下を回避するために、ユーザーのレポートアクティビティをミッションクリティカルな本番環境データベースシステムから別のレポートシステムに移動します。
- **データウェアハウジング**。オンプレミスデータベースを含む複数のデータベースからデータウェアハウスシステムにデータを転送することにより、データウェアハウスの構築を支援します。データウェアハウスへの

データの最初のバッチロードの後、一括取り込みデータベースはソースデータベースからデータの変更を継続的にプロパゲートして、データウェアハウス内のデータを最新の状態に保つことができます。

- **リアルタイムの不正検出**。一括取り込みデータベースが変更データを継続的に提供することで最新の状態に保たれているレプリカデータベースに対して、リアルタイムの不正検出分析を実行します。不正検出プロセスは、ソースシステムを劣化させることなく、最新のデータに対して実行できます。
- **ビッグデータアプリケーションとの調整**。データベース管理システム（DBMS）のオンプレミスソースとデータレイクの同期を維持するか、大規模処理のデータ統合にデータを提供します。
- **クラウドベースのシステムへの移行**。オンプレミスデータベースシステムからクラウドベースのシステムにデータを移行します。

サポートされているソースタイプとターゲットタイプ

一括取り込みデータベースがサポートするソースタイプとターゲットタイプはデータベース統合タスクが使用するロードタイプによって異なります。ロードタイプとは、初期ロードのポイントインタイム操作、変更のみの増分ロード、初期ロードとそれに続く増分ロードのいずれかです。

サポートされているソースとターゲットのバージョンについては、KB 記事「What are the supported sources and targets for IICS Cloud Mass Ingestion service?」を参照してください。

ソースタイプ

次の表に、各ロードタイプでサポートされているソースタイプ（S）を示します。

ソースタイプ	初期ロード	増分ロード	初期ロードと増分ロード
DB2 for i	S	S	S
DB2 for Linux、UNIX、および Windows（LUW）	S	S[1]	S[1]
DB2 for z/OS	S	S	S
Microsoft SQL Server（オンプレミスの SQL Server、RDS for SQL Server、Microsoft Azure SQL Database、Azure SQL Managed Instance を含む）	S	S[2]	S[2]
MongoDB	S	S	-
MySQL（Amazon Aurora MySQL を含む）、Cloud SQL for MySQL、および RDS for MySQL を含む）	S	S	S
Netezza	S	-	-
Oracle（RDS for Oracle を含む）	S	S	S

ソースタイプ	初期ロード	増分ロード	初期ロードと増分ロード
PostgreSQL（オンプレミスの PostgreSQL を含む）、Amazon Aurora PostgreSQL、Azure Database for PostgreSQL - Flexible Server、Cloud SQL for PostgreSQL、および RDS for PostgreSQL	S	S	S
SAP HANA、SAP HANA Cloud	S	S	-
Teradata	S	-	-
1. Db2 for LUW ソースの場合、増分ロードジョブおよび組み合わせロードジョブは、クエリベースのキャプチャメソッドでのみサポートされます。 2. Azure SQL Database ソースの場合、増分ロードジョブおよび組み合わせロードジョブは、クエリベースおよび CDC テーブルキャプチャメソッドでサポートされます。トランザクションログを使用したログベースのキャプチャはサポートされていません。			

これらのソースタイプに使用するコネクタを判断するには、「コネクタと接続」 > 「一括取り込みデータベースコネクタ」を参照してください。

ターゲットタイプ

次の表に、各ロードタイプでサポートされているターゲットタイプ（S）を示します。

ターゲットタイプ	初期ロード	増分ロード	初期ロードと増分ロード
Amazon Aurora PostgreSQL	S	S	S
Amazon Redshift、Amazon Redshift Serverless	S	S	S
Amazon S3	S	S	S
Apache Kafka、Confluent Kafka、Amazon Managed Streaming for Apache Kafka（MSK）	-	S	-
Kafka クライアントでの使用が有効になっている Azure Event Hubs	-	S	-
Microsoft SQL Server（オンプレミスの SQL Server、RDS for SQL Server、Microsoft Azure SQL Database、Azure SQL Managed Instance を含む）	S	S	S
Databricks Delta	S	S	S
フラットファイル	S	-	-
Google BigQuery	S	S	S
Google Cloud Storage	S	S	S
Microsoft Azure Data Lake Storage Gen2	S	S	S

ターゲットタイプ	初期ロード	増分ロード	初期ロードと増分ロード
Microsoft Azure Synapse Analytics	S	S	S
Microsoft Fabric OneLake	S	-	-
Oracle（RDS for Oracle を含む）	S	S	S
Oracle Cloud Infrastructure（OCI）Object Storage	S	S	S
Snowflake	S	S	S

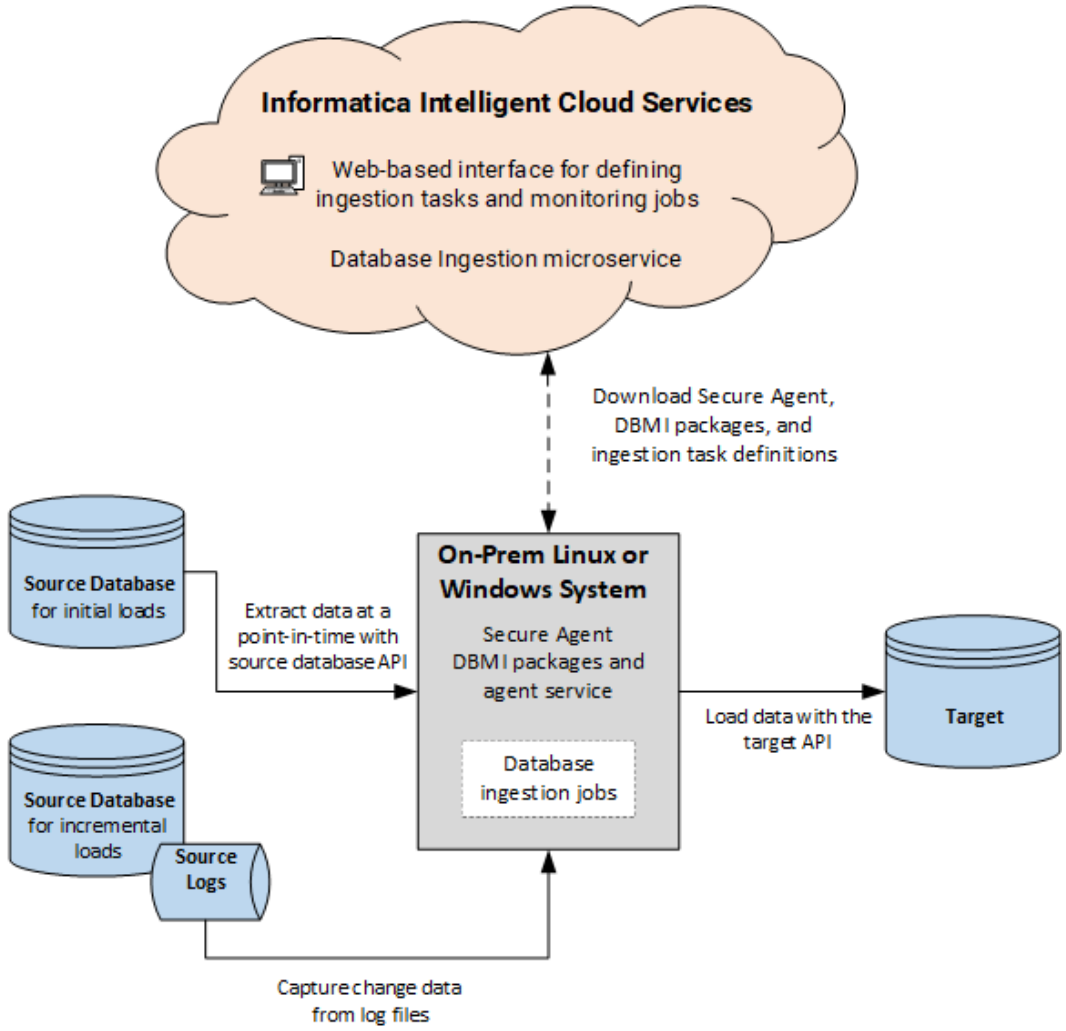
これらのターゲットタイプに使用するコネクタを判断するには、「コネクタと接続」 > 「一括取り込みデータベースコネクタ」を参照してください。

一括取り込みデータベースアーキテクチャ

一括取り込みデータベースは Secure Agent 上で実行されます。一括取り込みデータベースを使用する前に、Linux または Windows マシンに Secure Agent をインストールする必要があります。

Secure Agent を初めて起動した後、一括取り込みデータベースエージェントとパッケージがローカルにインストールされます。その後、一括取り込みを使用して、データベース統合タスクを設定して、データベース統合ジョブを実行および監視することができます。

次のイメージは、一括取り込みデータベースの一般的なアーキテクチャを示しています。



Informatica Intelligent Cloud Services の Web ベースのインタフェースから、取り込みタスクを作成および管理し、取り込みジョブを実行および監視できます。

次のようなインタラクションが発生します。

1. Secure Agent をオンプレミスシステムにダウンロードすると、Database Ingestion のライセンスがある場合には、Database Ingestion DBMI パッケージもダウンロードされます。その後、DBMI エージェントサービスを設定できます。
2. Informatica Intelligent Cloud Services Web ベースのインタフェースから、データベース取り込みタスクを定義します。
3. データ取り込みタスクをデプロイすると、対応する実行可能ジョブが Secure Agent システムに作成されます。
4. データベース取り込みジョブを実行すると、取り込みタスクのメタデータが Secure　Agent にプッシュダウンされます。取り込みジョブは、この情報を使用してデータを処理します。
   - 初期ロード操作の場合、取り込みジョブは特定の時点のデータをソーステーブルとフィールドから抽出します。ジョブは、リレーショナルソースのデータベース API を使用してデータを取得します。

- 増分ロード操作の場合、取り込みジョブは、ソースデータベースログからソーステーブルとフィールドの挿入、更新、削除などの変更をキャプチャします。変更データのキャプチャは、継続的に実行されるか、ジョブが停止または終了するまで実行されます。

データは、適切なターゲット API を使用してターゲットにロードされます。

一括取り込みデータベースのシステム要件

次の表に、Secure Agent に対する一括取り込みデータベースの最小システム要件を示します。

コンポーネント	最小要件
CPU あたりのコア数	最小 8、初期ロードで多数のソーステーブルを処理する必要がある場合は 16 を推奨
メモリ	32GB
ディスク容量	2 KB の行サイズに基づいて、ジョブごとに 5 GB

一般的な制限とガイドライン

データベース統合タスクを初期ロード操作、増分ロード操作、または初期ロード操作と増分ロード操作の組み合わせに設定する前に、次の制限とガイドラインを確認してください。

- 一括取り込みデータベースは Hosted Agent をサポートしていません。
- データベース統合タスクに割り当てられているランタイム環境の Secure Agent が終了すると、関連付けられた取り込みジョブをデプロイ解除し、タスクを更新して別のランタイム環境を指定し、タスクを再度デプロイすることはできません。この場合、次のいずれかのアクションを実行します。
  - 別の Secure Agent をランタイム環境に割り当てます。新しい Secure Agent が動作していることを確認します。次に、関連する取り込みジョブを再開します。
  - このタスクをコピーします。タスクコピーで、アクティブな Secure Agent を持つ別のランタイム環境を指定します。次に、タスクをデプロイし、関連する取り込みジョブを実行します。
- 一括取り込みデータベースはコードページとして UTF-8 を使用します。接続を定義するときに別のコードページタイプを選択した場合、一括取り込みデータベースはそれを無視します。
- 一括取り込みデータベースは、ソーステーブルの各行が一意であることを前提としています。Informatica では、ソーステーブルにプライマリキーを設定することをお勧めします。テーブルにプライマリキーがない場合、一括取り込みデータベースは、LOB カラムを除くすべてのカラムの値を使用して、各ソース行を一意に識別します。この場合、各更新操作は、ターゲットに対する削除とそれに続く挿入として処理されます。

  注: ソースプライマリキーを変更し、Amazon S3、フラットファイル、Google Cloud Storage、または Microsoft Azure Data Lake Storage ターゲットがある場合、一括取り込みデータベースは、各更新操作をターゲットに対する削除とそれに続く挿入として処理します。
- 初期ロードジョブに対して生成されるログファイル名は、タスク名、スキーマ名、テーブル名の組み合わせです。結果として得られたファイル名が Secure Agent オペレーティングシステムで許可されている最大長を超える場合、ファイルが作成されないか、見つからないことがあります。ファイル名がオペレーティング

システムの制限を超えそうな場合は、タスク名を短くして、予想されるファイル名の全体的な長さが短くなるようにします。

- 初期ロードと増分ロードの組み合わせジョブで、初期アンロードフェーズ中に増分挿入変更レコードをキャプチャした場合、ジョブは同じ行に対して削除を実行して、初期アンロードで取得された可能性のある重複を削除します。この意図的に作成されたアクティビティは、監査および論理削除適用モードで反映されます。
- 一括取り込みデータベースでは、1024 文字を超えるエラーメッセージはユーザーインタフェースに表示しません。その代わり、一括取り込みデータベースは自動的にダウンロードされる、エラーのあるログファイルを表示するように求めます。
- ソースカラムに、ターゲット上のどの数値データ型とも互換性のない数値データ型がある場合、一括取り込みデータベースはソースカラムをターゲット varchar カラムにマップします。

一括取り込みデータベースソース - 準備と使用

データベース統合タスクを初期ロード、増分ロード、または初期操作と増分操作の組み合わせに設定する前に、ソースデータベースを準備し、ソースの使用に関する考慮事項を確認して、予期しない結果を回避します。

DB2 for i ソース

データベース取り込みタスクで DB2 for i ソースを使用するには、最初にソースデータベースを準備し、使用に関する考慮事項を確認してください。

ソースの準備:

- 増分ロードジョブ、および初期ロードジョブと増分ロードジョブの組み合わせの場合、一括取り込みデータベースで使用する DB2 ジャーナルオブジェクトおよびファイルオブジェクトにアクセスするための適切なレベルの権限を、データベース統合ジョブを実行するユーザー ID に付与します。以下の表に、これらのオブジェクトとその DB2 権限要件を示します。

オブジェクト	権限
ジャーナル	*USE *OBJEXIST
ジャーナルライブラリ	*USE
ジャーナルレシーバ	*USE
ジャーナルレシーバライブラリ	*USE
ファイル	*USE
ファイルライブラリ	*USE

- 増分ロードジョブの場合、選択したソーステーブルに対応する各データベース物理ファイルでジャーナルをアクティブにする必要があります。また、各ジャーナルは、変更データの前後のイメージの両方を保存するために IMAGES(*BOTH)オプションを使用して設定する必要があります。

ソーステーブルの物理ファイルでジャーナルがアクティブになっていない場合、データベース統合タスクを定義するときに、それをアクティブにする CDC スクリプトを生成できます。スクリプトは次のコマンドを発行します。これにより、ジャーナルがアクティブになり、IMAGES オプションが BOTH に設定されます。

```
CALL QSYS2.QCMDEXC('STRJRNPF FILE(library/physical-file) JRN(library/journal-name)
IMAGES(*BOTH)')
```

ソーステーブルの物理ファイルに対してジャーナルがすでにアクティブになっている場合、CDC スクリプト出力には次のコメントが含まれます。

```
Table 'table_name' is skipped because journaling is already enabled.
```

- 一括取り込みデータベースはデフォルトで、DataDirect JDBC for IBM DB2 ドライバを使用して DB2 for i データベースに接続します。ソースデータベースへの DB2 for i 接続を作成およびテストする最初のユーザーに、データベースに対する DBA 権限を持たせることをお勧めします。この権限は、ドライバが DB2 へのアクセスに使用するパッケージを作成してアップロードし、パッケージに対する EXECUTE 特権を PUBLIC に付与するために必要です。DBA ユーザーが最初の接続テストを実行しない場合は、パッケージを作成するための CRTSQLPKG コマンドに*USE 権限を付与し、パッケージの作成先のライブラリに*CHANGE 権限を付与する必要があります。
- DB2 for i ソースに SSL データ暗号化を使用するには、DB2 for i 接続プロパティーを設定するときに、**［JDBC ドライバ］**フィールドで **［JTOpen］**を選択し、**［暗号化方法］**フィールドで **［SSL］**を選択します。
  また、次のいずれかの場所にある Informatica Cloud Secure Agent JRE cacerts キーストアに必要な証明書を追加します。

  Linux の場合:

  Secure Agent Directory\jdk\jre\lib\security\cacerts

  Windows の場合:

  Secure Agent Directory\apps\jdk*LatestVersion*\jre

  証明書を追加したら、Secure Agent を再起動して、最新インスタンスのエージェントサービスの app-truststore.jks ファイルに変更が反映されていることを確認します。

  キーストアへの証明書の追加については、
  HOW TO: Import certificates into Informatica Cloud Secure Agent JRE を参照してください。

使用に関する考慮事項:

- 一括取り込みデータベースでは、ソーステーブルの各行が一意であることを想定しているため、各ソーステーブルにプライマリキーを持たせることをお勧めします。一括取り込みデータベースは、プライマリキーの代わりに一意のインデックスを許可しません。プライマリキーが指定されていない場合、一括取り込みデータベースはすべてのカラムをプライマリキーの一部であるかのように扱います。
- データベース取り込みタスクを定義する場合、**［ソース］**ページで、ジャーナルが有効になっているソーステーブルに関連付けられているジャーナル名を指定します。

  重要: ［ジャーナル名］フィールドの大文字と小文字および名前のスペル、テーブルの選択ルールが、DB2 ソースカタログのジャーナルとテーブル名の値と一致することを確認してください。
- スキーマドリフトオプションは、データベース取り込み増分ロードジョブ、および初期ロードジョブと増分ロードジョブの組み合わせで、DB2 for i ソースに対して使用できます。
  ［カラムの追加］オプションを［レプリケート］に設定してから、デフォルト値を持つカラムを DB2 for i ソーステーブルに追加すると、一括取り込みデータベースは、ターゲットに新たに追加されたテーブル行にデフォルト値を追加します。ただし、ターゲットの既存の行は更新されず、デフォルト値が反映されません。既存のターゲット行に移入されたデフォルト値を取得するには、別の初期ロードを実行してターゲットを再実体化します。
- 一括取り込みデータベースでは、次の DB2 for i データ型はサポートされません。

  - Blob

  - CLOB

- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

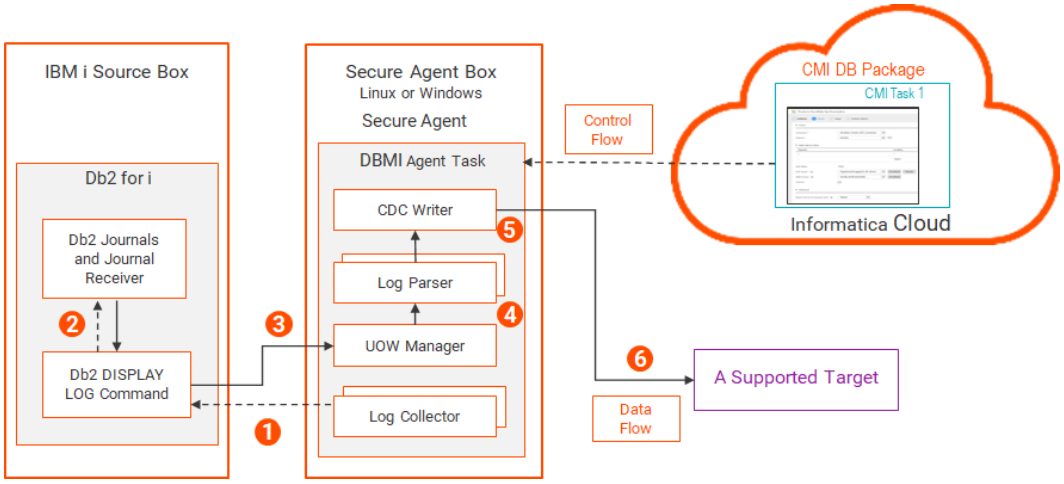
サポートされているソースデータ型からターゲットデータ型へのデフォルトのマッピングについては、「デフォルトデータ型のマッピング」 (ページ 169)を参照してください。

Db2 for i ソースの変更キャプチャメカニズム

一括取り込みデータベースは、IBM i 上の Db2 ソースから変更データをキャプチャしてそのデータをターゲットに適用するための、単一の変更キャプチャメカニズムとアーキテクチャを提供します。

Secure Agent は、IBM i ソースシステムとは別に、Linux または Windows ボックスで実行する必要があります。IBM i システムと Secure Agent ボックス間のネットワーク帯域幅は堅牢でなければなりません。Informatica では、数百ギガビットまたは 1 ギガバイト以上のログデータを処理できるネットワーク転送速度を推奨しています。Db2 が CDC 対象のログデータを生成する速度以上の速度でログデータを Secure Agent に配信できるほどのネットワーク転送速度ではない場合、データベース取り込みジョブはタイムリーにデータをターゲットに提供できません。データスループットが SLA を満たしていない場合は、ハードウェアを変更して、IBM i システムと Secure Agent ボックス間のイーサネット帯域幅を増やすことを検討してください。

次の図は、Db2 for i 変更キャプチャコンポーネントとデータフローを示しています。



1. Secure Agent 配下の DBMI Agent サービスで実行されるログコレクターは、Db2 for i のジャーナルされたデータに対する Db2 DISPLAY LOG コマンドに要求を送信します。
   各要求には、開始 RBA と、データベースの取り込み用の CDC 対象のテーブルのリストが含まれます。
2. このコマンドは、Db2 ジャーナルおよびジャーナルレシーバーからソーステーブルのデータを要求します。
3. このコマンドは、データを含むジャーナルエントリを UOW（Unit Of Work）マネージャに返します。
4. UOW マネージャは、コミットされたトランザクション順にジャーナル処理済みデータをログパーサーに送信します。

5. ログパーサーは、コミットされたトランザクションからの DML 変更を解析します。CDC ライターにデータを送信する前に、ログパーサーはデータを Db2 ジャーナルデータの標準的な形式に変換します。この形式は、DBMI Agent タスクによって消費でき、ターゲットに適用できます。

   注: このリソース集約型のアクティビティは Secure Agent ボックスで発生するため、IBM i システムの CPU 消費は最小限に抑えられます。

6. CDC ライターが、フォーマットされたデータをターゲットに適用します。

DB2 for LUW ソース

データベース取り込みタスクで DB2 for Linux、UNIX、および Windows（LUW）ソースを使用するには、最初にソースデータベースを準備し、使用に関する考慮事項を確認してください。

ソースの準備:

データベース取り込みタスクを定義するときにソースシステム上の DB2 テーブルのリストを取得するには、一括取り込みデータベースのユーザーに、いくつかのシステムカタログテーブルおよびビューに対する SELECT 特権が必要です。次の grant 文を使用します。

```
GRANT SELECT ON <catalog_table> TO <dbmi_user>
```

この付与を、次の各カタログテーブルまたはビューに対して発行します。

- SYSCAT.SCHEMATA
- SYSCAT.DATAPARTITIONEXPRESSION
- SYSCAT.DATAPARTITIONS
- SYSCAT.TABLESSYSCAT.TABLES
- SYSCAT.COLUMNS
- SYSCAT.INDEXCOLUSE
- SYSCAT.INDEXES
- SYSIBM.SYSVERSIONS
- SYSIBM.SYSDUMMY1
- SYSIBM.COLUMNS

使用に関する考慮事項:

- Db2 for LUW ソースを持つデータベース取り込みの増分ロードジョブ、および初期ロードと増分ロードの組み合わせジョブでは、**[クエリベース]** キャプチャメソッドを使用する必要があります。クエリベースの変更データキャプチャでは、共通の CDC クエリカラムを参照する WHERE 句を含む SQL 文を使用して、挿入および更新の変更がある行を識別します。ソースデータベースの設定は、各ソーステーブルへの CDC クエリカラムの追加に制限されます。ユーザーは、少なくともソーステーブルへの読み取り専用アクセス権を持つ必要があります。CDC クエリカラムタイプは、タイムゾーンがないタイムスタンプに相当する必要があります。現在、この CDC 機能は Snowflake ターゲットでのみテストされています。
- 一括取り込みデータベースでは、ソーステーブルの各行が一意であることを想定しているため、各ソーステーブルにプライマリキーを持たせることをお勧めします。一括取り込みデータベースは、プライマリキーの代わりに一意のインデックスを許可しません。プライマリキーが指定されていない場合、一括取り込みデータベースはすべてのカラムをプライマリキーの一部であるかのように扱います。
- 一括取り込みデータベースでは、次の DB2 for LUW データ型はサポートされません。

  - LONG VARCHAR

  - LONG VARGRAPHIC

- LONG VARCHAR FOR BIT DATA
- BLOB
- CLOB
- DBCLOB
- nclob
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

サポートされているソースデータ型からターゲットデータ型へのデフォルトのマッピングについては、「デフォルトデータ型のマッピング」 (ページ 169)を参照してください。

DB2 for z/OS ソース

データベース取り込みタスクで DB2 for z/OS ソースを使用するには、最初にソースデータベースを準備し、使用に関する考慮事項を確認してください。

ソースの準備:

- DB2 for z/OS ソースを使用したデータベース取り込み増分ロードジョブは、ストアドプロシージャを使用して DB2 Instrumentation Facility Interface（IFI）を呼び出し、z/OS ソースシステム上の DB2 ログから変更データを読み取ります。一括取り込みデータベースは、ストアドプロシージャライブラリと JCL を DB2 for zOS Database Ingestion コネクタパッケージの ZIP ファイルで提供します。ストアドプロシージャライブラリを APF 許可ライブラリで受け取り、ご使用の環境に合わせて JCL をカスタマイズする必要があります。ストアードプロシージャは、DB2 ソースシステムの Workload Manager（WLM）アドレススペースで実行されます。z/OS システム要件、ストアドプロシージャのセットアップ、および必要な権限の詳細については、「Db2 for z/OS CDC のストアドプロシージャのインストールと設定」 (ページ 20)」を参照してください。
- データベース取り込み増分ロードタスクで DB2 ソースを定義するときに、**［CDC スクリプト］**フィールドで**［すべてのカラムの CDC を有効化］**オプションを選択する必要があります。一括取り込みデータベースは、ソーステーブルおよび CDC に使用される特定の Db2 カタログテーブルで Db2 DATA CAPTURE CHANGES を有効にするためのスクリプトを生成します。この属性が 1 つのジョブに設定されると、他のすべてのジョブは、Db2 で必要なカタログテーブルに対してその属性が有効になったことを認識します。十分な権限がある場合はユーザーインタフェースから CDC スクリプトを実行するか、SYSDBA 権限を持つ DB2 DBA にスクリプトの実行を依頼することができます。
- 一括取り込みデータベースユーザーに、データベース取り込みロードタイプを実行するために必要な Db2 for z/OS 特権があることを確認してください。詳細については、「DB2 for z/OS の権限」 (ページ 24)を参照してください。
- 一括取り込みデータベースは、Progress DataDirect JDBC IBM DB2 ドライバを使用して、Db2 for z/OS ソースに接続します。Db2 for z/OS ソースを使用した一括取り込みデータベースの新しい実装の場合、増分ジョブを最初に実行するユーザーは、JDBC 接続を確立するための SYSADM または SYSDBA 権限を持っている必要があります。
- 初期ロードと増分ロードの両方を実行し、これらのジョブタイプに対して異なる Db2 for z/OS 特権を持った異なるユーザーを使用する場合は、次の手順を実行します。
  1. 初期ロードと増分ロード用に個別の Db2 for z/OS データベース取り込み接続を作成します。接続プロパティで、接続を使用する初期ロードジョブまたは増分ロードジョブに必要な Db2 特権を持つユーザーを指定します。
  2. 接続へのアクセスを特定のユーザーに制限するには、管理者で接続アセットに明示的なユーザー権限を設定します。ジョブを実行するために接続を使用するには、ユーザーが実行権限を持っている必要があ

ります。これにより、より低いレベルの特権を持つユーザーは、より高いレベルの権限を必要とするジョブを実行できなくなります。

3. 初期ロードまたは増分ロードタイプを使用するデータベース取り込みタスクを作成する場合は、必要な Db2 特権とアセット権限を持つユーザーが指定された接続を選択します。

使用に関する考慮事項:

- 一括取り込みデータベースでは、ソーステーブルの各行が一意であることを想定しているため、各ソーステーブルにプライマリキーを持たせることをお勧めします。一括取り込みデータベースは、プライマリキーの代わりに一意のインデックスを許可しません。プライマリキーが指定されていない場合、一括取り込みデータベースはすべてのカラムをプライマリキーの一部であるかのように扱います。
- z/OS のデフォルト設定では、カタログテーブルに対して ALTER 文を実行するとスキーマドリフトが有効になります。1 つのジョブに対してスキーマドリフトを有効にすると、すべてのジョブに対して有効になります。ALTER 文の実行中にジョブが実行されている場合、実行中のジョブでは、その後停止して再開するまでスキーマドリフトは有効になりません。
- 一括取り込みデータベースでは、Db2 11 for z/OS ソースのスキーマドリフトはサポートされていません。
- 一括取り込みデータベースでは、次の DB2 for z/OS データ型はサポートされません。

  - Blob

  - CLOB

  - DBCLOB

  - XML

  データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

  データ型のデフォルトのマッピングについては、「「デフォルトデータ型のマッピング」 (ページ 169)」を参照してください。

Db2 for z/OS CDC のストアドプロシージャのインストールと設定

増分ロードジョブの Db2 for z/OS CDC 処理を実行するために、一括取り込みデータベースは、z/OS ソースシステムで実行されるストアドプロシージャを提供します。ストアドプロシージャは、Db2 Instrumentation Facility Interface（IFI）を呼び出して、Db2 ログから変更データを収集します。

z/OS システム要件

開始する前に、Db2 for z/OS ソースシステムが次の要件を満たしていることを確認してください。

- z/OS システムには、2.3 以降の推奨オペレーティングシステムバージョンがあります。
- Db2 for z/OS ソースは、Db2 バージョン 11 または 12 を使用します。
- 一括取り込みデータベースストアドプロシージャを実行するための Workload Manager（WLM）アドレススペースが存在することを確認してください。

  Db2 WLM アドレススペースをセットアップしていない場合、詳細については、次の IBM ドキュメントを参照してください。

  - Db2 11 の場合:
    https://www.ibm.com/support/knowledgecenter/en/SSEPEK_11.0.0/inst/src/tpc/db2z_setupwlmenvironment.html

  - Db2 12 の場合:
    https://www.ibm.com/support/knowledgecenter/en/SSEPEK_12.0.0/inst/src/tpc/db2z_setupwlmenvironment.html

- Db2 for z/OS CDC のストアドプロシージャが受信されるライブラリが存在し、z/OS システムで APF 許可されていることを確認してください。

ストアドプロシージャライブラリをインストールし、JCL をカスタマイズします

クライアントマシンで、次の手順を実行します。

1. Db2 for zOS Database Ingestion コネクタが使用可能であることを確認します。

   Db2 for zOS Database Ingestion コネクタパッケージには、Db2 for z/OS ストアドプロシージャライブラリが含まれています。コネクタが Secure Agent Group（ランタイム環境）に対して有効になっている場合、コネクタパッケージの.zip ファイルがインストール場所のダウンロードフォルダにダウンロードされます。パッケージ名の形式は package-DB2ZMI です。*nnnnn*。*nnnnn* は増分されたパッケージバージョン番号です。複数のパッケージバージョンが存在する場合は、最新のバージョンを使用してください。

2. package-DB2ZMI*nnnnn*.zip ファイルを解凍します。ストアドプロシージャファイルは、パッケージ名の下の Db2WLMStoredProcedure フォルダに追加されます。

3. FTP を使用して、Db2WLMStoredProcedure フォルダ内の#STPINST ファイルを z/OS システム上のシーケンシャルファイル、PDS、または PDSE に転送します。

   注:

   - バイナリモードを設定せずにファイルを転送します。
   - システム要件を満たすために必要な場合は、高レベル修飾子（HLQ）を追加します。

4. FTP または別のファイル転送方法を使用して、次のファイルをそれぞれ z/OS システム上の別のデータセットに転送します。

   - DBMI.ZOS.DBRMLIB.XMI
   - DBMI.ZOS.LOADLIB.XMI
   - DBMI.ZOS.USERJCL.XMI

   注:

   - システム要件を満たすために必要な場合は、HLQ を追加または編集します。
   - バイナリモードでファイルを転送します。
   - 各データセットに LRECL=80、BLKSIZE=3120、および RECFM=FB の DCB 属性があることを確認してください。

     必要な属性値を指定するために、データセットの事前の割り当てが必要になる場合があります。

z/OS システムでは、TSO を使用して送信（XMI）データセットを APF 許可ライブラリで受信し、ストアドプロシージャ JCL メンバを編集します。また、DB2 ユーザー特権を設定し、使用されている場合はリソース制限テーブルに行を追加します。

1. DBRMLIB 転送データセットを受信します。

   ```
   RECEIVE INDATASET(DBMI.ZOS.DBRMLIB.XMI)
   ```

   注:

   - データセットを z/OS に転送するときに HLQ を指定した場合は、HLQ を含めてください。
   - メッセージ「INMR906A Enter restore parameters or 'DELETE' or 'END' +」が表示されたら、APF 許可ライブラリを入力します。

     DA(*your.library_name*) UNIT(*unit*) VOLUME(*volume*)

     UNIT()および VOLUME()オペランドはオプションです。インストールで RECEIVE ファイルがデフォルトでワークユニットまたはボリュームに配置されない場合は、それらを含めてください。

2. LOADLIB 転送データセットを受信します。

   ```
   RECEIVE INDATASET(DBMI.ZOS.LOADLIB.XMI)
   ```

   手順 1 の注を参照してください。

3. USERJCL 転送データセットを受信します。

   ```
   RECEIVE INDATASET(DBMI.ZOS.USERJCL.XMI)
   ```

   手順 1 の注を参照してください。

4. ご使用の環境のストアドプロシージャ JCL を含む#STPINST ファイルをカスタマイズします。

   JCL は、データの Db2 IFI への要求の結果を保持するストアドプロシージャとグローバル一時テーブルを作成します。また、ストアドプロシージャパッケージをバインドします。

   注:

   - JCL ファイルの先頭にあるコメントに基づいて、JCL 内の変数を、ステップ 1 で受信した Db2 サブシステム名（!DSN!）、ストアドプロシージャスキーマ名(!SCHEMA!)、プロシージャ名（!STRPRC!）、WLM 環境名（!WLMENV!）、および DBRMLIB 送信データセットの名前（!DBRMLIB!）など z/OS 環境に適した値に置き換えます。
   - 受信したデータセットに HLQ を使用した場合は、HLQ を JCL に含めてください。
   - WLM 環境名は、プロシージャ APPLENV パラメータまたは WLM アドレススペースの EXEC PARM で指定されます。
     プロシージャパラメータ:

     ```
     //STARTING EXEC DSNBWLMG,DB2SSN=DSNB,APPLENV='DSNBWLM_GENERAL'
     ```

     EXEC PARM:

     ```
     PARM='DSNB,40,DSNBWLM_GENERAL'
     ```

   - 受信した LOADLIB ライブラリを APF 許可後に使用するか、ライブラリのコンテンツを独自の APF 許可ライブラリにコピーすることができます。
   - WLM アドレススペースの STEPLIB 連結には、Db2 IFI を実行するために、APF 許可ライブラリのみが含まれている必要があります。

5. 受信した USERJCL データセットのメンバの JCL をカスタマイズします。詳細については、「「 Db2 for z/OS USERJCL データセット 」 (ページ 22)」を参照してください。

6. ストアドプロシージャ JCL ジョブを実行する前に、必要な DB2 権限が付与されていることを確認します。詳細については、「DB2 for z/OS の権限」 (ページ 24)を参照してください。

7. データベース取り込みパッケージの DB2 DSNRLST*xx* リソース制限テーブルに行を追加して、プロセッサリソースがストアドプロシージャの処理に十分であることを確認します。そうしないと、増分ロードジョブが異常終了する可能性があります。次のカラムを含む行を追加します。

   - データベース取り込みタスクが使用する認証 ID を持つ AUTHID カラムか、データベース取り込みタスクと同じパッケージ名を持つ RLFPDG カラム、またはこれら両方のカラム。
   - NULL またはデフォルトの制限値より大きいリソース制限が定義された ASUTIME カラム。

   次に、リソース制限テーブルへの変更を有効にするために、DB2 -START RLIMIT コマンドを発行します。

Db2 for z/OS USERJCL データセット

ダウンロードされた USERJCL データセットは区分データセット（PDS）または拡張区分データセット（PDSE）であり、Informatica 提供の WLM Db2 ストアドプロシージャの使用時に他の方法では容易に入手できない情報を収集するジョブを実行するための JCL メンバが含まれています。

メンバ JCL は、USERJCL PDS または PDSE で完全にカスタマイズできます。元のメンバが参照用にそのまま保持されるように、提供された JCL メンバのコピーを別の名前で作成し、そのコピーをカスタマイズすることをお勧めします。

複数の Db2 サブシステムに複数のストアドプロシージャをインストールする場合、インストールできる USERJCL ライブラリは 1 つだけで、そのライブラリに Db2 サブシステムごとに合わせてカスタマイズしたメンバを作成できます。あるいは、特定の Db2 サブシステム用に別のライブラリを作成することもできます。データベース統合ジョブに、Db2 サブシステムの正しいライブラリおよびメンバ情報が含まれていることを確認してください。

USERJCL PDS または PDSE には、以下のメンバが含まれています。

LOGINV **メンバ**

LOGINV には、Db2 ログインベントリリストを取得するジョブの JCL が含まれています。タスクウィザードで取り込みタスクの **［増分ロード操作の当初の開始点］** プロパティを **［特定の日付と時刻］** に設定した場合、インベントリリストを使用して、データベース取り込みジョブの初回実行時または再開時にログ内の開始点が決定されます。ログインベントリリストは一括取り込みデータベースに、開始 RBA または LSN と終了 RBA または LRSN、および Db2 サブシステムのすべてのアクティブログとアーカイブログの開始タイムスタンプと終了タイムスタンプを提供します。一括取り込みデータベースは、この情報を使用して適切なログアーカイブを選択し、要求された開始点を検索します。**［増分ロード操作の当初の開始点］** プロパティが **［特定の日付と時刻］** 以外のオプションに設定されている場合は、USERJCL ライブラリをインストールする必要はありません。

LOGINV メンバの内容:

```
//<USERID>I JOB  'LOG INVENTORY',MSGLEVEL=(1,1),MSGCLASS=X,
//      NOTIFY=&SYSUID,CLASS=A,REGION=0M
//* ------------------------------------------------------
//*
//*  PLEASE CHANGE DSN? TO THE DB2 SUBSYSTEM OF YOUR CHOICE.
//*
//*  THIS JCL MEMBER CAN BE USED TO RUN A LOG INVENTORY
//*  LIST. DBMI WILL REQUEST THIS IF A DBMI JOB IS TRYING TO
//*  RETART USING TIME. THE LOG INVERNTORY GIVES DBMI THE
//*  ABILITY TO CORRELATE RBA/LSRN TO ACTUAL LOG RESTART
//*  POSITIONS. DBMI PARSES THE SYSPRINT OUTPUT TO GET THE
//*  REQUIRED INFORMATION.
//*
//*
//*  SUBSTITUTION TAGS
//*  -----------------
//*
//*  SUBSTITUTION TAGS ARE USED TO INSERT DATA INTO THE JOB
//*  BY DBMI BEFORE IT IS SUBMITTED. YOU MAY COPY THIS JCL
//*  INTO ANOTHER MEMBER AND MODIFY ITS CONTENTS. SUBSTITUTION
//*  TAGS MAY ALSO BE REMOVED AND HARD CODED FOR INDIVIDUAL
//*  JOB NEEDS.
//*
//*
//*  <USERID>     WILL BE REPLACED WITH THE USER ID.
//*
//* ------------------------------------------------------
//LOGINV EXEC PGM=DSNJU004
//STEPLIB DD DISP=SHR,DSN=DSN?10.SDSNLOAD
//        DD DISP=SHR,DSN=DSN?10.DSNC.RUNLIB.LOAD
//SYSUT1   DD DISP=SHR,DSN=DSN?.BSDS01
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//
```

注: この JCL では、一括取り込みデータベースは<USERID>を Db2 for z/OS データベース取り込み接続プロパティで指定されたユーザーに置き換えます

USERJCL メンバジョブをサブミットし、サブミットされたジョブから出力を取得するために、一括取り込みデータベースは、必要に応じて Db2 提供のストアドプロシージャをバッチで実行します。一括取り込みデータベースのユーザーが、これらのストアドプロシージャを実行するために必要な特権を持っていることを確認します。Db2 提供のプロシージャの詳細については、https://www.ibm.com/docs/en/db2-for-zos/12?topic=sql-procedures-that-are-supplied-db2 を参照してください。

USERJCL メンバは、次のように処理されます。

1. LOGINV メンバの場合、データベース取り込みタスクウィザードの［**ソース**］ページで［**増分ロード操作の当初の開始点**］詳細プロパティが［**特定の日付と時刻**］に設定されている場合、一括取り込みデータベースは、ソースに次のカスタムプロパティが指定されているかどうかをチェックします。
   - **pwx.cdcreader.ZOS.Db2JobsDSN**（インストールされている USERJCL PDS または PDSEd の名前を指定）
   - **pwx.cdcreader.ZOS.Db2JobLLOGINVMember**（データベース統合ジョブに使用される LOGINV メンバ名を指定）
2. 一括取り込みデータベースは、Db2 提供のストアドプロシージャ ADMIN_DS_BROWSE を使用して LOGINV メンバを読み取ります。
3. 一括取り込みデータベースは、ジョブの実行に必要なタグを置き換えます。
4. 一括取り込みデータベースは、Db2 提供のストアドプロシージャ ADMIN_SUBMIT_JOB を使用して、ジョブを Db2 for z/OS データベースにサブミットします。
5. 一括取り込みデータベースは、Db2 提供のストアドプロシージャ ADMIN_JOB_QUERY を使用して、サブミットされたジョブのステータスを照会します。
6. ジョブが完了すると、一括取り込みデータベースは、Db2 提供のストアドプロシージャ ADMIN_JOB_FETCH を使用して、ジョブ出力を取得します。

DB2 for z/OS の権限

Db2 for z/OS ソースを持つデータベース統合タスクをデプロイして実行するには、ソース接続で取り込みロードタイプに必要な特権を持つ一括取り込みデータベースユーザーを指定する必要があります。

初期ロード処理の特権

- 初期ロードジョブの場合は、Db2 for z/OS データベース取り込み接続プロパティで指定されたユーザーに次の特権を付与します。

  SELECT on *schema.table* TO *user*

  ここで、*schema.table* はソーステーブルを表します。
- ソースのアンロード処理中に Db2 カタログテーブルに対してクエリを実行するには、次のカタログテーブルに対する SELECT 特権を付与します。

  - SYSIBM.SYSCOLUMNS
  - SYSIBM.SYSINDEXES
  - SYSIBM.SYSKEYS
  - SYSIBM.SYSTABLEPART
  - SYSIBM.SYSTABLES
  - SYSIBM.SYSTABLESPACE

増分ロード処理の特権

増分ロードジョブの場合は、次の特権を付与します。

注: これらを付与するために必要な権限がない場合は、SYSDBA 権限以上の権限レベルを持つ DB2 管理者に発行を依頼してください。

- ユーザーインタフェースからデータベース CDC オプションを有効にする CDC スクリプトを生成するには、インタフェースユーザーに次の権限を発行します。

```
GRANT ALTER TABLE schema.table  DATA CAPTURE CHANGES TO user   <--for each source table
 COMMIT;
```

- 増分変更データ処理の場合は、DB2 for zOS Database Ingestion 接続プロパティで指定されたユーザーに次の権限を付与します。

  - DB2 ログの変更データをストアドプロシージャで使用できるようにする方法:

    ```
    GRANT ALTER TABLE schema.table  DATA CAPTURE CHANGES TO user   <--for each source table
    COMMIT;
    ```

  - ユーザーが DB2 Instrumentation Facility Interface（IFI）から必要な情報を取得できるようにする方法:

    ```
    GRANT MONITOR2 TO user;
    COMMIT;
    ```

  - グローバル一時テーブルの権限をユーザーに付与する方法:

    ```
    GRANT READ ON !SCHEMA!.!STRPRC!_RS_TBL to user;
    GRANT DELETE ON !SCHEMA!.!STRPRC!_RS_TBL to user;
    GRANT EXECUTE !SCHEMA!.!STRPRC! to user;
    COMMIT;
    ```

    ここで、!SCHEMA! および、!STRPRC! はジョブ JCL の変数であり、それぞれストアドプロシージャスキーマ名とプロシージャ名を表します。

    最初の 2 つの特権により、ユーザーは、ストアドプロシージャがログデータを書き込んでいるグローバル一時テーブルのコンテンツを読み取り、削除することができます。3 番目の特権により、ユーザーはストアドプロシージャを実行できます。

  - ストアドプロシージャがその DB2 プランをバインドできるようにするには、次の権限を PUBLIC に付与します。

    ```
    GRANT BIND, EXECUTE ON PLAN !STRPRC! TO PUBLIC;
    COMMIT;
    ```

  - ソースのアンロード処理中に Db2 カタログテーブルに対してクエリを実行するには、次のカタログテーブルに対する SELECT 特権を付与します。

    - SYSIBM.SYSCOLUMNS
    - SYSIBM.SYSDUMMY1
    - SYSIBM.SYSINDEXES
    - SYSIBM.SYSKEYS
    - SYSIBM.SYSTABLEPART
    - SYSIBM.SYSTABLES
    - SYSIBM.SYSTABLESPACE

z/OS でストアドプロシージャを実行するために必要な権限

ストアドプロシージャ JCL ジョブを実行する前に、次の DB2 権限が付与されていることを確認します。

- ストアドプロシージャジョブを実行するユーザーに SYSADM 権限があることを確認するか、Db2 for z/OSDBA にそれを実行するように依頼してください。
- ストアドプロシージャを実行するには、#STPINST JCL ファイルで指定されたプロシージャスキーマ名に次の Db2 権限を付与する必要があります。

  - Db2 カタログテーブルに対する SELECT 権限:

    ```
    GRANT SELECT ON SYSIBM.* TO schema;
    ```

  - JCL で指定されたパッケージ名に対する EXECUTE 権限。

    ```
    GRANT EXECUTE ON PACKAGE package_name TO schema;
    ```

さらに、グローバル一時テーブルの JCL で指定されているスキーマおよびストアドプロシージャ名に対する INSERT および DELETE 権限を付与します。

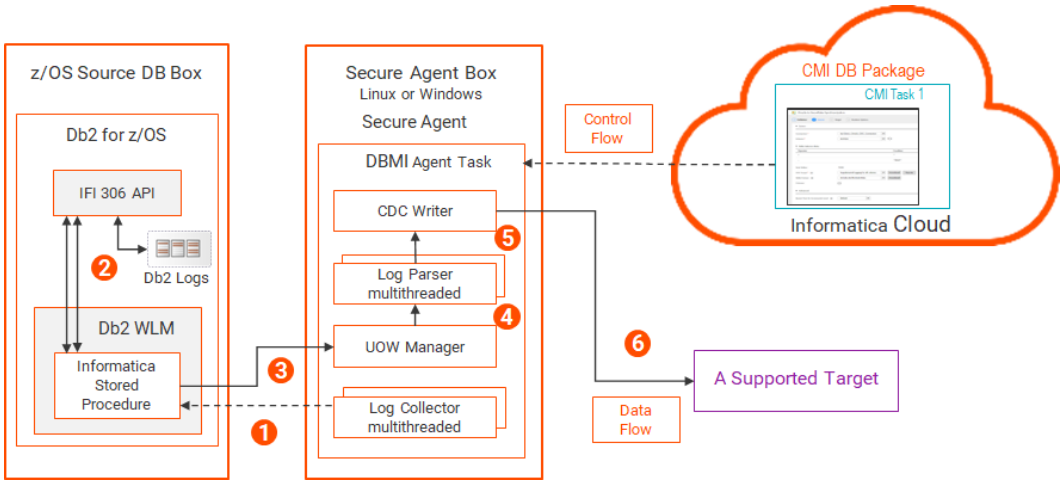
```
GRANT INSERT, DELETE ON schema.stored_procedure_name_RS_TBL TO user
```

Db2 for z/OS ソースの変更キャプチャメカニズム

一括取り込みデータベースは、z/OS 上の Db2 ソースから変更データをキャプチャしてそのデータをターゲットに適用するための、単一の変更キャプチャメカニズムとアーキテクチャを提供します。このアーキテクチャはマルチスレッド処理を使用して、データの収集と、データの解析およびターゲットが受け入れる形式への変換のパフォーマンスを最適化します。

Secure Agent は、Db2 z/OS ソースシステムとは別に、Linux または Windows ボックスで実行する必要があります。z/OS システムと Secure Agent ボックス間のネットワーク帯域幅は堅牢でなければなりません。Informatica では、数百ギガバイトまたは 1 ギガビット以上のログデータを処理できるネットワーク転送速度を推奨しています。Db2 が CDC 対象のログデータを生成する速度以上の速度でログデータを Secure Agent に配信できるほどのネットワーク転送速度ではない場合、データベース取り込みジョブはタイムリーにデータをターゲットに提供できません。データスループットが SLA を満たしていない場合は、ハードウェアを変更して、z/OS システムと Secure Agent ボックス間のイーサネット帯域幅を増やすことを検討してください。

次の図は、Db2 for z/OS 変更キャプチャコンポーネントとデータフローを示しています。



1. Secure Agent 配下の DBMI Agent サービスで実行されるマルチスレッドログコレクタは、Db2 ログデータの複数の同時要求を Db2 ストアドプロシージャに発行します。
   各要求には、開始 RBA、またはログがデータ共有環境にある場合は開始 LRSN と、データベース取り込み用の CDC 対象のテーブルのリストが含まれます。

   一連のログデータの処理中に、ログコレクタは次の一連のログデータを要求できます。
2. Db2 Instrumentation Facility Interface（IFI）API は、アクティブログとアーカイブログから、CDC 対象の選択したソーステーブルのデータを抽出します。その後、IFI はデータを未加工のネイティブ形式で、z/OS Workload Manager（WLM）の Informatica Db2 ストアドプロシージャに転送します。
3. Db2 ストアドプロシージャは、キャプチャされたデータとともに UOW を UOW マネージャに返します。
4. UOW マネージャは、UOW をコミット順にログパーサーに送信します。
5. マルチスレッドログパーサーは、コミットされた UOW からの DML 変更を同時に解析します。結果は、ターゲットタイプが期待する形式の未加工のネイティブ Db2 ログデータです。

   注: このリソース集約型のアクティビティは Secure Agent ボックスで発生するため、z/OS システムの CPU 消費は最小限に抑えられます。
6. CDC ライターが、フォーマットされたデータをターゲットに適用します。

Microsoft SQL Server、RDS for SQL Server、Azure SQL Database、Azure Managed Instance ソース

データベース取り込みタスクで Microsoft SQL Server ソースを使用するには、最初にソースデータベースを準備し、使用に関する考慮事項を確認してください。SQL Server ソースタイプには、オンプレミスの SQL Server、Relational Database Service（RDS）for SQL Server、Azure SQL Database、および Azure SQL Managed Instance が含まれます。

ソースの準備:

- SQL Server ソースタイプの場合、一括取り込みデータベースがサポートしている SQL Server のエディションとバージョンを使用していることを確認します。KB 記事「FAQ: What are the supported sources and targets for IICS Cloud Mass Ingestion service?」を参照してください。
- SQL Server ソースを使用した増分変更データキャプチャ（CDC）操作の場合、一括取り込みデータベースには複数の変更キャプチャメソッドが用意されています。SQL Server ソースデータベースの準備は、使用する CDC メソッドによって異なります。
  使用可能な変更キャプチャメソッドは次のとおりです。
  - トランザクションログと CDC テーブルを使用するログベースの変更データキャプチャ。一括取り込みデータベースは、SQL Server トランザクションログと有効な SQL Server CDC テーブルからデータの変更を読み取ります。この方法では、ユーザーの権限を拡張する必要があります。ソーステーブルで SQL Server CDC を有効にする必要があります。
  - CDC テーブルのみを使用した変更データキャプチャ。ユーザーは少なくともソーステーブルと CDC テーブルに対する SELECT 権限を持っている必要があります。ソーステーブルで SQL Server CDC を有効にする必要があります。
  - クエリベースの変更データキャプチャ。変更データキャプチャでは、共通の CDC クエリカラムを参照する WHERE 句を含む SQL ステートメントを使用して、挿入および更新の変更がある行を識別します。ソースデータベースの設定は、各ソーステーブルへの CDC クエリカラムの追加に制限されます。ユーザーは、少なくともソーステーブルへの読み取り専用アクセス権を持つ必要があります。

  詳細については、「SQL Server ソースの変更キャプチャメカニズム」 (ページ 31)を参照してください。
- トランザクションログによるログベースの CDC を使用する増分ロードジョブの場合は、SQL Server ソース接続で指定したデータベースユーザーに db_owner ロールと VIEW ANY DEFINITION 特権があることを確認します。これらの特権を付与するには、SQL Server ソースタイプに応じて、次のいずれかの SQL 文を使用します。
  Azure SQL Managed Instance を含む SQL Server オンプレミスソースの場合:

```
USE master;
CREATE DATABASE <database>;
CREATE LOGIN <login_name> WITH PASSWORD = '<password>';
CREATE USER <user> FOR LOGIN <login_name>;
GRANT SELECT ON master.sys.fn_dblog TO <user>;
GRANT VIEW SERVER STATE TO <login_name>;
GRANT VIEW ANY DEFINITION TO <login_name>;

USE <db>;
CREATE USER <user> FOR LOGIN <login_name>;
EXEC sp_addrolemember 'db_owner', '<user>';
EXEC sys.sp_cdc_enable_db
```

  RDS for SQL Server の場合:

```
USE master;
CREATE DATABASE <database>;
CREATE LOGIN <login> WITH PASSWORD = '<password>';

USE <database>;
EXEC msdb.dbo.rds_cdc_enable_db '<database>';
CREATE USER <user> FOR LOGIN <login_name>;
```

```
USE master;
GRANT VIEW SERVER STATE TO <login_name >;
GRANT VIEW ANY DEFINITION TO <login_name >;

USE <database>;
EXEC sp_addrolemember 'db_owner', '<user>';
```

- SQL Server ソースを持ち、ログベースの CDC を使用するデータベース取り込み増分ロードジョブ、および初期ロードと増分ロードの組み合わせジョブの場合は、ソースデータベースで SQL Server 変更データキャプチャ（CDC）を有効にする必要があります。

  - オンプレミスの SQL Server ソースの場合は、データベースコンテキストの sys.sp_cdc_enable_db ストアドプロシージャを実行します。sysadmin ロールが必要です。

  - Amazon Relational Database Service（RDS）for SQL Server ソースの場合は、マスターユーザーとしてログインし、msdb.dbo.rds_cdc_enable_db '*database_name*'ストアドプロシージャを実行します。

  SQL Server CDC が有効になると、SQL Server はトランザクションログと CDC テーブルに追加情報を書き込みます。この情報は、一括取り込みデータベースが増分 CDC 処理中に使用します。

  または、データベース取り込みタスクを作成するときに、データベースおよび選択したソーステーブルのすべてのカラムで CDC を有効にするスクリプトを生成することもできます。CDC スクリプトを実行するには、sysadmin ロールが必要です。

  制限: 一括取り込みデータベースでは、1019 を超えるカラムを含むテーブルに対して CDC を有効にすることはできません。

- クエリベースの CDC を使用する増分ロード操作、または初期ロードと増分ロードの組み合わせ操作の場合、ソーステーブルには、変更行を示すために使用される CDC クエリカラムを含める必要があります。データベース統合タスクを作成する前に、クエリカラムをソーステーブルに追加する必要があります。CDC クエリカラムタイプは、タイムゾーンがないタイムスタンプに相当する必要があります。クエリカラムでサポートされている SQL Server データ型は、DATETIME と DATETIME2 です。

使用に関する考慮事項:

- 一括取り込みデータベースは、SQL Server ソースに対して次の代替キャプチャメソッドを提供します。

  - **ログベース**。SQL Server トランザクションログと CDC テーブルから変更データをキャプチャします。

  - **CDC テーブル**。SQL Server CDC テーブルからのみ変更データをキャプチャします。

  - **クエリベース**。CDC クエリカラムを指す SQL WHERE 句を使用して、挿入と更新をキャプチャします。

  各キャプチャメソッドの詳細については、「SQL Server ソースの変更キャプチャメカニズム」 (ページ 31)を参照してください。

- 一括取り込みデータベースは、データベース統合ジョブでオンプレミスの SQL Server、Amazon RDS for SQL Server、Azure SQL Managed Instance、および Azure SQL Database ソースのすべてのロードタイプをサポートします。ただし、増分ロードジョブ、または初期ロードと増分ロードの組み合わせジョブの Azure SQL Database ソースの場合、トランザクションログによる**ログベース**のキャプチャメソッドを使用することはできません。

- ログベースの CDC を使用するタスクの場合に SQL Server データベースで CDC を有効にすると、SQL Server は、SQL Server エージェントによって実行されるキャプチャジョブとクリーンアップジョブを自動的に作成します。キャプチャジョブは、SQL Server CDC テーブルへの入力を担当します。クリーンアップジョブは、CDC テーブルからのレコードをクリーンアップする役割を果たします。CDC テーブルのデータ保持期間のデフォルト値は 72 時間、つまり 3 日です。sys.sp_cdc_help_jobs ストアドプロシージャを実行して結果の保持期間の値を確認すると、現在の保持期間を確認できます。ダウンタイムが 3 日を超えることが予想される場合は、sys.sp_cdc_change_job ストアドプロシージャまたは SQL Server エージェントのクリーンアップジョブで保持期間を調整できます。クリーンアップジョブを一時停止することもできます。

- 一括取り込みデータベースは SQL Server のページ圧縮とソースデータの行圧縮をサポートします。

- 一括取り込みデータベースでは、ソーステーブルの各行が一意であることを想定しているため、各ソーステーブルにプライマリキーを持たせることをお勧めします。一括取り込みデータベースは、プライマリキーの代わりに一意のインデックスを許可しません。プライマリキーが指定されていない場合、一括取り込みデータベースはすべてのカラムをプライマリキーの一部であるかのように扱います。例外: SQL Server クエリベースの CDC の場合、各ソーステーブルにプライマリキーが必要です。
- トランザクションログを使用するログベースの CDC の場合、一括取り込みデータベースには、ソースデータベースに対する読み取り/書き込みアクセス権が必要です。SQL Server Always On 可用性グループを使用する場合、この要件が意味するのは、一括取り込みデータベースが読み取り/書き込みプライマリレプリカから変更データをキャプチャできるが、読み取り専用セカンダリレプリカからはキャプチャできないことです。
- Microsoft SQL Server ソースで Always Encrypted 方式を使用してカラムデータを暗号化している場合、データベース取り込みタスクの［**ソース**］ページの［**CDC スクリプト**］フィールドから生成された CDC スクリプトは実行できません。この問題は、SQL Server の制限が原因で発生します。この問題は、Transparent Data Encryption（TDE）では発生しません。
- 一括取り込みデータベースは、データベース取り込み増分ロードジョブで Microsoft SQL Server ソースのスキーマドリフトオプションをサポートします。次の制限が適用されます。
  - Microsoft SQL Server は、変更データキャプチャ（CDC）が有効になっているテーブルとカラムの名前変更をサポートしていません。
  - Microsoft SQL Server は、CDC テーブルのプライマリキーの変更をサポートしていません。
  - 一括取り込みデータベースが CDC テーブルから変更データを直接読み取った場合、作成後に CDC テーブルが変更されることはありません。ソーステーブルで発生した DDL の変更は、CDC テーブルに NULL としてレプリケートされます。DDL の変更を CDC テーブルにレプリケートするには、タスクウィザードの［**ソース**］ページで pwx.custom.sslr_cdc_manage_instances カスタムプロパティを 1 に設定します。このカスタムプロパティを使用すると、CDC テーブルを変更してソーステーブルの DDL の変更を反映し、DML キャプチャを強化することができます。CDC テーブルのアクティブな管理を有効にするには、db_owner ロールが必要です。
- ソーステーブルのパーティションの変更により行セット ID が変更された場合、一括取り込みデータベースは変更を処理して、データベース取り込みジョブがテーブルから DML 変更をキャプチャし続けることができるようにします。
- クエリベースの CDC メソッドを使用した増分ロードおよび初期ロードと増分ロードの組み合わせジョブには、次の制限が適用されます。
  - 選択したソーステーブルごとにプライマリキーが必要です。ソーステーブルにプライマリキーが存在しない場合、変更データキャプチャはテーブルを無視し、選択されたソーステーブルの残りの処理を続行します。どのソーステーブルにもプライマリキーがない場合、ジョブは失敗します。
  - クエリベースの CDC は、削除操作をキャプチャしません。
  - 挿入と更新の操作はすべて更新/挿入として扱われ、監視インタフェースに表示され、更新としてログに記録されます。
  - ラージオブジェクト（LOB）カラムからのデータレプリケーションはサポートされていません。ソーステーブルに LOB カラムが含まれている場合、一括取り込みデータベースはこれらのカラムに対して NULL をプロパゲートします。
  - 特定のサイクルの開始時に夏時間またはタイムゾーンの変更が検出された場合、またはジョブが失敗状態または停止状態から再開されたときに、一括取り込みデータベースは再開してそのサイクルで発生した変更を処理します。
- タスクウィザードの［**ソース**］ページの［**詳細**］で［**LOB を含める**］を選択した場合、データベース取り込みジョブで、Microsoft SQL Server のラージオブジェクト（LOB）カラムからデータをレプリケートできます。

サポートされるターゲットタイプは、ロードタイプによって異なります。

- 初期ロードジョブの場合: Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Microsoft Azure Data Lake Storage Gen2、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、Snowflake、および SQL Server。
- 増分ロードジョブの場合: Azure Event Hubs、Databricks Delta、Snowflake、および SQL Server。
- 初期ロードと増分ロードの組み合わせジョブの場合: Databricks Delta、Snowflake、および SQL Server。

LOB データ型は、GEOGRAPHY、GEOMETRY、IMAGE、VARBINARY(MAX)、VARCHAR(MAX)、NVARCHAR(MAX)、TEXT、NTEXT、および XML です。LOB データは、ターゲットに書き込まれる前に切り詰められる場合があります。切り詰めポイントは、データ型、ターゲットタイプ、およびロードタイプによって異なります。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

- 一括取り込みデータベースは、SQL Server の永続化されていない計算カラムからのデータをレプリケートしません。ログベースまたはクエリベースの CDC メソッドを使用する初期ロードジョブ、増分ロードジョブ、および初期ロードと増分ロードの組み合わせジョブの場合、永続化された計算カラムはターゲットにレプリケートされます。CDC テーブルからのみ変更をキャプチャする増分ロードジョブ、および初期ロードと増分ロードの組み合わせジョブの場合、永続化された計算カラムは、カラムが NULL 値を許容するかどうかに応じて、NULL または空の値としてレプリケートされます。
- SQL Server ソースと SQL Server ターゲットを持ち、sql_variant ソースカラムを含むデータベース取り込みの初期ロードジョブは、ターゲット上で sql_variant データを 16 進形式に変換します。データを 16 進形式から varbinary 形式に変換するには、次のクエリを実行します。

  `SELECT <column_name>, CONVERT(varbinary,<column_name>) from <table_name>;`

  <*column_name*>と<*table_name*>を実際のターゲットカラムとテーブル名に置き換えます。
- SQL Server Hierarchyid データ型は、データベース統合増分ロード、および Snowflake ターゲットを持つ初期ロードジョブと増分ロードジョブの組み合わせではサポートされていません。一括取り込みデータベースは、このデータ型を持つカラムに対して null をプロパゲートします。詳細については、「デフォルトデータ型のマッピング」 (ページ 169)を参照してください。
- SQL Server ソースを持ち、複数のエージェントが含まれている Secure Agent グループを使用するデータベース取り組み増分ロードジョブおよび初期ロードと増分ロードの組み合わせジョブは、アクティブなエージェントの実行が停止すると、次の制限に従ってグループ内の別のエージェントに切り替えることができます。
  - ジョブに Kafka ターゲットを含めることはできません。
  - ジョブで永続ストレージを有効にすることはできません。
  - ジョブでは、クエリベースの CDC メソッドを使用して、タイムスタンプカラムをクエリして変更をキャプチャすることはできません。
  - 切り替えるには、ジョブを停止して再開する必要があります。
- SQL Server ソースが Always On 可用性グループ内にある場合、データベース取り込み増分ロードジョブおよび組み合わせロードジョブは、プライマリノードまたはセカンダリノードのトランザクションログまたは CDC テーブルから変更データをキャプチャできます。また、ノードが使用できなくなった場合、可用性グループリスナを指すように SQL Server 接続を構成していれば、データベース取り込みジョブは可用性レプリカ内のプライマリデータベースまたはセカンダリデータベースにフェールオーバーして処理を続行できます。可用性グループリスナーは、SQL Server 物理インスタンス名を知らなくても、一括取り込みデータベースが可用性グループの可用性レプリカ内のデータベースにアクセスするために使用できる仮想ネットワーク名（VNN）です。

- SQL Server ソースを使用したデータベース取り込みジョブの実行後、レプリケーション用に追加のソースカラムを選択してタスクを再デプロイした場合、ジョブは追加のカラムを使用してターゲットテーブルをすぐに再作成したり、それらのデータをレプリケートしたりしません。ただし、増分ロードジョブ、または初期ロードと増分ロードの組み合わせジョブでは、スキーマドリフトの**［カラムの追加］**オプションを**［レプリケート］**に設定した場合、次の新しい DML 変更レコードを処理するときに、新しく選択したカラムがターゲットに追加され、データがレプリケートされます。初期ロードジョブでは、次回のジョブ実行時に、新しく選択したカラムがターゲットに追加され、データがレプリケートされます。

SQL Server ソースの変更キャプチャメカニズム

一括取り込みデータベースは、SQL Server ソースから変更データをキャプチャしてそのデータをターゲットに適用するための、複数の変更キャプチャメカニズムを提供します。

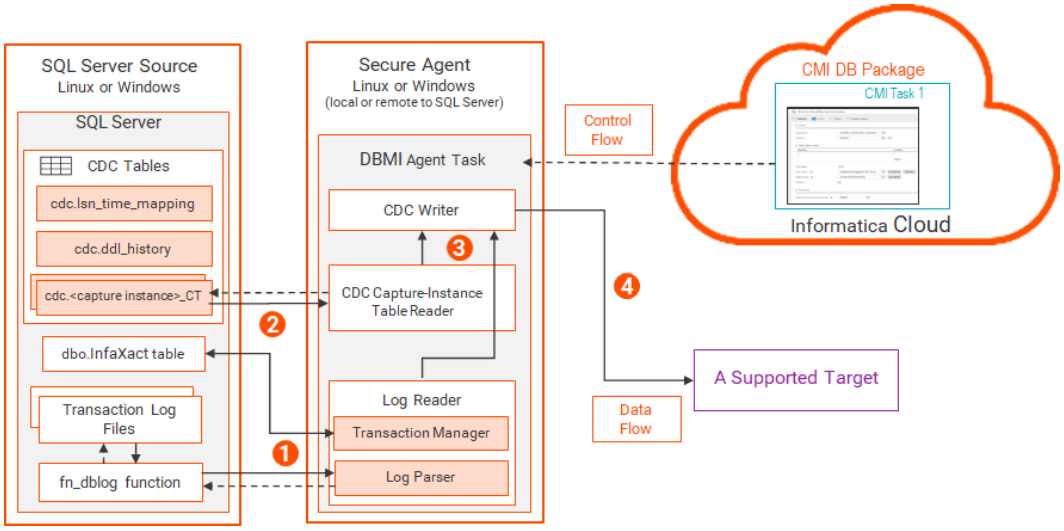
トランザクションログと CDC テーブルを使用するログベースの変更データキャプチャ

ログベースの CDC を使用するデータベース取り込み増分ロードジョブは、アクティブなトランザクションログからレコードを解析し、CDC テーブルから変更レコードを直接読み取ることで、DML と DDL の変更をキャプチャします。必要な再開ポイント（LSN）が利用可能な場合、変更データはアクティブなトランザクションログから読み取られます。キャプチャ開始点がトランザクションログ内のレコードより前の日付である場合、またはその他の特定の状況では、一括取り込みデータベースは自動的に CDC テーブルからの変更データの読み取りに移行します。CDC テーブルから変更を読み取った後、一括取り込みデータベースは、トランスペアレントな方法でアクティブなトランザクションログの読み取りに戻ります。

ログベースの変更キャプチャ処理には、次のコンポーネントが関わります。

- Informatica Intelligent Cloud Services Secure Agent。SQL Server インスタンスに対してローカルまたはリモートの Linux ボックスまたは Windows ボックスで実行できます。
- dbo.$InfaXact テーブル。オープントランザクションに関連付けられ、現在実行中のキャプチャジョブに関する情報を一時的に格納します。テーブルが存在しない場合は、変更キャプチャプロセスによって作成されます。
- ログリーダーとそのサブコンポーネント。トランザクションログから DML および DDL 変更レコードを解析して読み取るために必要です。
- SQL Server fn_dblog()関数。ログリーダーが呼び出しをループして、ソースデータベースのトランザクションログファイルのアクティブな部分からログレコードを取得します。
- SQL Server CDC テーブル: cdc.lsn_time_mapping、cdc.ddl_history、および cdc.<*capture instance*>_CT。ソースデータベースで CDC が有効になっている場合、SQL Server はこれらのテーブルを CDC スキーマに作成します。1 つまたは 2 つの cdc.<*capture instance*>_CT テーブル。CDC が有効になっているソーステーブルごとに作成され、ネイティブログリーダーによってキャプチャされた DML 変更を格納します。
- CDC キャプチャインスタンステーブルリーダー。cdc.<*capture_instance*>_CT テーブルから変更レコードを読み取ります。
- CDC ライター。変更をターゲットに書き込みます。

次の図は、SQL Server のログベースの変更データキャプチャコンポーネントとデータフローを示しています。



1. ログリーダープロセスは、ログレコードを読み取り、コミットされたトランザクションの DML および DDL の変更をキャプチャします。

   - トランザクションマネージャサブコンポーネントは、オープントランザクションに関連付けられた dbo.$InfaXact テーブルと対話します。
   - ログパーサーサブコンポーネントは、fn_dblog()関数の呼び出しをループして、CDC が有効になっている選択されたソーステーブルのアクティブなトランザクションログからログレコードを読み取ります。

   キャプチャプロセスは、コミットされたトランザクションが完了するか、致命的なエラーによってキャプチャプロセスが停止または中断されるか、cdc.<*capture_instance*>_CT テーブルからの変更レコードの読み取りへの切り替えがトリガーされるまで続行されます。
2. 特定の状況では、一括取り込みデータベースは、CDC キャプチャインスタンステーブルリーダーを使用して、cdc.<*capture instance*>_CT テーブルからの変更の読み取りに自動的に切り替わります。
   以下の条件下では、処理は、cdc.<*capture instance*>_CT テーブルに切り替わります。

   - 開始 LSN がアクティブなトランザクションログの LSN よりも前であり、cdc.<*capture instance*>_CT テーブル内に存在する場合、キャプチャプロセスの初期化および開始中である。
   - トランザクションログの切り詰めが発生し、データ損失が発生した。通常、SQL Server は、オープントランザクションが dbo.$InfaXact に関連付けられている場合、ログの切り詰めを防止します。ただし、ネットワーク接続が失われるとトランザクションが終了し、ログが切り詰められる可能性があります。

     注: 定期的なログバックアップでも、トランザクションログが切り詰められる可能性があります。データの損失を防ぐために、dbo.$InfaXact テーブルを使用するトランザクションは、アクティブなトランザクションログをロックします。
   - 行外 LOB またはプライマリ行レコードは切り詰められます。この状況では、ログリーダーは cdc.<*capture instance*>_CT テーブルからカラム情報を読み取ることにより、ログレコードに選択的にパッチを適用します。

     注: fn_dblog()関数から読み取られたレコードは、8000 バイトに切り詰められます。
3. ログリーダーと CDC キャプチャインスタンステーブルリーダーは、変更レコードを CDC ライターに送信します。
4. CDC ライターは変更レコードをフォーマットし、ターゲットに適用します。

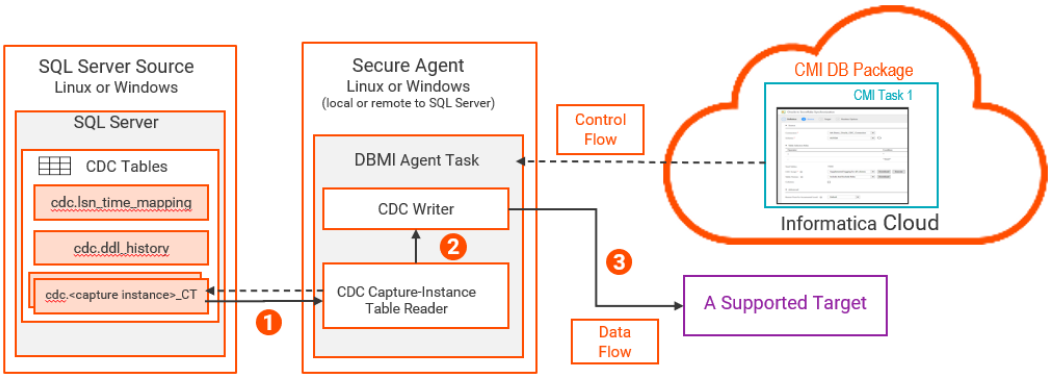
CDC テーブルのみを使用した変更キャプチャ

データベース取り込みの増分ロードジョブは、トランザクションログを使用せずに SQL Server CDC テーブルから直接変更をキャプチャできるようになりました。

CDC テーブルのみを使用する変更キャプチャ処理には、次のコンポーネントが関わります。

- Informatica Intelligent Cloud Services Secure Agent。SQL Server インスタンスに対してローカルまたはリモートの Linux ボックスまたは Windows ボックスで実行できます。
- SQL Server CDC テーブル: cdc.lsn_time_mapping、cdc.ddl_history、および cdc.<*capture instance*>_CT。ソースデータベースで CDC が有効になっている場合、SQL Server はこれらのテーブルを CDC スキーマに作成します。1 つまたは 2 つの cdc.<*capture instance*>_CT テーブル。CDC が有効になっているソーステーブルごとに作成され、ネイティブログリーダーによってキャプチャされた DML 変更を格納します。
- CDC キャプチャインスタンステーブルリーダー。cdc.<*capture_instance*>_CT テーブルから変更レコードを読み取ります。
- CDC ライター。変更をターゲットに書き込みます。

次の図は、SQL Server のログベースの変更データキャプチャコンポーネントとデータフローを示しています。



1. CDC キャプチャインスタンステーブルリーダーは cdc から変更を読み取ります。<*capture instance*>_CT テーブル内に存在する場合、キャプチャプロセスの初期化および開始中である。
2. CDC キャプチャインスタンステーブルリーダーは変更レコードを CDC ライターに送信します。
3. CDC ライターは変更レコードをフォーマットし、ターゲットに適用します。

クエリベースの変更キャプチャ

データベース取り込みジョブは、変更が発生したときに更新されるタイムスタンプカラムを照会することで、ソースにおける挿入と更新の変更をキャプチャします。ソースの設定は、各ソーステーブルへの共通 CDC クエリカラムの追加に制限されます。クエリベースの CDC メソッドはクエリカラムを使用して、指定の CDC 間隔の開始以降に変更された行を識別します。

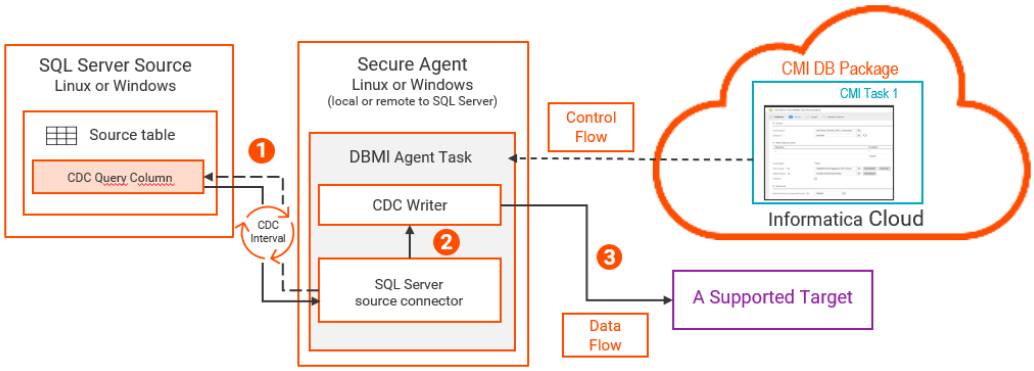
クエリベースの変更キャプチャを実装するには、タスクウィザードの **[ソース]** ページで次のオプションを設定します。

- **CDC メソッド。[クエリベースの]** を選択して、このキャプチャメソッドを有効にします。
- **CDC クエリカラム名。**ソーステーブルにおける CDC クエリカラムの、大文字小文字が区別された名前です。クエリカラムでサポートされている SQL Server データ型は、DATETIME と DATETIME2 です。カラムはソーステーブルに存在する必要があります。
- **CDC 間隔。**クエリベースの変更データキャプチャサイクルの頻度。デフォルトは 5 分です。
- **増分ロード操作の当初の開始点。**変更キャプチャサイクルを開始するポイント。デフォルトは **[使用可能な最新]** です。

CDC 間隔が経過した後、一括取り込みデータベースは、CDC クエリカラムを参照する WHERE 句を含む SQL クエリを使用して、CDC 間隔の間に変更を受け取った行を識別します。変更データがキャプチャされ、ターゲットに適用されます。

クエリベースの CDC 用に選択されたソーステーブルに CDC クエリカラムがない場合、変更データキャプチャはこれらのテーブルを無視し、残りのテーブルで処理を続行します。スキップされたテーブルの場合、ターゲットデータベースで生成された、対応するテーブルは空になります。どのソーステーブルにも CDC クエリカラムがない場合、ジョブは実行時に失敗します。

次の図は、SQL Server のクエリベースの変更データキャプチャコンポーネントとデータフローを示しています。



1. CDC 間隔が経過した後、一括取り込みデータベースは、CDC クエリカラムを使用して変更データを抽出する SQL クエリをソースデータベースで実行します。
2. 変更レコードが CDC ライターに送信されます。
3. CDC ライターは変更レコードをフォーマットし、ターゲットに適用します。

MongoDB ソース

データベース取り込みタスクで MongoDB ソースを使用するには、以下の考慮事項を確認してください。

ソースの準備:

一括取り込みデータベースは、MongoDB 変更ストリームを使用して、単一のコレクション、データベース、またはデプロイメント全体のリアルタイムのデータ変更にアクセスします。

増分ロードジョブ、および初期ロードジョブと増分ロードジョブの組み合わせで変更データキャプチャを実行するには、読み取りロールで十分です。読み取りロールには、MongoDB 変更ストリームへのアクセス権がすでにあります。

使用に関する考慮事項:

- データベース統合タスクは、MongoDB データをキーと値のペアとしてターゲットに移動します。ここで、キーは ObjectID であり、値は BSON ドキュメントを構成する JSON 文字列です。
- MongoDB ソースの場合、データ型のマッピングは行われません。すべてのデータは、文字列データとしてターゲットに保持されます。
- 増分ロード操作では、ソースでのデータ変更は一意のキー（ObjectID）によって追跡され、変更された同じ JSON 文字列がターゲットに適用されます。
- 一括取り込みデータベースは、MongoDB ソースを持つ増分ロードジョブの時系列コレクションをサポートしていません。
- 増分ロード操作では、一括取り込みデータベースは、再開ポイントとして指定された日時から変更レコードを取得します。MongoDB ソースの場合、再開ポイントのデフォルト値は現在の時刻です。別の日時をグリニッジ標準時（GMT）で指定できます。

- MongoDB ソースでスキーマドリフトが発生した場合、ターゲットに送信される BSON ドキュメントのデータにはスキーマの変更が反映されます。ただし、一括取り込みデータベースはスキーマの変更を具体的に検出して報告するわけではありません。

MySQL ソース

データベース取り込みタスクで MySQL ソースを使用するには、最初にソースデータベースを準備し、使用に関する考慮事項を確認してください。

ソースの準備:

- MySQL ソースを含むデータベース統合タスクをデプロイして実行するには、ソース接続で、必要な特権を持つデータベースユーザーを指定する必要があります。次の SQL 文を使用して、これらのユーザーに特権を付与します。

  ```
  GRANT SELECT ON database_name.* TO 'user_name'@'%';
  GRANT SELECT TABLES ON database_name.* TO 'user_name'@'%';
  ```

  増分ロードジョブの場合、次の追加の特権をユーザーに付与します。

  ```
  /* To fetch table and column details from system tables */
  GRANT SELECT ON 'sys'.* TO 'user_name'@'%';

  /* To allow the user to monitor binary log information such as file name, position, and GTID */
  GRANT REPLICATION CLIENT ON *.* TO 'user_name;

  /* Required for a multi-node MySQL cluster with master and slave configuration */
  GRANT REPLICATION SLAVE ON *.* TO 'user_name;
  ```

- 増分ロードジョブの場合、my.cnf ファイルの［mysqld］セクションで default_storage_engine 変数を InnoDB に設定します。次に、MySQL サーバーを再起動します。default_storage_engine 設定を確認するには、次の文を使用します。

  ```
  SHOW VARIABLES LIKE '%engine%';
  ```

  出力で、default_storage_engine 変数が InnoDB に設定されていることを確認します。

- 一括取り込みデータベースは、ソースで変更イベントをキャプチャするために、MySQL によって生成されたバイナリログファイルを使用します。binlog は、MySQL サーバーインスタンスに対して行われたデータ変更に関する情報を含む一連のログファイルです。
  バイナリログを有効にするには、--log-bin オプションを使用してサーバーを起動するか、または my.cnf ファイルでキーと値の log-bin="[HostName]-bin"設定を使用します。[HostName]は使用するホストの名前に置き換えてください。次に、MySQL サーバーを再起動します。バイナリログが有効になっていることを確認するには、次の文を使用します。

  ```
  SHOW VARIABLES LIKE 'log_bin';
  ```

  出力で、log_bin 変数が ON に設定されていることを確認します。

- 増分ロードジョブの場合、次の文を使用して行ベースのログを有効にします。

  ```
  SET GLOBAL binlog_format = 'ROW';
  ```

  行ベースのログが有効になっていることを確認するには、次の文を使用します。

  ```
  SHOW VARIABLES LIKE 'binlog_format';
  ```

  出力で、binlog_format システム変数が ROW に設定されていることを確認します。

- 一括取り込みデータベースは、次のいずれかの方法で binlog ファイルを読み取ることができます。

  - グローバルトランザクション ID（GTID） - MySQL GTID モードを有効にすると、MySQL のすべてのトランザクションに、トランザクションを一意に識別するための GTID が割り当てられます。マルチクラスタ環境では GTID モードを使用してください。

- binlog ファイルの名前と位置 - MySQL のすべてのトランザクションは匿名として保存され、binlog ファイルの名前と位置を使用して取得されます。MySQL GTID モードが有効になっている場合、またはマルチクラスタ環境を使用している場合は、この方法を使用しないでください。マルチクラスタ環境では、フェイルオーバーが発生すると、binlog ファイルの位置が変化し、データの一貫性が失われる可能性があります。

GTID モードを有効にするには、各 MySQL サーバーで次の文を使用します。

```
SET @@GLOBAL.ENFORCE_GTID_CONSISTENCY = WARN;
SET @@GLOBAL.ENFORCE_GTID_CONSISTENCY = ON;
SET @@GLOBAL.GTID_MODE = OFF_PERMISSIVE;
SET @@GLOBAL.GTID_MODE = ON_PERMISSIVE;
```

各サーバーで、ステータス変数'Ongoing_anonymous_transaction_count'が 0（ゼロ）になるまで待ちます。次の文を使用して、ステータス変数の値を確認できます。

```
SHOW STATUS LIKE 'Ongoing_anonymous_transaction_count';
```

カウントが 0 の場合、次の文を使用して GTID モードを有効にします。

```
SET @@GLOBAL.GTID_MODE = ON;
```

- 次のロードタイプとソースエディションの組み合わせを使用する場合、MySQL ソースに接続するには、MySQL ドライバファイルをダウンロードして特定のインストールサブディレクトリにコピーする必要があります。

  - MySQL Community Edition または MySQL Enterprise Edition ソースを持つ増分ロードジョブ

  - MySQL Community Edition ソースを持つ初期ロードジョブ

  - Amazon Relational Database Service (RDS) for MySQL ソースを持つ初期ロードジョブ

  注: MySQL Enterprise Edition ソースを持つ初期ロードジョブのみを実行する場合は、ドライバをダウンロードする必要はありません。

  MySQL JDBC ドライバファイル、mysql-connector-java-<version>.jar を MySQL Community Downloads Web サイトからダウンロードし、次のディレクトリにコピーします。

  <*Secure_Agent_installation_directory*>/ext/connectors/thirdparty/com.mysql/

  接続プロパティを定義した後で Administrator で接続をテストできるようにする場合は、Secure Agent のシステム構成の詳細で Data Integration Server サービスの MySQL_JDBC_DRIVER_JARNAME パラメータも設定する必要があります。テスト後、パラメータを削除できます。このパラメータは、接続を使用してデータベース取り込みタスクを作成したり、関連するジョブを実行する場合には使用されません。

使用に関する考慮事項:

- 一括取り込みデータベースは、初期ロード、増分ロード、および初期ロードと増分ロードの組み合わせジョブに対する MySQL、Amazon Aurora MySQL、Cloud SQL for MySQL、および RDS for MySQL ソースをサポートしています。
- 一括取り込みデータベースでは、ソーステーブルの各行が一意であることを想定しているため、各ソーステーブルにプライマリキーを持たせることをお勧めします。一括取り込みデータベースは、プライマリキーの代わりに一意のインデックスを許可しません。プライマリキーが指定されていない場合、一括取り込みデータベースはすべてのカラムをプライマリキーの一部であるかのように扱います。
- ターゲットテーブルに存在しないレコードに対し、MySQL ソーステーブルのプライマリキー値を更新すると、そのレコードはターゲットにレプリケートされません。ただし、監視インタフェースは更新カウントを増分してプライマリキーの更新を含めます。プライマリキー値の更新前にターゲットテーブルにレコードがすでに存在する場合、データはターゲットにレプリケートされます。
- SET または ENUM データ型の MySQL ソースカラムを含むデータベース取り込みジョブは、SET カラムと ENUM カラムのデータを数値としてターゲットにレプリケートします。初期ロードジョブの場合は、タスクウィザードの **[ソース]** ページで mysql.set.and.enum.as.numeric カスタムプロパティを false に設定して、SET または ENUM データを文字列または varchar 形式でレプリケートすることができます。デフォルト値は true で、これにより、一括取り込みデータベースは SET または ENUM データを数値としてレプリケートします。

注: 増分ロードジョブの場合は、SET または ENUM カラムデータの数値表現と文字列表現または varchar 表現を切り替えることはできません。mysql.set.and.enum.as.numeric カスタムプロパティを false に設定し、初期ロードジョブを実行した後に増分ロードジョブを実行すると、一括取り込みデータベースは SET および ENUM データを数値のみとしてターゲットにレプリケートします。

- 一括取り込みデータベースでは、次の MySQL データ型はサポートされません。

  - BLOB
  - JSON
  - LONGBLOB
  - LONGTEXT
  - MEDIUMBLOB
  - MEDIUMTEXT
  - TEXT
  - TINYBLOB
  - TINYTEXT

  データベース取り込みタスクで、JSON データ型のカラムを含むソーススキーマを指定する場合、タスクをデプロイすると、JSON カラムが無視され、ターゲットに対応するカラムは作成されません。その他のサポートされないデータ型の場合、データベース取り込みジョブは null をプロパゲートします。

  サポートされているソースデータ型からターゲットデータ型へのデフォルトのマッピングについては、「デフォルトデータ型のマッピング」 (ページ 169)を参照してください。

Netezza ソース

データベース取り込みタスクで Netezza ソースを使用するには、最初にソースデータベースを準備し、使用に関する考慮事項を確認してください。

ソースの準備:

- Netezza JDBC ドライバのダウンロードとインストール
  1. Netezza JDBC ドライバを IBM Web サイトからダウンロードします。
  2. Netezza　JDBC ドライバ jar ファイルの nzjdbc.jar を次のディレクトリにコピーします。

     `<Secure Agent installation directory>/apps/Database_Ingestion/ext/`
  3. Secure Agent を再起動します。
- Netezza ソースを含むデータベース取り込みタスクをデプロイして実行するには、ソース接続で初期ロード操作を実行するために必要な特権を持つデータベースユーザーを指定する必要があります。Netezza ユーザーアカウントに以下のシステムビューに対する SELECT 権限を設定します。

  - _V_JDBC_SCHEMA1
  - _V_JDBC_SCHEMA3
  - _V_ODBC_TABLES3
  - _V_ODBC_COLUMNS3
  - _V_ODBC_PRIMARYKEYS3

使用に関する考慮事項:

- 一括取り込みデータベースでは、次の Netezza データ型はサポートされません。

  - ST_GEOMETRY

データベース統合ジョブは、このデータ型を持つカラムにはデプロイしたり null をプロパゲートしたりすることはできません。

サポートされているソースデータ型からターゲットデータ型へのデフォルトのマッピングについては、「デフォルトデータ型のマッピング」 (ページ 169)を参照してください。

Oracle ソース

データベース取り込みタスクで Oracle ソースを使用するには、最初にソースデータベースを準備し、使用に関する考慮事項を確認してください。

ソースの準備:

- Secure Agent が実行されている Linux または Windows システムで次のシステム環境変数を定義します。
  - ORACLE_HOME 環境変数。Windows の bin ディレクトリまたは Linux の lib ディレクトリの 1 つ上のレベルにある、Oracle クライアントインストールディレクトリを指します。この環境変数は必須ではありません。ただし、これを定義しない場合は、サブディレクトリへのパスを指定する他の環境変数または Secure Agent プロパティを定義するときに、Oracle クライアントの完全なインストールパスを指定する必要があります。
  - TNS_ADMIN 環境変数。Oracle データベース取り込み接続プロパティの **[データベース接続文字列]** プロパティで TNS 名を指定している場合、ファイルがデフォルトの$ORACLE_HOME/network/admin ディレクトリにないときは、この環境変数を使用して tsnnames.ora ファイルのディレクトリの場所を指すようにします。tnsnames.ora ファイルは、Oracle ソースデータベースと通信するために、Oracle Call Interface（OCI）とともに使用されます。

  注: Administrator で、データベース取り込みエージェントサービス（DBMI エージェント）の ociPath プロパティを、oci.dll または libclntsh.so ファイルを含む OCI ライブラリを指すように設定できます。OCI ライブラリは、データベース取り込み CDC タスクによって Oracle に接続するために使用されます。Oracle は、Linux では$ORACLE_HOME/lib、Windows では%ORACLE_HOME%\bin の ociPath 値をデフォルトで使用します。
- 一括取り込みデータベースユーザーに、データベース取り込みロードタイプを実行するために必要な Oracle 権限があることを確認してください。

  注: ログベースの CDC を使用した初期ロードと増分ロードの組み合わせの場合は、選択したソーステーブルごとに GRANT FLASHBACK 特権が発行されていることを確認するか、ANY TABLE オプションを使用します。一括取り込みデータベースは、SELECT AS OF *scn* 文で構成される Oracle Flashback Query を使用して、Oracle データベースのソーステーブルの行データをクエリします。Oracle では、このクエリを使用するには GRANT FLASHBACK 特権が必要です。

  詳細については、「Oracle 特権」 (ページ 45)を参照してください。
- ログベースの CDC を使用したデータベース統合ジョブには、増分変更データを読み取るために Oracle のオンライン REDO ログとアーカイブ REDO ログへの読み取りアクセスが必要です。REDO ログが、Secure Agent が実行されているオンプレミスシステムからリモートにある場合は、ログへの読み取りアクセスが提供されていることを確認してください。例えば、Oracle Automatic Storage Management（ASM）を使用して、ログをネットワークファイルシステム（NFS）にマウントするか、または Oracle ファイルシステム上にあるログへの BFILE アクセスを設定することによって、それを実現します。
- Oracle ASM の REDO ログファイルからデータを読み取る予定の場合、Informatica では、ローカルの sqlnet.ora ファイルの sqlnet.recv_timeout パラメータを 5 分未満に設定することをお勧めします。このパラメータは、クエリがタイムアウトになるまでに Oracle クライアントが ASM からの応答を待機する時間を指定します。ネットワークの中断やその他の要因により、Oracle 接続が応答しなくなることがあります。この値を設定すると、リーダーがそのような状況に適時応答してリカバリできるようになります。
- Oracle 11.2.04 を使用する場合は、Oracle COMPATIBLE 初期化パラメータを 11.2.04 に設定して、そのリリースの最新の Redo ログ修正がすべて Oracle に適用されるようにします。

- Secure Agent が Oracle と通信できるように、Oracle Database Client または Instant Client が Secure Agent サーバーにインストールされ、設定されていることを確認します。Oracle クライアントをまだインストールしていない場合は、Oracle Web サイトからクライアントをダウンロードしてインストール情報にアクセスするか、Oracle DBA に Oracle クライアントのダウンロードと設定を依頼してください。
- ログベースの CDC を使用した増分ロードまたは初期ロードと増分ロードの組み合わせ操作の場合、Oracle で次の要件タスクを実行します。

  - Oracle データベースの ARCHIVELOG モードを有効にします。データベースが Amazon RDS 環境にない場合は、次の SQL 文を発行します。

    ```
    SHUTDOWN IMMEDIATE;
    STARTUP MOUNT;
    ALTER DATABASE ARCHIVELOG;
    ALTER DATABASE OPEN;
    SHUTDOWN IMMEDIATE;
    STARTUP;
    ```

    Amazon RDS for Oracle データベースの場合、データベースを ARCHIVELOG モードにして、自動バックアップを有効にするために、バックアップの保存期間を設定します。

  - ログのアーカイブ先を定義します。

  - ソースデータベースで Oracle の最小限のグローバルサプリメンタルロギングを有効にします。

  - Oracle ソーステーブルにプライマリキーがある場合は、すべてのプライマリキーカラムに対してサプリメンタルロギングが有効になっていることを確認してください。プライマリキーのないソーステーブルの場合、変更データがキャプチャされるすべてのカラムでサプリメンタルロギングが有効になっていることを確認してください。

    注: データベース取り込みタスクを作成するときに、選択したソーステーブルのすべてのカラムまたはプライマリキーカラムのみのサプリメンタルロギングを実装するスクリプトを生成するオプションがあります。

  - Oracle MAX_STRING_SIZE 初期化パラメータが EXTENDED に設定されて*いない*ことを確認してください。EXTENDED に設定されている場合、一括取り込みデータベースは、大きな（拡張サイズ）VARCHAR2、NVARCHAR2、または RAW カラムで定義されたカラムを含むテーブルの挿入と更新をレプリケートできません。

  これらのタスクを実行する権限がない場合は、Oracle データベース管理者に実行を依頼してください。詳細については、Oracle のマニュアルを参照してください。

- クエリベースの CDC を使用する増分ロード操作の場合、ソーステーブルには、変更行を示すために使用される CDC クエリカラムを含める必要があります。データベース統合タスクを作成する前に、クエリカラムをソーステーブルに追加する必要があります。クエリカラムでサポートされる Oracle データ型は TIMESTAMP です。
  クエリベースの CDC 用に選択されたソーステーブルに CDC クエリカラムがない場合、変更データキャプチャはこれらのテーブルを無視し、残りのテーブルで処理を続行します。スキップされたテーブルの場合、ターゲットデータベースで生成された、対応するテーブルは空になります。どのソーステーブルにも CDC クエリカラムがない場合、ジョブのデプロイは失敗します。

Oracle ソース準備用の Amazon Relational Database Service（RDS）:

1. RDS ファイルシステム上にオンライン REDO ログとアーカイブ REDO ログをそれぞれ保持する ONLINELOG_DIR ディレクトリおよび ARCHIVELOG_DIR のディレクトリを作成します。次の実行文を使用します。

   ```
   exec rdsadmin.rdsadmin_master_util.create_archivelog_dir;
   exec rdsadmin.rdsadmin_master_util.create_onlinelog_dir;
   ```

2. Amazon RDS for Oracle ソースタイプに必要な Oracle 特権を一括取り込みデータベースユーザーに付与します。
   Amazon RDS for Oracle ソースに必要な特権の詳細については、「Amazon RDS for Oracle ソースに対する Oracle 特権」 (ページ 47)を参照してください。

3. アーカイブ REDO ログの適切な保持時間を定義します。次の実行文を使用します。

   ```
   exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention days',number_of_days);
   ```

4. Amazon RDS コンソールで、データベースインスタンスの自動バックアップを有効にするために、ソースデータベースのバックアップ保持期間を 0 より大きい値に設定します。

   注: この手順では、データベースに対して ARCHIVELOG モードを有効にします。

5. データベースレベルでサプリメンタルロギングが有効になっていることを確認します。次の文を使用します。

   ```
   exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD');
   ```

   データベース取り込みタスクを作成するときに、選択したソーステーブルのサプリメンタルロギングを有効にするスクリプトを生成できます。

6. オプションで、Amazon RDS コンソールでは、パラメータグループを作成し、デフォルトのバッファープールのキャッシュサイズを定義することができます。デフォルトのバッファプールは、プライマリデータベースのブロックサイズを使用するバッファを保持します。次の DB_CACHE_SIZE パラメータ値を使用します。

   - DB_2K_CACHE_SIZE
   - DB_4K_CACHE_SIZE
   - DB_16K_CACHE_SIZE
   - DB_32K_CACHE_SIZE

   次に、ソースデータベースのパラメーターグループを選択します。

使用に関する考慮事項:

- Oracle ソースを使用する増分ロード操作の場合、一括取り込みデータベースは、ソースから変更データをキャプチャし、そのデータをターゲットに適用するための代替キャプチャメソッドを提供します。
  使用可能な変更キャプチャメソッドは次のとおりです。
  - ログベースの変更データキャプチャ。一括取り込みデータベースは、Oracle REDO ログからデータ変更を読み取ります。この方法では、ユーザーの権限を拡張する必要があります。
  - クエリベースの変更データキャプチャ。変更データキャプチャでは、共通の CDC クエリカラムを参照する WHERE 句を含む SQL ステートメントを使用して、挿入および更新の変更がある行を識別します。ソースデータベースの設定は、各ソーステーブルへの CDC クエリカラムの追加に制限されます。ユーザーは、少なくともソーステーブルへの読み取り専用アクセス権を持つ必要があります。
- 一括取り込みデータベースでは、ソーステーブルの各行が一意であることを想定しているため、各ソーステーブルにプライマリキーを持たせることをお勧めします。一括取り込みデータベースは、プライマリキーの代わりに一意のインデックスを許可しません。プライマリキーが指定されていない場合、一括取り込みデータベースはすべてのカラムをプライマリキーの一部であるかのように扱います。例外: Oracle クエリベースの CDC の場合、各ソーステーブルにプライマリキーが必要です。
- マルチテナントアーキテクチャを使用する Oracle ソースの場合、ソーステーブルはマルチテナントコンテナデータベース（CDB）内の単一のプラガブルデータベース（PDB）に存在する必要があります。
- Oracle Transparent Data Encryption（TDE）を使用して、増分ロード処理のための Oracle ソーステーブルを含むテーブルスペース内のデータを暗号化できます。一括取り込みデータベースは、ファイルシステム、ASM、または Oracle Key Vault（OKV）など PKCS11 インタフェースを提供する外部ハードウェアセキュリティモジュール（HSM）にある TDE キーストアへのマスター暗号化キーの保存をサポートします。詳細については、Informatica グローバルカスタマサポートにお問い合わせください。
- Oracle ソースの CHAR カラムまたは VARCHAR カラムに null が含まれている場合、データベース統合ジョブは、Amazon S3、フラットファイル、Microsoft Azure Data Lake、または Microsoft Azure Synapse Analytics ターゲットにデータを書き込むときに、null 値を二重引用符（"）マークまたはその他の区切り文字で区切りません。

- 一括取り込みデータベースは、Oracle Data Guard の論理および物理スタンバイデータベースおよび Far Sync インスタンスをソースとしてサポートします。詳細については、「ソースとしての Oracle Data Guard データベースまたは Far Sync インスタンス」 (ページ 55)を参照してください。
- 一括取り込みデータベースは、RESETLOGS 境界を越えてデータを処理できます。ソースとターゲットが同期しなくなるのを避けるため、RESETLOGS を実行する前にキャプチャ処理を停止し、RESETLOGS イベントの後にキャプチャ処理を再開することをお勧めします。そうしないと、キャプチャプロセスによってデータがターゲットに送信され、その後 RESETLOGS イベントによって元に戻されて、ソースとターゲットが同期しなくなる可能性があります。
- Oracle REDO ログにアクセスするための別の戦略を利用できます。詳細については、「CDC の Oracle ログアクセス方法」 (ページ 51)を参照してください。
- データベース取り込み増分ロードタスクまたは初期ロードと増分ロードの組み合わせタスクに、30 文字を超える Oracle ソーステーブル名または 1 つ以上のカラム名が含まれている場合、Oracle ではプライマリキーと外部キーを含むテーブル全体の補足ログが抑制されます。その結果、テーブルに対するほとんどの操作が失敗します。この問題は、Oracle の制限が原因で発生します。この状況では、テーブルをキャプチャ処理から除外するか、長いテーブル名とカラム名を 30 文字以下の名前に変更してください。
- データベース取り込み初期ロードジョブで、Oracle BLOB、CLOB、および NCLOB カラムから Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Microsoft Azure Data Lake Storage Gen2、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、Snowflake、および SQL Server ターゲットにデータをレプリケートできます。BLOB、CLOB、または NCLOB カラムからデータをレプリケートするには、タスクを設定するときに、**[ソース]** ページの **[詳細]** で **[LOB を含める]** を選択する必要があります。LOB カラムデータは、LOB タイプとターゲットタイプによって異なるバイト制限よりもサイズが大きい場合、ターゲットに書き込まれる前に切り詰められます。詳細については、「ソースの設定」 (ページ 102)を参照してください。
- 一括取り込みデータベースでは、ターゲットタイプまたはロードタイプを含む次の Oracle ソースデータ型はサポートされません。

  - ANYTYPE、ANYDATA、ANYDATASET などの「ANY タイプ」
  - 拡張タイプ
  - INTERVAL
  - JSON
  - LOB（初期ロードジョブの BLOB、CLOB、および NCLOB を除く）CSV 出力形式を使用する Amazon S3、Google Cloud Storage、または Microsoft Azure Data Lake Storage Gen2 ターゲット以外のターゲットを持つ
  - TIMESTAMP WITH LOCAL TIME ZONE
  - UROWID
  - XMLTYPE、URI タイプ、URIFactory パッケージのサブタイプなどの XML 定義タイプ
  - SDO_GEOMETRY などの空間タイプ
  - OBJECT、REF、VARRAY、ネストされたテーブルタイプなどのユーザー定義タイプ

  サポートされていないデータ型を持つソースカラムは、ターゲット定義から除外されます。

  サポートされている Oracle データ型からターゲットタイプへのデフォルトのマッピングについては、「デフォルトデータ型のマッピング」 (ページ 169)を参照してください。
- Oracle ソースの RAW カラムをターゲットの CHAR または VARCHAR カラムにマッピングするときに、カスタムデータ型マッピングルールを使用しないようにしてください。カスタムデータ型マッピングルールを使用した場合、データベース取り込みタスクのデプロイメントが失敗する可能性があります。
- 一括取り込みデータベースでは、ターゲットタイプに関係なく、Oracle ソースカラムの非表示カラムはサポートされていません。これらのカラムについては、データベース統合増分ロードジョブと、初期ロードジョブと増分ロードジョブの組み合わせは、対応するターゲットカラムに null をプロパゲートします。

- ターゲットテーブルに存在しないレコードに対して Oracle ソーステーブルのプライマリキー値を更新すると、そのレコードはターゲットにレプリケートされません。ただし、監視インタフェースは更新カウントを増分してプライマリキーの更新を含めます。プライマリキー値の更新前にターゲットテーブルにレコードがすでに存在する場合、データはターゲットにレプリケートされます。
- Oracle テーブルへの更新によって既存のカラム値が変更されない場合、テーブルの監視詳細の更新カウントは引き続き増加しますが、更新行がターゲットに適用されることはありません。一括取り込みデータベースは、実際に値を変更しない更新行を無視します。また、ほとんどのデータベースターゲットにおいて、一括取り込みデータベースはターゲットに変更を書き込む前にマイクロバッチレベルで変更レコードの集計を行います。この場合、監視統計の更新カウントとターゲットに適用された行との間に不一致が発生する可能性もあります。
- テーブル名またはテーブルカラム名が 30 文字を超える場合、テーブルのサプリメンタルロギング設定が Oracle によって無視されることがあります。この場合、データベース取り込み増分ロードまたは組み合わせロードジョブの結果は予測できません。
- 一括取り込みデータベースは、Oracle ソースを持つジョブの派生カラムをサポートしていません。
- Oracle の初期ロードと増分ロードの組み合わせジョブの場合、Oracle Flashback クエリを使用して、変更ストリーム内の特定の時点で最新のコミット済みデータを取得します。初期ロード期間中にソーステーブルが切り詰められないようにしてください。切り詰めが発生した場合、フラッシュバッククエリ中に DDL 変更を実行すると、クエリは失敗します。
- Oracle ソースを持ち、複数のエージェントが含まれている Secure Agent グループを使用するデータベース取り組み増分ロードジョブおよび初期ロードと増分ロードの組み合わせジョブは、アクティブなエージェントの実行が停止すると、次の制限に従ってグループ内の別のエージェントに切り替えることができます。
  - ジョブに Kafka ターゲットを含めることはできません。
  - ジョブで永続ストレージを有効にすることはできません。
  - ジョブでは、クエリベースの CDC メソッドを使用して、タイムスタンプカラムをクエリして変更をキャプチャすることはできません。
  - 切り替えるには、ジョブを停止して再開する必要があります。
- クエリベースの CDC メソッドを使用した増分ロードジョブ、および初期ロードと増分ロードの組み合わせジョブには、次の制限が適用されます。
  - 選択したソーステーブルごとにプライマリキーが必要です。ソーステーブルにプライマリキーが存在しない場合、変更データキャプチャはテーブルを無視し、選択されたソーステーブルの残りの処理を続行します。どのソーステーブルにもプライマリキーがない場合、ジョブは失敗します。
  - クエリベースの CDC は、削除操作をキャプチャしません。
  - 挿入と更新の操作はすべて更新/挿入として扱われ、監視インタフェースに表示され、更新としてログに記録されます。
  - 特定のサイクルの開始時に夏時間またはタイムゾーンの変更が検出された場合、またはジョブが失敗状態または停止状態から再開されたときに、一括取り込みデータベースは再開してそのサイクルで発生した変更を処理します。夏時間またはタイムゾーンの変更を適用するには、Oracle データベースを再起動する必要があります。
  - Oracle ソースを持つデータベース取り込み初期ロードと増分ロードの組み合わせジョブでは、アーカイブ REDO ログのコピーから変更を読み取ることができます。Oracle データベース取り込み接続プロパティの **[読み取りモード]** プロパティを ARCHIVECOPY に設定し、ソースのカスタムプロパティ pwx.cdcreader.oracle.reader.additional を dir および file パラメータを指定して設定する必要があります。dir パラメータでは、CDC ログリーダーがアーカイブログコピーをスキャンするベースディレクトリの名前を指すようにし、file パラメータでは、ログコピーのフィルタリングに使用するマスクを指定します。
- Oracle ソースを使用したデータベース取り込みジョブの実行後、レプリケーション用に追加のソースカラムを選択してタスクを再デプロイした場合、ジョブは追加のカラムを使用してターゲットテーブルをすぐに

再作成したり、それらのデータをレプリケートしたりしません。ただし、増分ロードジョブ、または初期ロードと増分ロードの組み合わせジョブでは、スキーマドリフトの［**カラムの追加**］オプションを［**レプリケート**］に設定した場合、次の新しい DML 変更レコードを処理するときに、新しく選択したカラムがターゲットに追加され、データがレプリケートされます。初期ロードジョブでは、次回のジョブ実行時に、新しく選択したカラムがターゲットに追加され、データがレプリケートされます。

- 一括取り込みデータベースは、Oracle Exadata マシンから変更データをキャプチャできますが、Oracle Exadata Hybrid Columnar Compression（EHCC）はサポートしていません。

一括取り込みデータベース環境に関する情報の収集

データベース取り込みタスクの作成を開始する前に、以下の情報を収集してください。

全般的な情報

どの Oracle **バージョンを使用しているか。**

答え: _____

Oracle **をオンプレミスで実行するか、それともクラウドベースの** Amazon RDS for Oracle **環境で実行するか。**

答え: _____

ターゲットタイプは何か。

答え: _____

どのタイプのロード操作を実行する予定か。初期ロード（ポイントインタイムバルクロード）、増分ロード（変更のみ）、または初期ロードと増分ロードの組み合わせ（初期ロードとそれに続く増分ロード）のどれか。

答え: _____

Secure Agent **を実行するシステムのコア数、メモリ量、およびディスク容量はいくつか。**

答え: _____

Oracle 環境

Oracle **ソースデータベースサーバーのホスト名とポート番号は何か。**

答え: _____

データベースの Oracle **システム識別子（**SID**）は何か。**

答え: _____

データベースへの接続に使用する Oracle **データベースのユーザー名とパスワードは何ですか**?

答え: _____

Secure Agent **を実行するシステムに** Oracle Database Client **または** Instant Client **がインストールされていますか**?

答え: _____

データベースは Oracle Real Application Cluster（RAC）**で実行されますか**? **非アクティブなノードを含め、**RAC **メンバーの最大数はいくつですか**?

答え: _____

Oracle Data Guard **の論理または物理スタンバイデータベースから変更データをキャプチャする必要があるか。**

答え：_____

Oracle **マルチテナント環境のプラガブルデータベース（**PDB**）のテーブルから変更データをキャプチャする必要がありますか**?

答え: _____

Oracle Transparent Data Encryption（TDE）を使用するテーブルスペースから変更データをキャプチャする必要がありますか? はいの場合、TDE ウォレットディレクトリとパスワードは何ですか?

答え: _____

ソーステーブルの Unit of Work（UOW）の一般的なサイズはいくつか。

答え：_____

Oracle REDO ログ

REDO ログは Oracle Automatic Storage Management（ASM）環境にありますか? ASM インスタンスに接続して REDO ログを読み取る場合に、SYSDBA または SYSASM 権限を持つ ASM のログインユーザー ID の作成を許可されているか。

答え: _____

Oracle ソースデータベースの ARCHIVELOG モードと最少グローバル補足ログが有効になっているか。されていない場合は、有効にできるか。

答え：_____

変更データを読み取るアーカイブ REDO ログのプライマリおよびセカンダリアーカイブ先はどこか。

答え: _____

Oracle データベースのピーク期間中とピーク期間外に作成されるアーカイブ REDO ログの 1 時間あたりの平均量はいくつか。

答え: _____

使用している環境の REDO ログへの読み取りアクセスを持っているか。

答え: _____

REDO ログを直接読み取る権限を持っていない場合、アーカイブ REDO ログファイルを共有ディスクまたはファイルシステムにコピーし、その共有ディスクまたはファイルシステムから、コピーした REDO ログファイルにアクセスできるか。

答え: _____

一括取り込みデータベースで、アーカイブログだけでなくオンラインログから変更データを読み取れるようにするか。

答え: _____

CDC 処理中のエラーまたは異常を診断するために、必要に応じて、診断に使用するアーカイブ REDO ログを Informatica グローバルカスタマサポートに提供できるか。

答え: _____

データベース取り込みを設定するための詳細情報

データのレプリケート元となるソーステーブルのスキーマ名は何か。

答え: _____

スキーマ内のすべてのテーブルまたはそれらのテーブルのサブセットからデータをレプリケートするか。サブセットの場合は、それらのリストを作成してください。

答え: _____

ソーステーブルにプライマリキーがあるか。すべてのプライマリキーに対して補足ログを有効にできるか。

答え: _____

キーなしのソーステーブルはあるか。

答え: _____

ソーステーブルに、サポートされていないデータ型のカラムが含まれているか。ソースタイプでサポートされていないデータ型を判別するには、一括取り込みヘルプの「一括取り込みデータベースのソースに関する考慮事項」のソース固有のトピックを参照してください。

答え: _____

UTF-8 **のデフォルトのコードページは容認できるか。できない場合、どのコードページを使用するか。**

答え: _____

SSL **を使用して、**Secure Agent **とデータベースサーバー間で交換するデータを暗号化するか。暗号化** SSL **または** TLS **プロトコルはどれを使用するか。**

答え: _____

新しい Oracle **ユーザーを作成し、一括取り込みデータベースで必要となる特権をそのユーザーに割り当てることができるか。使用するユーザー名を決定してください。**

答え: _____

ソーステーブルに、一括取り込みデータベースがサポートしていない Oracle **データ型が含まれているか。**

答え: _____

カラムの追加、削除、変更、名前変更操作など、ソースでのスキーマドリフトの変更をキャプチャするか。

答え: _____

Oracle 特権

Oracle ソースを持つデータベース統合タスクをデプロイして実行するには、ソース接続で取り込みロードタイプに必要な権限を持つ一括取り込みデータベースユーザーを指定する必要があります。

ログベースの CDC を使用した増分ロード処理の権限

注: Oracle ログが ASM によって管理されている場合、ユーザーは SYSASM または SYSDBA 権限を持っている必要があります。

ログベースの CDC メソッドを使用した増分ロードまたは、初期ロードと増分ロードの組み合わせを実行するデータベース統合タスクの場合、一括取り込みデータベースのユーザー（*cmid_user*）に次の権限が付与されていることを確認してください。

```
GRANT CREATE SESSION TO <cmid_user>;

GRANT SELECT ON table TO <cmid_user>;     -- For each source table created by user
GRANT EXECUTE ON DBMS_FLASHBACK TO <cmid_user>;

-- The following grant is required for combined initial and incremental loads only. Do not
-- use ANY TABLE unless your security policy allows it.
GRANT FLASHBACK ON table|ANY TABLE TO <cmid_user>;

-- Include the following grant only if you want to Execute the CDC script for enabling
-- supplemental logging from the user interface. If you manually enable supplemental
-- logging, this grant is not needed.
GRANT ALTER table|ANY TABLE TO <cmid_user>;

GRANT SELECT ON DBA_CONSTRAINTS TO <cmid_user>;
GRANT SELECT ON DBA_CONS_COLUMNS TO <cmid_user>;
GRANT SELECT ON DBA_INDEXES TO <cmid_user>;
GRANT SELECT ON DBA_LOG_GROUPS TO <cmid_user>;
GRANT SELECT ON DBA_LOG_GROUP_COLUMNS TO <cmid_user>;
GRANT SELECT ON DBA_OBJECTS TO <cmid_user>;
GRANT SELECT ON DBA_OBJECT_TABLES TO <cmid_user>;
GRANT SELECT ON DBA_TABLES TO <cmid_user>;
GRANT SELECT ON DBA_TABLESPACES TO <cmid_user>;
GRANT SELECT ON DBA_USERS TO <cmid_user>;

GRANT SELECT ON "PUBLIC".V$ARCHIVED_LOG TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$CONTAINERS TO <cmid_user>;  -- For Oracle multitenant environments
```

```
GRANT SELECT ON "PUBLIC".V$DATABASE TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$DATABASE_INCARNATION TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$ENCRYPTION_WALLET TO <cmid_user>;     -- For Oracle TDE access
GRANT SELECT ON "PUBLIC".V$LOG TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$LOGFILE   TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$PARAMETER TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$PDBS TO <cmid_user>;     -- For Oracle multitenant environments
GRANT SELECT ON "PUBLIC".V$SPPARAMETER TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$STANDBY_LOG TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$THREAD TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$TRANSACTION TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$TRANSPORTABLE_PLATFORM TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$VERSION TO <cmid_user>;

GRANT SELECT ON SYS.ATTRCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CDEF$ TO <cmid_user>;
GRANT SELECT ON SYS.COL$ TO <cmid_user>;
GRANT SELECT ON SYS.COLTYPE$ TO <cmid_user>;
GRANT SELECT ON SYS.IDNSEQ$ TO <cmid_user>;
GRANT SELECT ON SYS.IND$ TO <cmid_user>;
GRANT SELECT ON SYS.INDPART$  TO <cmid_user>;
GRANT SELECT ON SYS.OBJ$ TO <cmid_user>;
GRANT SELECT ON SYS.PARTOBJ$ TO <cmid_user>;
GRANT SELECT ON SYS.RECYCLEBIN$ TO <cmid_user>;
GRANT SELECT ON SYS.TAB$ TO <cmid_user>;
GRANT SELECT ON SYS.TABCOMPART$  TO <cmid_user>;
GRANT SELECT ON SYS.TABPART$ TO <cmid_user>;
GRANT SELECT ON SYS.TABSUBPART$ TO <cmid_user>;

-- Also ensure that you have access to the following ALL_* views:
ALL_CONSTRAINTS
ALL_CONS_COLUMNS
ALL_ENCRYPTED_COLUMNS
ALL_INDEXES
ALL_IND_COLUMNS
ALL_OBJECTS
ALL_TABLES
ALL_TAB_COLS
ALL_TAB_PARTITIONS
ALL_USERS
```

クエリベースの CDC を使用した増分ロード処理の権限

クエリベースの CDC メソッドを使用した増分ロード、または初期ロードと増分ロードの組み合わせを実行するデータベース統合タスクの場合、ユーザーが少なくとも次の特権を持っていることを確認してください。

```
GRANT CREATE SESSION TO <cmid_user>;

GRANT SELECT ON DBA_INDEXES TO <cmid_user>;
GRANT SELECT ON DBA_OBJECT_TABLES TO <cmid_user>;
GRANT SELECT ON DBA_OBJECTS TO cmid_user;
GRANT SELECT ON DBA_TABLES TO <cmid_user>;
GRANT SELECT ON DBA_USERS TO <cmid_user>;
GRANT SELECT ON DBA_VIEWS TO <cmid_user>;  -- Only if you unload data from views

GRANT SELECT ANY TABLE TO <cmid_user>;
-or-
GRANT SELECT ON table TO <cmid_user>;         -- For each source table created by user

GRANT SELECT ON ALL_CONSTRAINTS TO <cmid_user>;
GRANT SELECT ON ALL_CONS_COLUMNS TO <cmid_user>;
GRANT SELECT ON ALL_ENCRYPTED_COLUMNS TO <cmid_user>;
GRANT SELECT ON ALL_IND_COLUMNS TO <cmid_user>;
GRANT SELECT ON ALL_INDEXES TO <cmid_user>;
GRANT SELECT ON ALL_OBJECTS TO <cmid_user>;
GRANT SELECT ON ALL_TAB_COLS TO <cmid_user>;
GRANT SELECT ON ALL_USERS TO <cmid_user>;

GRANT SELECT ON "PUBLIC"."V$DATABASE"  TO cmid_user;
GRANT SELECT ON "PUBLIC"."V$CONTAINERS" TO cmid_user;
```

```
GRANT SELECT ON SYS.ATTRCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CDEF$ TO <cmid_user>;
GRANT SELECT ON SYS.COL$ TO <cmid_user>;
GRANT SELECT ON SYS.COLTYPE$ TO <cmid_user>;
GRANT SELECT ON SYS.IND$ TO <cmid_user>;
GRANT SELECT ON SYS.IDNSEQ$ TO cmid_user;
GRANT SELECT ON SYS.OBJ$ TO <cmid_user>;
GRANT SELECT ON SYS.RECYCLEBIN$ TO <cmid_user>;
GRANT SELECT ON SYS.TAB$ TO <cmid_user>;
```

初期ロード処理の特権

初期ロードを実行するデータベース統合タスクの場合、ユーザーが少なくとも次の権限を持っていることを確認してください。

```
GRANT CREATE SESSION TO <cmid_user>;

GRANT SELECT ON DBA_INDEXES TO <cmid_user>;
GRANT SELECT ON DBA_OBJECT_TABLES TO <cmid_user>;
GRANT SELECT ON DBA_OBJECTS TO cmid_user;
GRANT SELECT ON DBA_TABLES TO <cmid_user>;
GRANT SELECT ON DBA_USERS TO <cmid_user>;
GRANT SELECT ON DBA_VIEWS TO <cmid_user>;  -- Only if you unload data from views

GRANT SELECT ANY TABLE TO <cmid_user>;
-or-
GRANT SELECT ON table TO <cmid_user>;         -- For each source table created by user

GRANT SELECT ON ALL_CONSTRAINTS TO <cmid_user>;
GRANT SELECT ON ALL_CONS_COLUMNS TO <cmid_user>;
GRANT SELECT ON ALL_ENCRYPTED_COLUMNS TO <cmid_user>;
GRANT SELECT ON ALL_IND_COLUMNS TO <cmid_user>;
GRANT SELECT ON ALL_INDEXES TO <cmid_user>;
GRANT SELECT ON ALL_OBJECTS TO <cmid_user>;
GRANT SELECT ON ALL_TAB_COLS TO <cmid_user>;
GRANT SELECT ON ALL_USERS TO <cmid_user>;

GRANT SELECT ON "PUBLIC"."V$DATABASE"  TO cmid_user;
GRANT SELECT ON "PUBLIC"."V$CONTAINERS" TO cmid_user;
GRANT SELECT ON SYS.ATTRCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CDEF$ TO <cmid_user>;
GRANT SELECT ON SYS.COL$ TO <cmid_user>;
GRANT SELECT ON SYS.COLTYPE$ TO <cmid_user>;
GRANT SELECT ON SYS.IND$ TO <cmid_user>;
GRANT SELECT ON SYS.IDNSEQ$ TO cmid_user;
GRANT SELECT ON SYS.OBJ$ TO <cmid_user>;
GRANT SELECT ON SYS.RECYCLEBIN$ TO <cmid_user>;
GRANT SELECT ON SYS.TAB$ TO <cmid_user>;n
```

Amazon RDS for Oracle ソースに対する Oracle 特権

Amazon RDS for Oracle ソースを使用している場合は、一括取り込みデータベースユーザーに特定の権限を付与する必要があります。

重要: GRANT 文とプロシージャを実行するには、マスターユーザー名で Amazon RDS にログインする必要があります。

少なくとも、CDC 処理に必要なオブジェクトおよびシステムテーブルに対する SELECT 特権を、一括取り込みデータベースユーザー（*cmid_user*）に付与します。特定の状況では、追加の権限の付与が必要になります。

次の GRANT 文を使用します。

```
GRANT SELECT ON "PUBLIC"."V$ARCHIVED_LOG"          TO "cmid_user";

GRANT SELECT ON "PUBLIC"."V$DATABASE"              TO "cmid_user";
GRANT SELECT ON "PUBLIC"."V$LOG"                   TO "cmid_user";
GRANT SELECT ON "PUBLIC"."V$LOGFILE"               TO "cmid_user";
```

```
GRANT SELECT ON "PUBLIC"."V$TRANSPORTABLE_PLATFORM" TO "cmid_user";
GRANT SELECT ON "PUBLIC"."V$THREAD"                 TO "cmid_user";
GRANT SELECT ON "PUBLIC"."V$DATABASE_INCARNATION"   TO "cmid_user";
GRANT SELECT ON "PUBLIC"."V$TRANSACTION"            TO "cmid_user";

GRANT SELECT ON "SYS"."DBA_CONS_COLUMNS"      TO "cmid_user";
GRANT SELECT ON "SYS"."DBA_CONSTRAINTS"       TO "cmid_user";
GRANT SELECT ON DBA_INDEXES                   TO "cmid_user";
GRANT SELECT ON "SYS"."DBA_LOG_GROUP_COLUMNS" TO "cmid_user";
GRANT SELECT ON "SYS"."DBA_TABLESPACES"       TO "cmid_user";

GRANT SELECT ON "SYS"."OBJ$"          TO "cmid_user";
GRANT SELECT ON "SYS"."TAB$"          TO "cmid_user";
GRANT SELECT ON "SYS"."IND$"          TO "cmid_user";
GRANT SELECT ON "SYS"."COL$"          TO "cmid_user";

GRANT SELECT ON "SYS"."PARTOBJ$"      TO "cmid_user";
GRANT SELECT ON "SYS"."TABPART$"      TO "cmid_user";
GRANT SELECT ON "SYS"."TABCOMPART$"   TO "cmid_user";
GRANT SELECT ON "SYS"."TABSUBPART$"   TO "cmid_user";
COMMIT;

/* For combined load jobs:*/
GRANT EXECUTE ON DBMS_FLASHBACK TO "cmid_user";

/*To provide read access to the Amazon RDS online and archived redo logs:*/
GRANT READ ON DIRECTORY ONLINELOG_DIR TO "cmid_user";
GRANT READ ON DIRECTORY ARCHIVELOG_DIR TO "cmid_user";
```

さらに、マスターユーザーとしてログインし、次の Amazon RDS プロシージャを実行して、さらにいくつかのオブジェクトに対する SELECT 特権を付与します。

```
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_TABLES',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_OBJECTS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_OBJECT_TABLES',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_VIEWS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
/begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_USERS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$CONTAINERS',
```

```
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$PARAMETER',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$SPPARAMETER',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$STANDBY_LOG',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$VERSION',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_CONS_COLUMNS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_CONSTRAINTS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_OBJECTS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_TABLES',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_TAB_PARTITIONS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
```

```
p_obj_name => 'ALL_USERS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rd 'ALL_TABLES',
p_grantee => 'sadmin_util.grant_sys_object(
p_obj_name => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_TAB_PARTITIONS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ATTRCOL$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'CCOL$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'COLTYPE$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'INDPART$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'IDNSEQ$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'CDEF$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'RECYCLEBIN$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
/* Only required for RDS21 which supports PDB*/
```

```
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$PDBS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
```

CDC の Oracle ログアクセス方法

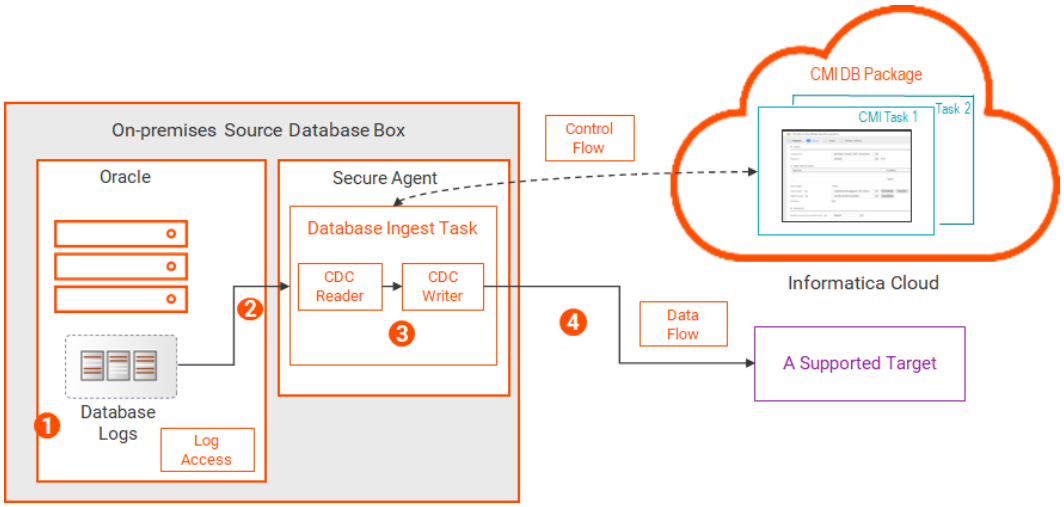
データベース取り込み増分ロードジョブ、および初期ロードジョブと増分ロードジョブの組み合わせは、環境と要件に応じて、CDC 処理のために別の方法で Oracle REDO ログにアクセスできます。

直接ログアクセス

データベース取り込みジョブは、オンプレミスのソースシステム上の物理 Oracle REDO ログに直接アクセスして、変更データを読み取ることができます。

注: ログをソリッドステートディスク（SSD）に保存すると、この方法で最高のパフォーマンスを実現できます。

次の図は、データフローを示しています。

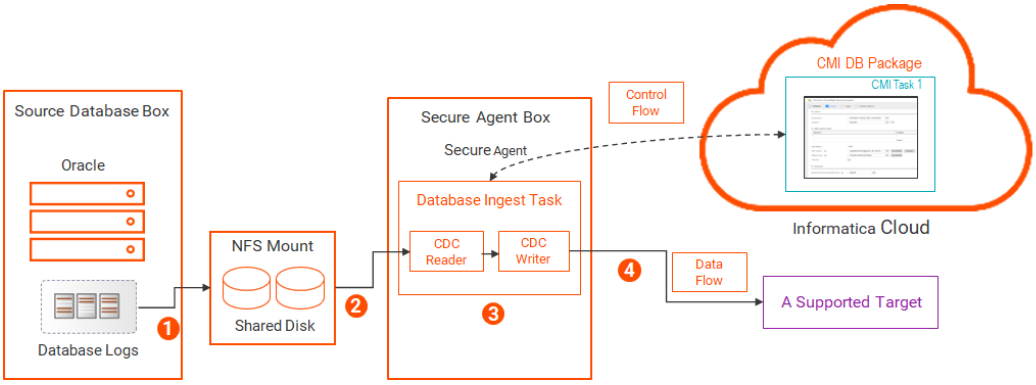


1. Oracle データベースが、変更レコードをディスク上のデータベースログファイルに書き込みます。
2. 一括取り込みデータベース CDC リーダーが、物理ログファイルを読み取り、CDC の対象となるソーステーブルのログファイルから変更レコードを抽出します。
3. 一括取り込みデータベース CDC ライターが変更レコードを読み取ります。
4. CDC ライターが、変更レコードをターゲットに適用します。

NFS マウントされたログ

データベース取り込みジョブは、ネットワークファイル共有（NFS）マウント、またはネットワーク接続ストレージ（NAS）やクラスタ化されたストレージなど別の方法を使用して、共有ディスクから Oracle データベースログにアクセスできます。

次の図は、データフローを示しています。



1. Oracle データベースが、変更レコードをデータベースログファイルに書き込みます。ログファイルが共有ディスクに書き込まれます。
   共有ディスクは、ファイルをデータベースと Secure Agent ホストの両方に対してローカルとして見える任意のシステムに配置できます。この共有は、上記のように NFS を使用するか、ネットワーク接続ストレージ（NAS）またはクラスタ化されたストレージを使用して実現できます。
2. 一括取り込みデータベース CDC リーダーが、ネットワークを介して NFS サーバーからログファイルを読み取り、CDC の対象となるソーステーブルの変更レコードを抽出します。
3. 一括取り込みデータベース CDC ライターが変更レコードを読み取ります。
4. CDC ライターが、変更レコードをターゲットに適用します。

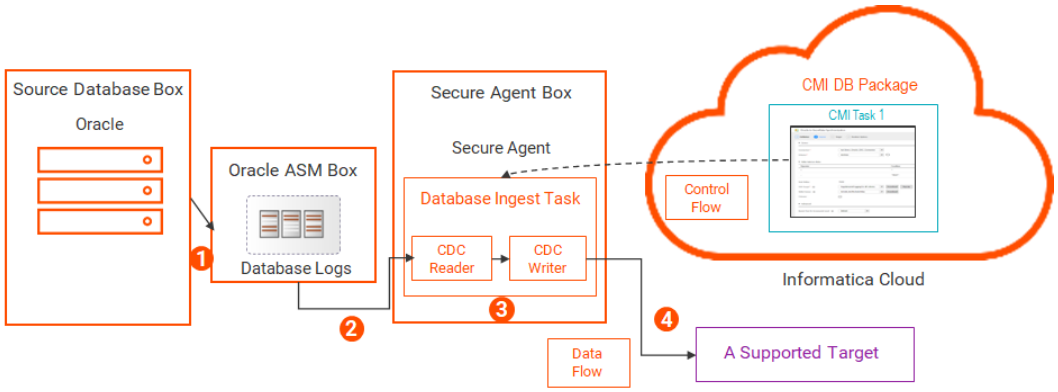
ASM 管理によるログ

データベース取り込みジョブは、Oracle Automatic Storage Management（ASM）システムに格納されている Oracle REDO ログにアクセスできます。ASM 管理による REDO ログから変更データを読み取るには、ASM ユーザーが、ASM インスタンスに対する SYSASM または SYSDBA 権限を持っている必要があります。

Oracle Database Ingestion 接続を設定するときは、名前に「ASM」を含むプロパティを入力します。

また、Informatica では、Oracle ASM の REDO ログファイルからデータを読み取る場合、ローカルの sqlnet.ora ファイルの sqlnet.recv_timeout パラメータを 5 分未満に設定することをお勧めします。このパラメータは、クエリがタイムアウトになるまでに Oracle クライアントが ASM からの応答を待機する時間を指定します。ネットワークの中断やその他の要因により、Oracle 接続が応答しなくなることがあります。この値を設定すると、リーダーがそのような状況に適時応答してリカバリできるようになります。

次の図は、データフローを示しています。



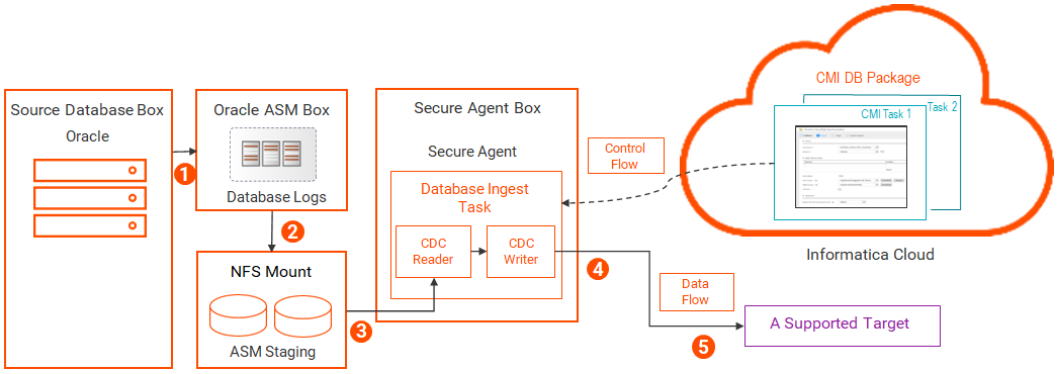
1. Oracle データベースが、変更レコードを ASM 管理によるデータベースログファイルに書き込みます。
2. 一括取り込みデータベース CDC リーダーが、ASM 管理によるログファイルを読み取り、CDC の対象となるソーステーブルの変更レコードを抽出します。

3. 一括取り込みデータベース CDC ライターが変更レコードを読み取ります。
4. CDC ライターが、変更レコードをターゲットに適用します。

ステージングディレクトリを使用した ASM 管理によるログ

データベース取り込みジョブは、ASM 環境のステージングディレクトリから ASM 管理による REDO ログにアクセスできます。ASM のみを使用する場合と比較して、この方法ではログファイルへのアクセスが高速になり、ASM システムの I/O が低減されます。ASM 管理によるログから変更データを読み取るには、ASM ユーザーが、ASM インスタンスに対する SYSASM または SYSDBA 権限を持っている必要があります。

次の図は、データフローを示しています。

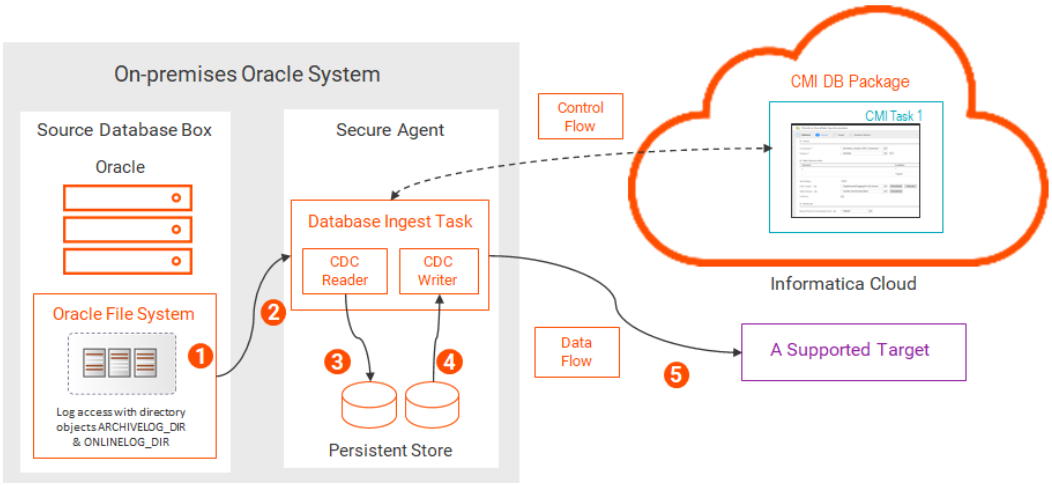


1. Oracle データベースが、変更レコードを ASM 管理によるログファイルに書き込みます。
2. ASM が、ログをステージングディレクトリにコピーします。
   ステージングディレクトリは、NFS マウントなどの共有ディスク上にある必要があります。そうすることにより、ASM はそこにデータを書き込み、データベース取り込みジョブはそこからデータを読み取ることができます。
3. 一括取り込みデータベース CDC リーダーが、ステージングディレクトリにあるログファイルを読み取り、CDC の対象となるソーステーブルの変更レコードを抽出します。
4. 一括取り込みデータベース CDC ライターが変更レコードを読み取ります。
5. CDC ライターが、変更レコードをターゲットに適用します。

ディレクトリオブジェクトを使用した Oracle サーバーファイルシステムのログへの BFILE アクセス

オンプレミスの Oracle ソースシステムでは、BFILE ロケータを備えた Oracle ディレクトリオブジェクトを使用して、ローカル Oracle サーバーファイルシステムからオンライン REDO ログとアーカイブ REDO ログを読み取るように、一括取り込みデータベースを設定できます。Oracle REDO ログファイルの場所を指す ARCHIVELOG_DIR および ONLINELOG_DIR という名前の Oracle ディレクトリオブジェクトを作成する必要があります。BFILE アクセスの設定については、「Oracle ファイルシステムの Oracle REDO ログへの BFILE アクセスの設定」 (ページ 54)を参照してください。

次の図は、データフローを示しています。



1. Oracle データベースが、ローカル Oracle サーバーファイルシステムの REDO ログファイルに変更レコードを書き込みます。データベース取り込みタスクがログファイルを読み取る必要があるときに、Oracle に接続し、ARCHIVELOG_DIR または ONLINELOG_DIR ディレクトリオブジェクトを参照してログにアクセスするための select 要求を発行します。
2. 一括取り込みデータベース CDC リーダーがログファイルを読み取り、CDC の対象となるソーステーブルの変更レコードを抽出します。
3. 一括取り込みデータベース CDC ライターが変更レコードを読み取ります。
4. CDC ライターが、変更レコードをターゲットに適用します。

Oracle ファイルシステムの Oracle REDO ログへの BFILE アクセスの設定

REDO ログをローカルの Oracle サーバーファイルシステムに保存し、BFILE で Oracle ディレクトリオブジェクトを使用してログにアクセスする場合は、次の設定タスクを実行します。

BFILE アクセスに固有ではない、次の通常の Oracle ソース準備タスクを完了します。

- 一括取り込みデータベースで Oracle Call Interface（OCI）を使用して Oracle ソースデータベースと通信するために Secure Agent が実行される Linux または Windows システムで、ORACLE_HOME 環境変数を定義します。
- 一括取り込みデータベースユーザーに、データベース取り込み増分ロード処理に必要な Oracle 権限があることを確認してください。詳細については、「Oracle 特権」 (ページ 45)を参照してください。
- Oracle データベースの ARCHIVELOG モードを有効にします。
- アーカイブログの格納先を定義します。

  注: BFILE アクセスの場合は、特定のアーカイブログの格納先ディレクトリを使用します。Oracle USE_DB_RECOVERY_FILE_DEST キーワードを使用して、アーカイブログを日付別に高速リカバリ領域（FRA）に保存しないでください。
- ソースデータベースで Oracle の最小限のグローバルサプリメンタルロギングを有効にします。
- Oracle ソーステーブルにプライマリキーがある場合は、すべてのプライマリキーカラムに対してサプリメンタルロギングが有効になっていることを確認してください。プライマリキーのないソーステーブルの場合、変更データがキャプチャされるすべてのカラムでサプリメンタルロギングが有効になっていることを確認してください。

  注: データベース取り込みタスクを作成するときに、選択したソーステーブルのすべてのカラムまたはプライマリキーカラムのみのサプリメンタルロギングを実装するスクリプトを生成するオプションがあります。

- Oracle MAX_STRING_SIZE 初期化パラメータが EXTENDED に設定されて*いない*ことを確認してください。EXTENDED に設定されている場合、一括取り込みデータベースは、大きな（拡張サイズ）VARCHAR2、NVARCHAR2、または RAW カラムで定義されたカラムを含むテーブルの挿入と更新をレプリケートできません。

さらに、BFILE アクセスの場合は、次の手順を実行します。

1. Oracle データベースに、Oracle サーバーファイルシステム内のオンライン REDO ログとアーカイブ REDO ログの場所をクエリします。次のサンプルクエリを使用できます。
   オンライン REDO ログの場所を取得するには、次の手順を実行します。

   ```
   select * from v$logfile;
   ```

   ログのアーカイブ先を取得するには、次の手順を実行します。

   ```
   select dest_id, dest_name, destination, status from V$ARCHIVE_DEST;
   ```

2. 手順 1 で取得したログファイルの場所を指す ONLINELOG_DIR および ARCHIVELOG_DIR ディレクトリオブジェクトを作成します。Oracle ディレクトリオブジェクトは、アクセスするログファイルが配置されている Oracle サーバーファイルシステム上の物理ディレクトリの論理エイリアス名を指定します。以下に例を示します。

   ```
   CREATE DIRECTORY ONLINELOG_DIR AS '/u01/oracle/data';
   CREATE DIRECTORY ARCHIVELOG_DIR AS '/u01/oracle/archivedata';
   ```

   注: Oracle Database Ingestion 接続でリーダーモードを［ARCHIVEONLY］に設定して、アーカイブログからのみ変更を読み取る場合は、ONLINELOG_DIR ディレクトリまたはディレクトリオブジェクトを作成する必要はありません。

   Oracle データベースは、指定したディレクトリが存在することを確認しません。Oracle ファイルシステムに存在する有効なディレクトリを指定していることを確認してください。

3. ディレクトリオブジェクトが REDO ログの正しいファイルシステムパスで作成されたことを確認するには、次のような select 文を発行します。

   ```
   select * from all_directories;
   OWNER    DIRECTORY_NAME      DIRECTORY_PATH
   -------- ------------------- --------------------------------
   SYS      ARCHIVELOG_DIR      /u01/oracle/data/JO112DTL
   SYS      ONLINELOG_DIR       /u01/oracle/data/JO112DTL
   ```

4. Oracle データベース取り込み接続プロパティで指定された一括取り込みデータベースユーザーに、ONLINELOG_DIR および ARCHIVELOG_DIR ディレクトリオブジェクトへの読み取りおよび書き込みアクセスを付与します。以下に例を示します。

   ```
   grant read on directory "ARCHIVELOG_DIR" to "cmid_user";
   grant read on directory "ONLINELOG_DIR" to "cmid_user";
   ```

5. Oracle Database Ingestion 接続プロパティで、**［BFILE アクセス］**チェックボックスを選択します。

ソースとしての Oracle Data Guard データベースまたは Far Sync インスタンス

一括取り込みデータベースは、Oracle Data Guard プライマリデータベース、論理および物理スタンバイデータベース、および Far Sync インスタンスから変更データをキャプチャできます。

Far Sync インスタンスは、プライマリデータベースから REDO を受け入れ、その REDO を Oracle Data Guard 構成の他のメンバーに送信する、リモート Oracle Data Guard の宛先です。

Oracle Data Guard プライマリデータベースまたは読み取りモードで開いているスタンバイデータベースからデータをターゲットに初期ロードできます。

設定

Oracle 変更キャプチャの設定は、Oracle Data Guard データベースタイプによって異なります。

- *プライマリ Oracle データベース*の場合は、V$STANDBY_LOG ビューへの SELECT 権限を一括取り込みデータベースユーザーに付与します。

  ```
  GRANT SELECT ON "PUBLIC".V$STANDBY_LOG TO <cmid_user>;
  ```

  プライマリデータベースが Amazon RDS for Oracle 環境にある場合は、次のように設定します。

  ```
  begin
  rdsadmin.rdsadmin_util.grant_sys_object(
  p_obj_name => 'V_$STANDBY_LOG',
  p_grantee => 'cmid_user',
  p_privilege => 'SELECT');
  end;
  ```

- *マウントモードの物理スタンバイデータベース*（読み取り専用アクセス権では開かない）の場合は、次の Oracle Database Ingestion 接続プロパティを設定します。

  - データベース接続文字列 - Oracle カタログを読み取れるようにプライマリデータベースを指していることを確認します。
  - スタンバイ接続文字列 - ログリーダーが Oracle 物理スタンバイデータベースに接続し、ログを監視するために使用する、TNS で定義された Oracle 接続文字列。
  - スタンバイユーザー名 - ログリーダーが Oracle 物理スタンバイデータベースに接続するために使用するユーザー ID。このユーザー ID には SYSDBA 権限が必要です。
  - スタンバイパスワード - ログリーダーが Oracle 物理スタンバイデータベースに接続するために使用するパスワード。

  注: マウントモードのデータベースでは、ユーザー認証にパスワードファイルを使用できます。最初に、SYSDBA 権限をユーザーに付与する必要があります。ユーザーに永続的な SYSDBA 権限を付与することを回避するには、プライマリパスワードファイルを物理スタンバイまたは Far Sync インスタンスにコピーしてから、ユーザーの SYSDBA 権限を取り消します。パスワードファイルを更新するたびに、このプロセスを繰り返します。

  必要に応じて、次の追加の接続プロパティを設定します。

  - RAC メンバ - データベースが RAC 環境にある場合に、Data Guard プライマリデータベース上にあるアクティブなスレッドの最大数。
  - リーダースタンバイログマスク - Oracle スタンバイデータベースで REDO ログの多重化を使用しているときに、ログリーダーがデータベースの REDO ログを選択するために使用するマスク。

  詳細については、「*コネクタと接続*」の「Oracle Database Ingestion 接続プロパティ」を参照してください。

- *論理スタンバイデータベース*の場合、特別な設定タスクは必要ありません。Data Guard 環境にない Oracle データベースと同じ方法で設定します。

スタンバイからプライマリロールへの遷移

Oracle Data Guard 環境では、物理スタンバイデータベースがプライマリロールに遷移する可能性があります。通常、ロールの遷移はフェイルオーバーまたはスイッチオーバーが原因で発生します。遷移中は、物理スタンバイデータベースへのすべてのアクティブな接続が終了します。

物理スタンバイデータベースがプライマリロールに遷移した後に CDC 処理を再開できるようにするには、一括取り込みデータベースが遷移ポイントを過ぎて処理できるように、元のスタンバイシステムでいくつかの接続設定プロパティを調整することが必要になる場合があります。遷移が完了したら、新しいプライマリデータベース環境でパフォーマンスを最適化するようプロパティを再度調整できます。

次の表で、この接続プロパティについて遷移段階別に説明します。

接続プロパティ	遷移前	遷移中	遷移後
RAC メンバ	プライマリデータベース上のアクティブなスレッドの数を指定します。	スタンバイデータベースとプライマリデータベースの*両方*のデータベース上で一意のスレッド ID を持つアクティブなスレッドの総数を指定します。 例えば、プライマリデータベースがスレッド ID 1 と 2 を使用する 2 ノードの RAC データベースであり、スタンバイデータベースがスレッド ID 2、3、4 を使用する 3 ノードの RAC データベースの場合、プロパティ値を 4 に指定します。	再開ポイントが遷移ポイントを超えるまで進行したら、必要に応じてこのプロパティ値を編集し、新しいプライマリデータベースからの変更データキャプチャのパフォーマンスを最適化します。 CDC のスレッド追跡によるオーバーヘッドを最小限に抑えるため、環境に適した最小値を使用することをお勧めします。
リーダースタンバイログマスク スタンバイ接続文字列 スタンバイユーザー名 スタンバイパスワード	すべてのスタンバイプロパティを削除します。読み取り専用アクセス用に開いている物理スタンバイデータベースには適用されません。	プロパティは削除されたままです。	これらのプロパティは指定しないでください。これらはプライマリデータベースでは使用されません。
データベース接続文字列	スタンバイデータベースが開かれていない場合、プライマリデータベース用の接続文字列を定義します。 スタンバイデータベースが開かれている場合、スタンバイデータベース用の接続文字列を定義します。	ロールの遷移後にプライマリロールになるデータベース用の接続文字列を指定します。	この接続プロパティが、新しいプライマリデータベースの接続文字列を定義していることを確認してください。

Oracle アーカイブログの保持に関する考慮事項

データベース取り込み増分ロードジョブ、および初期ロードジョブと増分ロードジョブの組み合わせは、Oracle オンライン REDO ログおよびアーカイブ REDO ログのトランザクションデータにアクセスできる必要があります。ログを利用できない場合、データベース取り込みジョブはエラーで終了します。

通常、Oracle DBA は、組織の特定のビジネスニーズと Oracle 環境に基づいて、アーカイブログの保持期間を設定します。ログを再起動処理に使用できるように、変更キャプチャが停止または潜在的な状態になると予想される最長期間プラス約 1 時間にわたって、ソースアーカイブログが保持されていることを確認してください。

環境内の現在のログ保持ポリシーが、データベース取り込み変更キャプチャ処理に対応するのに十分かどうかを判断するには、次の要素を考慮してください。

- Oracle トランザクションは、通常、ソース上でどのくらいの期間開かれていますか?
- 週末と休日を考慮して、変更キャプチャがダウンまたは潜在的な状態になることが許容される最長期間はどれくらいですか?

- ソースからターゲットへのレプリケーション待ち時間はどのくらいですか?
- データベース取り込みジョブをスケジュールに基づいて実行していますか? そうである場合、どのようなスケジュールですか?
- タスクウィザードの［**ソース**］ページで、pwx.cdcreader.oracle.option.additional ageOutPeriod=*minutes* カスタムプロパティが設定されていますか?

  注: このプロパティは、CDC 対象の変更レコードのない未処理の UOW を次の再開ポイントの計算から削除する際の条件となる経過時間を指定します。このプロパティを使用すると、トランザクションが未処理で、UOW が開始された REDO ログが使用できないときにキャプチャ処理をシャットダウンして再開した場合に発生する可能性のある CDC 障害を防ぐことができます。
- REDO 生成率はどのくらいですか?
- アーカイブログのコピーをセカンダリシステムに配布していますか?

ログのキャプチャ処理を再開する必要があるときにアーカイブログを使用できない場合は、DBA に、アーカイブログを復元して必要に応じて保存期間を変更するよう依頼してください。それ以外の場合は、別の初期ロードを実行してターゲットを再マテリアライズしてから、増分変更データ処理を再度開始してください。ただし、この場合、一部の変更が失われる可能性があります。

PostgreSQL ソース

データベース取り込みタスクで PostgreSQL ソースを使用するには、最初にソースデータベースを準備し、使用に関する考慮事項を確認してください。

ソースの準備:

データベース取り込みタスクで PostgreSQL ソースを使用するには、最初にソースデータベースを準備し、使用に関する考慮事項を確認してください。

*Secure Agent システム上*で、オペレーティングシステムに適した ODBC ドライバをインストールします。

- Windows では、64 ビット PostgreSQL ODBC ドライバの最新バージョンをインストールします。
  1. PostgreSQL ODBC ドライバをダウンロードしてインストールします。

     注: ソースデータベースにマルチバイト文字名のオブジェクト（テーブル名、カラム名、パブリケーション名など）が含まれている場合は、PostgreSQL Unicode ODBC ドライバまたは PostgreSQL 用 DataDirect ODBC ドライバのいずれかを使用する必要があります。この要件は、Amazon Aurora PostgreSQL、Azure Database for PostgreSQL - Flexible Server、Cloud SQL for PostgreSQL、および RDS for PostgreSQL などのすべての PostgreSQL ソースタイプに適用されます。Unicode 互換の ODBC ドライバを使用しない場合、マルチバイト文字の名前が検出されると、増分ロードジョブは失敗します。
  2. PGSQL_ODBC_DRIVER 環境変数を、ODBC Data Source Administrator（64 ビット）によって表示されるドライバ名に設定します。

     注: タスクウィザードの［ソース］ページで pwx.custom.pgsql_odbc_driver カスタムプロパティを設定することにより、データベース取り込みタスクのこのドライバをオーバーライドできます。

- Linux または UNIX では、PostgreSQL 用 DataDirect ODBC ドライバが Linux インストールの一部として提供されます。unixODBC ドライバマネージャか iODBC ドライバマネージャ、または PostgreSQL ODBC ドライバをインストールすることもできます。

  1. Linux インストールで提供される DataDirect ODBC for PostgreSQL ドライバを使用しない場合は、unixODBC ドライバマネージャまたは iODBC ドライバマネージャをインストールするか、PostgreSQL ODBC ドライバをインストールします。

     注: ソースデータベースにマルチバイト文字名のオブジェクト（テーブル名、カラム名、パブリケーション名など）が含まれている場合は、PostgreSQL Unicode ODBC ドライバまたは PostgreSQL 用 DataDirect ODBC ドライバのいずれかを使用する必要があります。この要件は、Amazon Aurora PostgreSQL、Azure Database for PostgreSQL - Flexible Server、Cloud SQL for PostgreSQL、および RDS for PostgreSQL などのすべての PostgreSQL ソースタイプに適用されます。Unicode 互換の ODBC ドライバを使用しない場合、マルチバイト文字の名前が検出されると、増分ロードジョブは失敗します。

  2. PostgreSQL エントリを odbcinst.ini に追加します。

     ```
     [PGSQL]
     Description = ODBC for PostgreSQL
     Driver =
     Setup =
     Driver64 = /usr/pgsql-9.6/lib/psqlodbca.so
     Setup64 = /usr/lib64/libodbcpsqlS.so
     FileUsage = 1
     ```

  3. オプション。以下の環境変数を設定します。

     - ODBCSYSINI 変数を odbcinst.ini が配置されているディレクトリに設定します。odbcinst.ini がデフォルトの/etc ディレクトリに配置されている場合は、ODBCSYSINI 変数を設定する必要はありません。
     - PostgreSQL ODBC ドライバがインストールされているディレクトリを LD_LIBRARY_PATH 変数に追加します。ドライバが/usr/lib64 のデフォルトディレクトリにインストールされている場合は、LD_LIBRARY_PATH 変数にパスを追加する必要はありません。
     - PGSQL_ODBC_DRIVER パラメータを、odbcinst.ini で指定したドライバ名に設定します。

     例:

     ```
     export ODBCSYSINI=/root/infaagent
     export LD_LIBRARY_PATH=/usr/pgsql-9.6/lib
     export PGSQL_ODBC_DRIVER=PGSQL
     ```

*PostgreSQL データベースシステム*で、次の設定手順を実行します。

1. 増分ロードジョブ、初期および増分ロードジョブの場合、PostgreSQL postgresql.conf 構成ファイルで wal_level=logical パラメータが指定されていることを確認します。
   このパラメータは、PostgreSQL がログ先行書き込み（WAL）に書き込む情報の量を決定します。論理の設定によって、論理デコードのサポートに必要な情報が追加されます。

   Amazon Aurora PostgreSQL または Amazon RDS for PostgreSQL ソースで wal_level を logical に設定するには、クラスタパラメータグループで rds.logical_replication パラメータを 1 に設定します。Azure Database for PostgreSQL - フレキシブル サーバ-の場合、Azure ポータルの［サーバーパラメータ］ページで wal_level パラメータを logical に設定します。

   Cloud SQL for PostgreSQL ソースの場合は、次のアクションを実行します。

   a. パブリック IP を使用してデータベースに接続します。

      注: Google Cloud コンソールの**［承認済みネットワーク］**に必要な IP を必ず追加してください。

   b. Cloud SQL for PostgreSQL データベースインスタンスレプリカを作成します。

c. Cloud Shell で、ローカル管理者ユーザーとして次のコマンドを実行します。

```
alter database postgres set default_transaction_read_only = off;
gcloud sql connect database_replica --user=postgres --quiet;
ALTER USER postgres WITH REPLICATION;
CREATE USER replication_user WITH REPLICATION IN ROLE cloudsqlsuperuser LOGIN PASSWORD 'password';
ALTER USER postgres WITH REPLICATION;
```

d. Google Cloud コンソールで、次のデータベースフラグを追加します:

- **cloudsql.logical_decoding**。値を on に設定します。
- **max_replication_slots**.値を 64 に設定します。
- **cloudsql.enable_pglogical**.値を on に設定します。
- **max_wal_senders**。値を 64 に設定します。

e. データベースインスタンスを再起動します。

2. PostgreSQL 用 DataDirect ODBC ドライバを使用する場合は、データベースで SCRAM-SHA-256 認証方法を使用していないことを確認してください。MD5 などの別の認証方法を使用します。

   注: PostgreSQL ODBC ドライバは、SCRAM-SHA-256 認証方法をサポートしています。PostgreSQL 13 では、この認証方法がデフォルトの方法になりました。

3. PostgreSQL ソースを含むデータベース取り込みタスクをデプロイして実行するには、ソース接続で必要な特権を持つデータベースユーザーを指定する必要があります。次の方法でユーザーを作成し、そのユーザーに特権を付与します。

   - 初期ロードジョブの場合は、次の SQL 文を使用します。

     ```
     CREATE USER dbmi_user WITH PASSWORD 'password';
     GRANT SELECT ON ALL TABLES IN SCHEMA schema TO dbmi_user;
     ```

   - オンプレミスの PostgreSQL ソースを使用する増分ロードジョブ、初期および増分ロードジョブの場合、次の SQL 文を使用します。

     ```
     CREATE USER dbmi_user WITH PASSWORD 'password' REPLICATION;
     ```

     Amazon Aurora PostgreSQL ソースと RDS for PostgreSQL ソースの場合、次の文を使用します。

     ```
     CREATE USER dbmi_user WITH PASSWORD 'password';
     GRANT rds_replication to dbmi_user;
     ```

     また、pgoutput プラグインを使用する場合は、次の SQL 文を使用して、pgoutput パブリケーションに追加するデータベース内のテーブルの所有権を、作成した *dbmi_user* に付与します。

     ```
     GRANT CREATE ON DATABASE database TO dbmi_user;
     ```

4. 増分ロードジョブ、または初期および増分ロードジョブの論理デコード出力に wal2json プラグインを使用する場合は、プラグインをインストールします。

5. 増分ロードジョブ、または初期および増分ロードジョブに pgoutput プラグインを使用する場合は、次の SQL 文を使用して、データベース取り込みジョブのパブリケーションを作成します。

   ```
   CREATE PUBLICATION publication_name [FOR TABLE [ONLY] table_name [*] [,...] | FOR ALL TABLES ];
   ```

   ターゲットにレプリケートするすべてのテーブルがパブリケーションに含まれていることを確認してください。

6. PostgreSQL 9.6 ソースを使用する増分ロードジョブ、初期および増分ロードジョブの場合、postgresql.conf 構成ファイルの max_replication_slots パラメータの値が、使用する予定の同時データベース取り込みジョブの数以上であることを確認します。

   重要: すべてのレプリケーションスロットは、すべての同時ジョブで一意である必要があります。

7. 増分ロードジョブ、初期および増分ロードジョブの場合、PostgreSQL ソースが UTF-8 エンコードを使用していることを確認します。

使用に関する考慮事項:

- 一括取り込みデータベースは、あらゆるロードタイプのデータベース取り込みジョブで、オンプレミス PostgreSQL、Amazon Aurora PostgreSQL、Azure Database for PostgreSQL - Flexible Server、Cloud SQL for PostgreSQL、および RDS PostgreSQL の PostgreSQL ソースのタイプをサポートします。
- PostgreSQL ソースを持つデータベース取り込みジョブは、PostgreSQL と SQL Server を除く任意のターゲットタイプを持つことができます。
- 一括取り込みデータベースは、Oracle または Snowflake ターゲットのみを持つ初期ロードと増分ロードの組み合わせジョブで PostgreSQL ソースをサポートします。
- 一括取り込みデータベースは、Google BigQuery または Snowflake ターゲットのみを持つあらゆるロードタイプの Cloud SQL for PostgreSQL ソースをサポートします。
- 一括取り込みデータベースでは、ソーステーブルの各行が一意であることを想定しているため、各ソーステーブルにプライマリキーを持たせることをお勧めします。一括取り込みデータベースは、プライマリキーの代わりに一意のインデックスを許可しません。プライマリキーが指定されていない場合、一括取り込みデータベースはすべてのカラムをプライマリキーの一部であるかのように扱います。
- 一括取り込みデータベースは、データベース取り込み増分ロードジョブ、初期および増分ロードジョブで PostgreSQL ソースのスキーマドリフトオプションをサポートしますが、次の制限があります。

  - PostgreSQL は、変更データキャプチャが有効になっているテーブルのプライマリキーの変更をサポートしていません。

  - データベース取り込みジョブは、テーブルパーティション ID が変更されたソーステーブルからの DML 変更をキャプチャできません。
- 一括取り込みデータベースでは、PostgreSQL ソースを含む増分ロードジョブで生成されたカラムはサポートされていません。生成されたカラムがソーステーブルに含まれている場合、変更データキャプチャはそれらのカラムを無視し、残りのカラムの処理を続行します。
- タスクウィザードの **[ソース]** ページの **[詳細]** で **[LOB を含める]** を選択した場合、データベース取り込みジョブで、PostgreSQL BYTEA、JSON、JSONB、TEXT、および XML カラムからデータをレプリケートできます。
  サポートされるターゲットタイプは、ロードタイプによって異なります。

  - 初期ロードジョブと増分ロードジョブの場合: Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Microsoft Azure Data Lake Storage Gen2、Microsoft Azure Synapse Analytics、Oracle Cloud Object Storage、および Snowflake。

  - 増分ロードジョブの場合: Azure Event Hubs。

  - 初期ロードと増分ロードの組み合わせジョブの場合: Snowflake。

  タスクウィザードの **[ソース]** ページで、pwx.custom.pgsql_enable_lobs カスタムプロパティを true に設定した場合、データベース取り込み増分ロードジョブ、初期および増分ロードジョブは、長さ制限のない TEXT、XML、BIT VARYING、および CHARACTER VARYING カラムからデータをレプリケートできます。増分ロードジョブ、初期および増分ロードジョブも常に、BYTEA、JSON、および JSONB カラムからデータをレプリケートします。

  LOB カラムデータは、LOB タイプとターゲットタイプによって異なるバイト制限よりもサイズが大きい場合、ターゲットに書き込まれる前に切り詰められます。詳細については、「ソースの設定」 (ページ 102)を参照してください。
- PostgreSQL 9.6 の場合、pgoutput プラグインは使用できません。
- 初期ロードジョブの場合、一括取り込みデータベースでは次の PostgreSQL データ型はサポートされていません。

  - ABSTIME

  - 配列型

- NAME
- オブジェクト識別子型
- PG_LSN
- RELTIME
- テキスト検索型:
  - TSQUERY
  - TSVECTOR
- ユーザー定義型

増分ロードジョブ、初期および増分ロードジョブの場合、一括取り込みデータベースでは、初期ロードジョブでサポートされていないものに加えて、次の PostgreSQL データ型はサポートされていません。

- 空間タイプ
  - Box
  - Circle
  - Line
  - LSeg
  - Path
  - Point
  - Polygon
- 無制限のさまざまなタイプ

データベース統合ジョブは、これらのデータ型を持つカラムにはデプロイしたり null をプロパゲートしたりすることはできません。

サポートされている PostgreSQL データ型からターゲットタイプへのデフォルトのマッピングについては、「デフォルトデータ型のマッピング」 (ページ 169)を参照してください。

- ターゲットテーブルに存在しないレコードに対して PostgreSQL ソーステーブルのプライマリキー値を更新すると、そのレコードはターゲットにレプリケートされません。ただし、監視インタフェースは更新カウントを増分してプライマリキーの更新を含めます。プライマリキーの更新が実行される前にターゲットテーブルにレコードが存在する場合にのみ、データがターゲットにレプリケートされます。

SAP HANA および SAP HANA Cloud ソース

データベース取り込みタスクで SAP HANA および SAP HANA Cloud ソースを使用するには、最初にソースデータベースを準備し、使用に関する考慮事項を確認してください。

ソースの準備:

- SAP HANA Database Ingestion コネクタは、JDBC を使用して SAP HANA および SAP HANA Cloud データベースに接続し、データとメタデータを読み取って、接続プロパティをテストします。SAP HANA JDBC ドライバファイル ngdbc.jar をダウンロードし、Secure Agent が実行されているマシンの Secure Agent インストールディレクトリの特定のサブディレクトリにコピーする必要があります。

  1. SAP HANA JDBC ドライバの jar ファイル ngdbc.jar を、Secure Agent が実行されている Linux または Windows マシンにダウンロードします。
     ダウンロードするファイルが最新バージョンであることを確認します。ファイルのダウンロードで問題が発生した場合は、SAP カスタマサポートにお問い合わせください。

2. 次のディレクトリに ngdbc.jar ファイルをコピーします。

   <*Secure Agent installation directory*>/ext/connectors/thirdparty/informatica.hanami

3. Secure Agent を再起動します。

- 一括取り込みデータベースユーザーを作成します。管理者権限を持つユーザーとしてソースデータベースに接続し、次の文を実行します。

  CREATE USER *dbmi_user* password "<*password*>" NO FORCE_FIRST_PASSWORD_CHANGE;

  この文は、デフォルトの権限を使用してデータベースにユーザーを作成します。これにより、基本的なデータディクショナリビューを読み取ることができ、必要な CDC オブジェクトをユーザー自身のスキーマに作成することができます。

- SAP HANA または SAP HANA Cloud ソースを含むデータベース取り込みタスクをデプロイして実行するには、ソース接続で、次のシステムビューからメタデータやその他の情報を読み取る権限を持つ一括取り込みデータベースユーザー（*dbmi_user*）を指定する必要があります。

  - SYS.M_DATABASE - データベースのバージョンを取得するために使用されます。

  - SYS.M_CS_PARTITIONS- テーブルがパーティション化されているかどうかを識別するために使用されます。（SAP HANA Cloud には適用されません）

  - SYS.SCHEMAS - データベースのスキーマのリストを取得するために使用されます。

  - SYS.TABLES - スキーマのテーブル名のリストを取得するために使用されます。

  - SYS.TABLE_COLUMNS - テーブルのカラムメタデータを取得するために使用されます。

  - SYS.INDEXES - テーブルのインデックス情報を取得するために使用されます。

  - SYS.INDEX_COLUMNS - テーブルのインデックス情報を取得するために使用されます。

- 増分ロードジョブの場合、次の権限を付与します。

  - *dbmi_user* の PKLOG およびシャドー_CDC テーブルに行を書き込むトリガの場合は、*dbmi_user* のスキーマに対する INSERT アクセス権を、ソーステーブルのスキーマを所有するユーザー（ *schema_user*）に付与します。

    GRANT INSERT ON SCHEMA *dbmi_user* TO *schema_user*;

  - トリガを使用して schema_user のスキーマ内のソーステーブルから変更データをキャプチャするには、次のいずれかの文を実行します。

    GRANT TRIGGER ON SCHEMA *schema_user* TO *dbmi_user*;

    この文は、スキーマ内のすべてのテーブルに対するトリガアクセス権を付与します。

    - or -

    GRANT TRIGGER ON *database.table_name* TO *dbmi_user*;

    この文は、特定のソーステーブルに対するトリガアクセス権を付与します。選択した少数のテーブルからデータをキャプチャする場合は、この文を使用します。CDC 対象のソーステーブルごとに、付与を繰り返します。

- 初期ロードジョブの場合、ソーステーブルからデータを読み取るために、次の GRANT 文のいずれかを実行します。

  GRANT SELECT ON SCHEMA *schema_user* TO *dbmi_user*;

  この文は、スキーマ内のすべてのテーブルに対する SELECT アクセス権を付与します。

  - or -

  GRANT SELECT ON *database.table_name* TO *dbmi_user*;

  この文は、特定のソーステーブルに対する SELECT アクセス権を付与します。データを読み取るソーステーブルごとに、この付与を繰り返します。

- SAP HANA Cloud ソースの場合、暗号化するために、接続には SAP HANA JDBC カスタム接続プロパティ設定が必要です。SAP HANA Database Ingestion プロパティの［**詳細接続プロパティ**］フィールドに次のプロパティを入力します。

  `encrypt=true&validateCertificate=false`

使用に関する考慮事項:

- 一括取り込みデータベースは、初期ロードジョブと増分ロードジョブでは Red Hat Linux または SUSE Linux 上の SAP HANA および SAP HANA Cloud ソースをサポートしますが、初期ロードジョブと増分ロードジョブの組み合わせではサポートしません。
- データベース取り込み増分ロードジョブは、最大 120 文字の長さのテーブル名をサポートします。
- SAP HANA または SAP HANA Cloud ソースを使用した増分ロードジョブでは、スキーマドリフトオプションはサポートされていません。
- 一括取り込みデータベースでは、初期ロードジョブまたは増分ロードジョブの場合、SAP HANA ソーステーブルにプライマリキーは必要ありません。
- 一括取り込みデータベースは、ターゲットのデフォルトのカラムデータ型にマッピングされている場合でも、次のソースデータ型をサポートしていません。

  - ARRAY

  - BINTEXT

  - BLOB

  - CLOB

  - nclob

  - ST_GEOMETRY

  - ST_POINT

  - TEXT

  一括取り込みデータベースジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

  注: ALPHANUM、BINTEXT、CHAR、および CLOB データ型は、SAP HANA Cloud では使用できません。

  SAP HANA データ型からターゲットデータ型へのデフォルトのマッピングについては、「デフォルトデータ型のマッピング」 (ページ 169)を参照してください。
- 増分ロードジョブの場合、一括取り込みデータベースでは、ソースデータベースに次のテーブルが必要です。

  - PKLOG ログテーブル。変更タイプとタイムスタンプ、トランザクション ID、スキーマ名、テーブル名など、キャプチャされた DML 変更に関するメタデータが含まれます。

  - PROCESSED ログテーブル。最新の変更データキャプチャサイクルの最大シーケンス番号（SCN）が含まれます。

  - シャドー<*スキーマ*>。<*テーブル名*>_CDC テーブル。トランザクション ID やタイムスタンプなどのメタデータとともに、ソーステーブルからキャプチャされた更新の操作前のイメージと、挿入、更新、および削除の操作後のイメージが含まれます。変更がキャプチャされるソーステーブルごとにシャドーテーブルが存在する必要があります。

  また、一括取り込みデータベースは、AFTER DELETE、AFTER INSERT、および AFTER UPDATE トリガを使用して、各ソーステーブルの DML 変更の操作前のイメージと操作後のイメージを取得し、変更のエントリを PKLOG テーブルとシャドー_CDC テーブルに書き込みます。一括取り込みデータベースは、処理された挿入、更新、削除行ごとに、SAP HANA シーケンス値を PKLOG テーブルとシャドー_CDC テーブルにも書き込みます。シーケンス値は、CDC 処理中にシャドー_CDC テーブルの行を PKLOG テーブルの行にリンクします。

タスクをデプロイすると、一括取り込みデータベースは、PKLOG、PROCESSED、およびシャドー_CDC テーブル、トリガ、およびシーケンスが存在することを検証します。これらのアイテムが存在しない場合、デプロイ操作は失敗します。

- タスクウィザードの**［ソース］**ページから、PKLOG、PROCESSED、シャドー_CDC テーブル、トリガ、およびシーケンスを作成する CDC スクリプトをダウンロードまたは実行できます。SAP HANA Database Ingestion プロパティで**［トリガプレフィックス］**の値を指定した場合、生成されたトリガの名前の先頭には *プレフィックス*_が付きます。

  デフォルトでは、トリガはアプリケーションのシステムユーザーをキャプチャします。代わりにトランザクションユーザーをキャプチャする場合は、CDC スクリプトをダウンロードし、スクリプト内の「APPLICATIONUSER」の箇所をすべて「XS_APPLICATIONUSER」に置き換えます。例えば、AFTER DELETE トリガでこの置換を行うと、アーカイブプロセスに関連する削除を識別して除外できます。

- 増分ロードジョブの実行中に、テーブルのサイズを維持するために、古いレコードを PKLOG テーブルおよびシャドー_CDC テーブルから削除するハウスキーピングがいくつか行われます。PKLOG テーブルとシャドー_CDC テーブルの自動ハウスキーピングを有効にするには、SAP HANA Database Ingestion 接続プロパティの**［ログのクリア］**フィールドで、0 より大きい値を指定します。デフォルト値は 14 日間で、最大値は 366 です。0 の値を指定すると、ハウスキーピングは無効になります。
  ハウスキーピングは増分ロードジョブの実行中に行われます。複数のジョブが異なるテーブルに対して実行されている場合、各ジョブは PKLOG テーブルと、そのジョブに対してのみ定義されているシャドー_CDC テーブルに対してハウスキーピングを実行します。ジョブからソーステーブルを削除しても、対応するシャドー_CDC テーブルのパージは行われません。

- SAP HANA ソースと Microsoft Azure Synapse Analytics ターゲットを持つデータベース取り込みタスクのデプロイで、ソーステーブルに長さの長い複数カラムのプライマリキーが含まれている場合、デプロイに失敗することがあります。この場合、プライマリキーの長さを短くしてから、タスクを再度デプロイしてください。

Teradata ソース

データベース取り込みタスクで Teradata ソースを使用するには、最初にソースデータベースを準備し、使用に関する考慮事項を確認してください。

ソースの準備:

- Teradata ソースを含むデータベース統合タスクをデプロイして実行するには、ソース接続で初期ロード操作を実行するために必要な特権を持つデータベースユーザーを指定する必要があります。次の SQL 文を使用して、これらのユーザーに特権を付与します。

  `GRANT SELECT ON database_name TO user_name/user_role;`

使用に関する考慮事項:

- Teradata ソースを持つデータベース取り込みジョブは、PostgreSQL と SQL Server を除く任意のターゲットタイプを持つことができます。
- 一括取り込みデータベースでは、次の Teradata データ型はサポートされていません。

  - ARRAY

  - Blob

  - CLOB

  - JSON

  - ST_GEOMETRY

  - XML

  サポートされているソースデータ型からターゲットデータ型へのデフォルトのマッピングについては、「デフォルトデータ型のマッピング」 (ページ 169)を参照してください。

一括取り込みデータベースターゲット - 準備と使用

データベース統合タスクを初期ロード、増分ロード、または初期ロードと増分ロードの組み合わせ操作に設定する前に、予期しない結果を回避するために、ターゲットタイプについて次のガイドラインを確認してください。

Amazon Redshift ターゲット

次のリストは、Amazon Redshift ターゲットを準備および使用する際の考慮事項を示しています。

- データベース取り込みジョブでは、Amazon Redshift または Amazon Redshift Serverless ターゲットを使用できます。
- Amazon Redshift ターゲットテーブルにデータを書き込む前に、データベース取り込みジョブはデータを Amazon S3 バケットにステージングします。データベース取り込みタスクを設定するときに、バケットの名前を指定する必要があります。取り込みジョブは COPY コマンドを使用して、Amazon S3 バケットから Amazon Redshift ターゲットテーブルにデータをロードします。COPY コマンドの詳細については、Amazon Web Services のドキュメントを参照してください。
- Amazon Redshift ターゲットの接続を定義するときに、データベース取り込みジョブによってデータをステージングして Amazon Redshift ターゲットテーブルにロードする、Amazon S3 バケットのアクセスキーとシークレットアクセスキーを指定します。
- 増分ロードジョブと、初期ロードジョブと増分ロードジョブの組み合わせにより、ターゲット上に INFORMATICA_CDC_RECOVERY という名前のリカバリテーブルが生成され、内部サービス情報が格納されます。リカバリテーブルのデータによって、障害後に再開されたジョブが以前に処理されたデータを再度プロパゲートすることが防止されます。リカバリテーブルは、ターゲットテーブルと同じスキーマで生成されます。
- データベース取り込みジョブは、ソーステーブルの FLOAT カラムの無限値と NaN 値を、Amazon Redshift ターゲットテーブルでは null 値に変換する。

Amazon S3、フラットファイル、Google Cloud Storage、Microsoft Azure Data Lake Storage、Microsoft Fabric OneLake、および Oracle Cloud Object Storage ターゲット

次のリストは、Amazon S3、フラットファイル、Google Cloud Storage、Microsoft Azure Data Lake Storage、Microsoft Fabric OneLake、および Oracle Cloud Infrastructure（OCI）Object Storage ターゲットを使用する際の考慮事項を示しています。

- Microsoft Fabric OneLake は、初期ロードジョブでのみターゲットとして使用できます。
- Amazon S3、Google Cloud Storage、Microsoft Azure Data Lake Storage、Microsoft Fabric OneLake、または Oracle Cloud Object Storage ターゲットを持つデータベース取り込みタスクを定義する場合、ターゲットに適用するソースデータを含む生成された出力ファイルに対して CSV 形式、Avro 形式、または Parquet 形式のいずれかを選択できます。フラットファイルターゲットの場合、出力ファイル形式として CSV または Avro を選択できます。
- ［CSV］出力形式を選択した場合、一括取り込みデータベースは、ソーステーブルごとに次のファイルをターゲットに作成します。
  - スキーマを記述し、ターゲット上の出力ファイルのいくつかの設定を含む schema.ini ファイル。
  - ソースデータを含む、ソーステーブルごとの 1 つ以上の出力ファイル。一括取り込みデータベースは、日付と時刻が追加されたソーステーブルの名前に基づいて、これらのテキストファイルに名前を付けます。

schema.ini ファイルには、対応する出力ファイルの行の一連のカラムが一覧表示されます。次の表で、schema.ini ファイルのカラムについて説明します。

カラム	説明
ColNameHeader	ソースデータファイルにカラムヘッダーが含まれているかどうかを示します。
Format	出力ファイルの形式を示します。一括取り込みデータベースはカンマ（,）を使用してカラムの値を区切ります。
CharacterSet	出力ファイルに使用される文字セットを指定します。一括取り込みデータベースは UTF-8 文字セットでファイルを生成します。
COL*<sequence_number>*	カラムの名およびデータ型。 **注意事項**: - タスクウィザードの［**ターゲット**］ページの［**詳細**］で［**操作の追加...**］プロパティのいずれかを選択した場合、カラムのリストには、操作のタイプ、時間、所有者、またはトランザクション ID のメタデータカラムが含まれます。 - ［**前のイメージを追加**］チェックボックスを選択した場合、ソースカラムごとにジョブが UNDO データの *column_name*_OLD カラムと REDO データの *column_name*_NEW カラムを作成します。

重要: schema.ini ファイルは編集しないでください。

- ［Avro］出力フォーマットを選択した場合、Avro 形式タイプ、ファイル圧縮タイプ、Avro データ圧縮タイプ、および各ソーステーブルに対して生成された Avro スキーマ定義を格納するディレクトリを選択できます。スキーマ定義ファイルの命名パターンは、*スキーマ名_テーブル名*.txt です。
- Parquet 出力形式を選択した場合、必要に応じて、Parquet がサポートする圧縮タイプを選択できます。
- フラットファイル、Microsoft Azure Data Lake Storage、および Microsoft Fabric OneLake ターゲットでは、一括取り込みデータベースは空のソーステーブルごとに空のディレクトリを作成します。一括取り込みデータベースは、Amazon S3、Google Cloud Storage、および Oracle Cloud Object Storage ターゲットに空のディレクトリを作成しません。
- Amazon S3 接続プロパティでアクセスキーと秘密鍵を指定しない場合、一括取り込みデータベースは DefaultAWSCredentialsProviderChain クラスによって実装されているデフォルトの資格情報プロバイダチェーンを使用して、AWS 資格情報を見つけようとします。詳細については、*Amazon Web サービス*のドキュメントを参照してください。
- データベース取り込み増分ロードジョブ、および初期ロードジョブと増分ロードジョブの組み合わせが、ソースのプライマリキー値を CSV 出力形式を使用するこれらのターゲットのいずれかに変更する更新操作をレプリケートする場合、ジョブは各更新レコードをターゲットでの 2 つのレコード（削除とそれに続く挿入）として処理します。削除には更新前のイメージが含まれています。挿入には同じ行の更新後のイメージが含まれています。

  プライマリキー値を変更しない更新操作の場合、データベース取り込みジョブは各更新を 1 つの操作として処理し、更新後のイメージのみをターゲットに書き込みます。

  注: ソーステーブルにプライマリキーがない場合、一括取り込みデータベースはすべてのカラムがプライマリキーの一部であるかのようにテーブルを扱います。この場合、各更新操作は、削除とそれに続く挿入として処理されます。
- 一括取り込みジョブは、データが Amazon S3、フラットファイル、Microsoft Azure Data Lake Storage、または Microsoft Fabric OneLake ターゲットに送信されるときに、16 進数形式でバイナリデータをアンロードします。各 16 進数カラムの値には、「0x」プレフィックスが付いています。出力ファイルを使用して

データをターゲットにロードする場合は、ファイルを編集して「0x」プレフィックスを削除する必要がある場合があります。

- Windows で Secure Agent サービスを実行していて、フラットファイル接続を使用する場合は、Secure Agent のログオンアカウントが管理者アカウントであることを確認してください。これを行わないと、フラットファイル接続を設定しようとしたときにエラーが発生します。

Databricks Delta ターゲット

データベース取り込みタスクで Databricks Delta ターゲットを使用するには、最初にターゲットを準備し、使用に関する考慮事項を確認してください。

ターゲットの準備:

1. DatabricksJDBC42 JDBC ドライババージョン 2.6.25 を Databricks JDBC ドライバダウンロード Web サイトからダウンロードします。

   一括取り込みと Cloud データ統合の両方に同じドライバを使用する場合は、データ統合のコネクタパッケージにドライバの 2.6.22 バージョンが用意されていることに注意してください。この場合、Databricks Web サイトからダウンロードする代わりに、DatabricksJDBC42.jar ファイルをデータ統合コネクタパッケージからコピーします。

   注: 一括取り込みでは、バージョン 2.6.5 以降またはバージョン 2.6.22 を使用できます。バージョン 2.6.5 以降を使用している場合、Administrator インタフェースの **[接続のテスト]** 機能は機能しません。

2. 次のディレクトリに DatabricksJDBC42.jar ファイルをコピーします。

   Secure_Agent_installation_directory/apps/Database_Ingestion/ext/

   古い SparkJDBC42.jar ファイルがすでに Secure Agent インストールディレクトリにインストールされている場合は、そのファイルを削除します。

3. Databricks Delta 接続プロパティで、**[JDBC ドライバクラス名]** プロパティを `com.databricks.client.jdbc.Driver` に設定します。

   注: 古い Cloud データ統合ドライバをダウンロードした場合は、クラス名として `com.simba.spark.jdbc.Driver` を指定します。

4. Windows では、Secure Agent が実行されているコンピュータに Visual Studio2013 用の Visual C++再配布可能パッケージをインストールします。

使用に関する考慮事項:

- 増分ロードジョブの場合、すべてのソースカラムに対して変更データキャプチャ（Change Data Capture: CDC）を有効にする必要があります。
- 次のストレージタイプの上に作成された Databricks Delta テーブルにアクセスできます。

  - Microsoft Azure Data Lake Storage（ADLS）Gen2

  - Amazon Web Services（AWS）S3

  Databricks Delta 接続は、JDBC URL を使用して Databricks クラスタに接続します。ターゲットを設定するときは、クラスタへの接続に使用する JDBC URL と資格情報を指定します。また、ターゲットが Amazon S3 または ADLS Gen2 のステージングロケーションに接続するために使用する接続情報を定義します。

- Databricks Delta ターゲットテーブルにデータを書き込む前に、データベース取り込みジョブはデータを Amazon S3 バケットまたはまたは ADLS ディレクトリにステージングします。データベース取り込みタスクを設定するときに、データのディレクトリを指定する必要があります。

  注: 一括取り込みデータベース一括取り込みデータベースは、ディレクトリを決定する際に Databricks Delta 接続プロパティの **[ADLS ステージングファイルシステム名]** と **[S3 ステージングバケット]** プロパティは使用しません。

- 一括取り込みデータベースは、1 回だけ実行されるジョブを使用して、Amazon S3 または Azure Data Lake Storage Gen2 のステージングファイルから外部テーブルにデータをロードします。
  デフォルトでは、一括取り込みデータベースは、Databricks Delta 接続プロパティで指定されたクラスタでジョブを実行します。別のクラスタでジョブを実行する場合は、データベース取り込みタスクウィザードの**［ターゲット］**ページで dbDeltaUseExistingCluster カスタムプロパティを false に設定します。
- Databricks Delta 接続プロパティで指定されたクラスタが稼働していない場合、データベース取り込みジョブはクラスタが開始するまで待機します。デフォルトでは、ジョブは 10 分間待機します。クラスタが 10 分以内に開始されない場合、接続がタイムアウトし、ジョブのデプロイが失敗します。
  接続のタイムアウト値を増やす場合は、dbClusterStartWaitingTime カスタムプロパティを、クラスタが稼働するまで取り込みジョブが待機する必要がある最大時間（秒単位）に設定します。データベース取り込みタスクウィザードの**［ターゲット］**ページでこのカスタムプロパティを設定できます。
- デフォルトでは、一括取り込みデータベースは、Databricks Delta の COPY INTO 機能を使用して、ステージングファイルから Databricks Delta ターゲットテーブルにデータをロードします。データベース取り込みタスクウィザードの**［ターゲット］**ページで writerDatabricksUseSqlLoad カスタムプロパティを false に設定すると、すべてのロードタイプでこれを無効にできます。
- AWS クラスタを使用する場合は、Databricks Delta 接続プロパティの**［S3 サービスリージョナルエンドポイント］**の値を指定する必要があります。以下に例を示します。

  ```
  s3.us-east-2.amazonaws.com
  ```

  Linux で Secure Agent を使用して Databricks Delta 接続をテストするには、Databricks Delta 接続プロパティの**［SQL エンドポイント JDBC URL］**フィールドで JDBC URL を指定する必要があります。接続をテストしたら、**［SQL エンドポイント JDBC URL］**の値を削除します。そうしないと、その接続を使用するデータベース取り込みタスクを定義するときに、一括取り込みで JDBC URL、必要な**［Databricks ホスト］**、**［クラスタ ID］**、**［組織 ID］**、および**［Databricks トークン］**の値を使用してターゲットに接続しようとしてログインに失敗するため、設計時エラーが発生します。
- Windows で Secure Agent を使用して Databricks Delta 接続をテストすることはできません。テストは失敗します。このような場合は、Linux で Secure Agent を使用してテストを実行できます。ただし、データベース取り込みタスクを作成するとき、またはデータベース取り込みジョブを実行するときに、Windows 上の Secure Agent で Databricks Delta 接続を使用できます。
- 基になる Parquet ファイルを書き換える必要なく、Databricks Delta ターゲット テーブルに対してカラム名の変更操作を処理するには、Databricks Runtime 10.2 以降の Databricks Delta カラムマッピング機能が必要です。タスクウィザードの**［スケジュールおよびランタイムオプション］**ページで**［カラム名の変更］**オプションを**［レプリケート］**に設定した場合、タスクのデプロイ後、ジョブを実行する前に、生成されたターゲットテーブルを変更して、次の Databricks テーブルプロパティを設定する必要があります。

  ```
  ALTER TABLE <target_table> SET TBLPROPERTIES (
  'delta.columnMapping.mode' = 'name',
  'delta.minReaderVersion' = '2',
  'delta.minWriterVersion' = '5')
  ```

  これらのプロパティにより、必要なリーダーとライターのバージョンの Databricks Delta カラムマッピング機能が有効になります。これらのプロパティを設定しない場合、データベース取り込みジョブは失敗します。
- Databricks Delta ターゲットを含むデータベース取り込みジョブは、Databricks Unity Catalog からターゲットテーブルを生成するためのスキーマ情報を取得できます。Unity Catalog 内の情報へのアクセスを有効にするには、Databricks Delta 接続プロパティの**［カタログ名］**フィールドにカタログ名を指定します。カタログ名は、データウェアハウスの**［SQL ウェアハウス JDBC URL］**値に追加されます。

  注: カタログの使用は SQL ウェアハウスの場合はオプションであり、ジョブクラスタには適用されません。

  Unity Catalog を使用する場合は、個人用ストレージの場所が自動的にプロビジョニングされます。個人用のステージング場所を使用するには、接続プロパティの**［ステージング環境］**フィールドで**［個人用のステージング場所］**を選択します。その後、取り込みジョブの Parquet データファイルをローカルの個人用ストレージの場所にステージングできます。データ保持期間は 7 日間です。デフォルトでは、ステージング場所は AWS または Azure のルートの場所 stage://tmp/<user_name>です。<user_name>は、**［データベースト**

ークン】接続プロパティから取得されます。このユーザーには、個人用のステージング場所に対する読み取りおよび書き込みアクセス権が必要です。

- 一括取り込みデータベースは、テーブルの特殊文字とソーステーブルのカラム名をサポートします。特殊文字は、Databricks Delta ターゲットテーブルまたはカラム名でアンダースコア（_）に置き換えられます。カスタムプロパティのキーと値のペア targetReplacementValue=toHex は、一括取り込みデータベースによって、生成されたターゲットスキーマで特殊文字がアンダースコアに置換されないようにし、特殊文字を 16 進形式に変換します。

  特殊文字を 16 進数値に変換するには、データベース統合タスクをデプロイする前に、次のアクションを実行します。

  1. metadata-manager レイヤーで使用するプロパティファイルを作成します。
     targetReplacementValue=toHex のキーと値のペアをプロパティファイルに追加します。
  2. 管理者の**［ランタイム環境］**ページを開き、Secure Agent を編集します。**［カスタム構成の詳細］**領域でカスタムプロパティを作成します。
     - **［データベース取り込み］**サービスを選択します。
     - **［DBMI_AGENT_ENV］**タイプを選択します。
     - プロパティ名として DBMI_TASK_OVERRIDE_PROPERTIES と入力します。
     - プロパティ値としてプロパティファイルの場所を入力します。
  3. タスクウィザードの**［ターゲット］**ページで、targetReplacementValue カスタムプロパティを toHex に設定します。

     タスクを実行する前に、<jobname>をプロパティファイルの targetReplacementValue キーに追加します。

     ```
     <jobname>.targetReplacementValue=toHex
     ```

     プロパティがすべてのジョブに影響する場合は、「alljobs」を targetReplacementValue キーに追加します。

     ```
     alljobs.targetReplacementValue=toHex
     ```

- Databricks Delta ターゲットのカラムにレプリケートするソースの小数値カラムまたは数値カラムを選択する場合は、各ソースの小数値カラムまたは数値カラムのスケールが精度の範囲内であることを確認してください。そうしないと、ジョブの実行時に、小数点スケールが無効だと報告するエラーが発行されます。この考慮事項は、Databricks Delta ターゲットにマッピングされているすべてのソースタイプに適用されます。

Google BigQuery ターゲット

次のリストは、Google BigQuery ターゲットを準備および使用する際の考慮事項を示しています。

ターゲットの準備

- Google BigQuery JDBC ドライバをダウンロードしてインストールします。
  1. Google BigQuery JDBC ドライババージョン 1.2.25.1029 を Google Cloud の Web サイトからダウンロードします。
  2. インストール zip 内のすべての jar ファイルを次のディレクトリにコピーします。
     Secure Agent installation directory/apps/Database_Ingestion/ext/
  3. Secure Agent を再起動します。
- Google BigQuery と Google Cloud Storage にアクセスするためのサービスアカウントが Google アカウントにあることを確認します。

- サービスアカウントに client_email、project_id、private_key、および regionID の値があることを確認してください。Google BigQuery 接続を作成するときに、対応する **[サービスアカウント ID]**、**[プロジェクト ID]**、**[サービスアカウントキー]**、および **[リージョン ID]** 接続プロパティに値を入力します。
- Google BigQuery 接続のタイムアウト間隔を設定する場合は、接続プロパティの **[オプションのプロパティを指定]** フィールドでタイムアウト間隔プロパティを指定します。次の形式を使用します。

  `"timeout": "<timeout_interval_in_seconds>"`
- 次のエンティティに対する読み取りおよび書き込みアクセスがあることを確認します。

  - ターゲットテーブルを含む Google BigQuery データセット。

  - 一括取り込みデータベースがステージングファイルを作成する Google Cloud Storage パス。
- Google BigQuery ターゲットを使用するには、必要な権限を設定する必要があります。まず、Google Cloud プロジェクトで IAM と管理者のサービスアカウントを作成してカスタムロールを割り当てます。次に、アカウントのカスタムロールに次の権限を追加します。

  - bigquery.datasets.get - データセットに関するメタデータを取得します。

  - bigquery.jobs.create - ジョブとクエリを実行します。

  - bigquery.models.create - 新しいモデルを作成します。

  - bigquery .models.delete - モデルを削除します。

  - bigquery .models.export - モデルをエクスポートします。

  - bigquery.models.getData - モデルデータを取得します。bigquery.models.getMetadata 権限も指定する必要があります。

  - bigquery.models.getMetadata - モデルのメタデータを取得します。bigquery.models.getData 権限も指定する必要があります。

  - bigquery .models.list - モデルとモデルのメタデータを一覧表示します。

  - bigquery.models.updateData - モデルデータを更新します。bigquery.models.updateMetadata 権限も指定する必要があります。

  - bigquery.models.updateMetadata - モデルのメタデータを更新します。bigquery.models.updateData 権限も指定する必要があります。

  - bigquery.routines.create - ストアドプロシージャを含む新しいルーチンを作成します。

  - bigquery.routines.delete - ルーチンを削除します。

  - bigquery.routines.get - ルーチンの定義とメタデータを取得します。

  - bigquery.routines.list - ルーチンとルーチンのメタデータを一覧表示します。

  - bigquery.routines.update - ルーチンの定義とルーチンのメタデータを更新します。

  - bigquery.routines.updateTag - ルーチンのタグを更新します。

  - bigquery.tables.create - 新しいテーブルを作成します。

  - bigquery.tables.delete - テーブルを削除します。

  - bigquery.tables.deleteIndex - テーブルの検索インデックスを削除します。

  - bigquery.tables.deleteSnapshot - テーブルスナップショットを削除します。

  - bigquery.tables.export - BigQuery からテーブルデータをエクスポートします。

  - bigquery.tables.get - テーブルのメタデータを取得します。bigquery.tables.getData 権限も指定する必要があります。

  - bigquery.tables.getData - テーブルデータを取得してクエリを実行します。bigquery.tables.get 権限も指定する必要があります。

  - bigquery.tables.list - テーブルとテーブルのメタデータを一覧表示します。

- bigquery.tables.update - テーブルのメタデータを更新します。bigquery.tables.updateData 権限も指定する必要があります。
- bigquery.tables.updateData - テーブルデータを更新します。bigquery.tables.update 権限も指定する必要があります。
- bigquery.tables.updateTag - テーブルのタグを更新します。
- resourcemanager.projects.get - プロジェクトに関連付けられた請求先アカウントの名前を取得します。
- storage.objects.create - ユーザーがオブジェクトを作成できるようにします。
- storage.objects.delete - オブジェクトを削除する権限を付与します。
- storage.objects.get - バケットメタデータの一覧表示および読み取り時にオブジェクトメタデータを読み取ります。
- storage.objects.list - バケット内のオブジェクトを一覧表示します。

ターゲットの用途

- 一括取り込みデータベースは、ソースデータをバルクモードで Google BigQuery ターゲットにロードします。
- データベース取り込み増分ロードタスクの場合、すべてのソースカラムに対してソースデータベース変更データキャプチャ（CDC）が有効になっていることを確認します。
- 一括取り込みデータベースは、テーブルの特殊文字とソーステーブルのカラム名をサポートします。Google BigQuery ターゲットテーブルまたはカラム名の特殊文字は、アンダースコア（_）に置き換えられます。
  カスタムプロパティのキーと値のペア targetReplacementValue=toHex は、一括取り込みデータベースによって、生成されたターゲットスキーマで特殊文字がアンダースコアに置換されないようにし、特殊文字を 16 進形式に変換します。

  特殊文字を 16 進数値に変換するには、データベース統合タスクをデプロイする前に、次のアクションを実行します。

  1. metadata-manager レイヤーで使用するプロパティファイルを作成します。
     targetReplacementValue=toHex のキーと値のペアをプロパティファイルに追加します。
  2. 管理者の **[ランタイム環境]** ページを開き、Secure Agent を編集します。**[カスタム構成の詳細]** 領域でカスタムプロパティを作成します。
     - **[データベース取り込み]** サービスを選択します。
     - **[DBMI_AGENT_ENV]** タイプを選択します。
     - プロパティ名として DBMI_TASK_OVERRIDE_PROPERTIES と入力します。
     - プロパティ値としてプロパティファイルの場所を入力します。
  3. タスクウィザードの **[ターゲット]** ページで、targetReplacementValue カスタムプロパティを toHex に設定します。

  タスクを実行する前に、<jobname>をプロパティファイルの「targetReplacementValue」キーに追加します。

  ```
  <jobname>.targetReplacementValue=toHex
  ```

  プロパティがすべてのジョブに影響する場合は、「alljobs」を「targetReplacementValue」キーに追加します。

  ```
  alljobs.targetReplacementValue=toHex
  ```
- データベース取り込みタスクで **[監査]** 適用モードを選択した場合は、タスクウィザードの **[ターゲット]** ページの **[詳細]** で、ターゲットテーブルに含める監査メタデータカラムを選択できます。**[メタデータカラムのプレフィックス]** フィールドに値を指定する場合は、特殊文字を含めないようにしてください。特殊文字を含めた場合、タスクのデプロイメントが失敗します。

- デフォルトでは、次のタイプのソースカラムは、長さの指定なしで Google BigQuery の文字列カラムにマッピングされます。

  - 文字データ型を持つソースカラム

  - longdate または timestamp データ型を持つ SAP HANA ソースカラム

- データベース取り込みジョブをデプロイすると、一括取り込みデータベースはデフォルトでは、プライマリキーカラムまたはユニークキーカラムでクラスタ化された Google BigQuery ターゲットテーブルを生成します。各キーカラムには、Google BigQuery がクラスタリングでサポートする次のいずれかのデータ型が必要です。

  - STRING

  - INT64

  - NUMERIC

  - BIGNUMERIC

  - DATE

  - DATETIME

  - TIMESTAMP

  - BOOL

  - GEOGRAPHY

  プライマリキーまたはユニークキーのカラムにサポートされていないデータ型がある場合、クラスタリング時にそのカラムはスキップされます。例えば、プライマリキーに C1、C2、C3、C4、C5 カラムが含まれていて、C2 にサポートされていないデータ型がある場合、ターゲットテーブルは CLUSTER BY 句に C1、C3、C4、および C5 カラムを使用して作成されます。

Kafka ターゲットと Kafka 対応 Azure Event Hubs ターゲット

次のリストは、Kafka ターゲットを使用する際の考慮事項を示しています。

- 一括取り込みデータベースは、増分ロードジョブのターゲットとして、Apache Kafka、Confluent Kafka、Amazon Managed Streaming for Apache Kafka（MSK）、および Kafka 対応 Azure Event Hubs をサポートします。これらすべての Kafka ターゲットタイプは、Kafka 接続タイプを使用します。

  Kafka ターゲットタイプを指定するには、タスク定義または Kafka 接続プロパティで Kafka プロデューサプロパティを指定する必要があります。タスクのこれらのプロパティを指定するには、タスクウィザードの **［ターゲット］** ページの **［プロデューサ設定プロパティ］** フィールドに、*key*:*value* ペアのカンマ区切りのリストを入力します。Kafka 接続を使用するすべてのタスクのプロデューサプロパティを指定するには、プロパティのリストを接続プロパティの **［追加接続プロパティ］** フィールドに入力します。タスクレベルでプロデューサプロパティを定義することにより、特定のタスクの接続レベルのプロパティをオーバーライドできます。プロデューサプロパティの詳細については、Apache Kafka、Confluent Kafka、Amazon MSK、または Kafka 用 Azure Event Hubs のドキュメントを参照してください。

- **［AVRO］** を Kafka ターゲットの出力形式として選択した場合、一括取り込みデータベースは次の形式の名前で、各テーブルのスキーマ定義ファイルを生成します。

  schemaname_tablename.txt

  ソーススキーマの変更により、増分ロードジョブのターゲットが変更されることが予想される場合は、一括取り込みデータベースがタイムスタンプを含む一意の名前で Avro スキーマ定義ファイルを再生成します。

  schemaname_tablename_YYYYMMDDhhmmss.txt

  この一意の命名パターンにより、古いスキーマ定義ファイルが監査目的で保持されます。

- Confluent Schema Registry を使用してスキーマを格納する Confluent Kafka ターゲットがある場合は、タスクウィザードの［**ターゲット**］ページで次の設定を行う必要があります。

  -［**出力形式**］フィールドで、［**AVRO**］を選択します。

  -［**Avro シリアル化形式**］フィールドで、［**なし**］を選択します。

- Kafka プロデューサプロパティをタスクウィザードの［**ターゲット**］ページの［**プロデューサ設定プロパティ**］フィールド、または Kafka 接続プロパティの［**追加接続プロパティ**］フィールドいずれかで指定できます。ビジネスニーズに合うように、Kafka ベンダーによってサポートされている property=value のペアを入力します。

  例えば、Confluent Kafka を使用する場合は、［**プロデューサ設定プロパティ**］フィールドまたは［**追加接続プロパティ**］フィールドで次のエントリを使用してスキーマレジストリの URL を指定し、基本認証を有効にすることができます。

  ```
  schema.registry.url=http://schema-registry:8081,
  key.serializer=org.apache.kafka.common.serialization.StringSerializer,
  value.serializer=io.confluent.kafka.serializers.KafkaAvroSerializer,
  basic.auth.credentials.source=USER_INFO,
  basic.auth.user.info=myname:mypassword
  ```

  Amazon MSK を使用する場合は、次の［**追加接続プロパティ**］エントリを使用して、Amazon MSK ターゲットにアクセスするための IAM ロール認証を有効にすることができます。

  ```
  security.protocol=SASL_SSL,sasl.mechanism=AWS_MSK_IAM,sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;,sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
  ```

  必ず Secure Agent がインストールされている Amazon EC2 インスタンスで IAM ロール認証を有効にしてください。

  Kafka プロパティの詳細については、Kafka ベンダーのドキュメントを参照してください。

- データベース取り込み増分ロードジョブで、Confluent Kafka、Amazon MSK、および Azure Event Hubs ターゲットを含め、SASL_SSL で保護されたアクセスをサポートする Kafka ターゲットに変更データをレプリケートできます。Administrator で、［**追加接続プロパティ**］フィールドの適切なプロパティをはじめとする Kafka 接続を設定する必要があります。例えば、Azure Event Hubs の場合、次の［**追加接続プロパティ**］エントリを使用して、SASL_SSL を有効にすることができます。

  ```
  bootstrap.servers=NAMESPACENAME.servicebus.windows.net:9093
  security.protocol=SASL_SSL
  sasl.mechanism=PLAIN
  sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required
  username="$ConnectionString" password="{YOUR.EVENTHUBS.CONNECTION.STRING}";
  ```

Kafka ターゲットのカスタムメッセージキーの生成

Avro 形式を使用するすべての Kafka ターゲットタイプに対し、ソーステーブルごとに 1 つ以上のカラムで構成されるカスタムメッセージキーを生成するルールを設定できます。ルールの設定後、Kafka ターゲットを持つデータベース取り込み増分ロードジョブは、ターゲットメッセージングシステムに送信するメッセージのヘッダーに、生成されたメッセージキーをを含めることができます。ターゲットメッセージングシステムはメッセージキーを使用して、特定のキー値を持つメッセージをマルチパーティショントピック内の同じパーティションに書き込むことができます。

この機能を実装するには、各ソーステーブルのキーカラムを識別するルールを含んだ構成ファイルを手動で作成する必要があります。次に、タスクウィザードのカスタム設定プロパティにファイルを指定します。

構成ファイルの作成

テキストエディタでルール構成ファイルを作成し、Secure Agent システム上の場所に保存します。このファイルには、各ソーステーブルのルールが含まれています。各ルールは、トピックパーティションへのデータの書き込みに使用するカスタムキーカラムを定義します。

注: データベース取り込みタスクがデプロイされた後にルールを変更または追加するか、他のパラメータのいずれかを変更した場合、ルールの変更を有効にするには、タスクを再デプロイする必要があります。

ルールの構文:

次の構文を使用して構成ファイルにルールを定義します。

```
rule=(schema.tablename,column1,column2,column3,... )
additional rules...
[tableNotFound=ABORT]
[trace={true|false}]
[delimiter=character]
```

ファイルにコメントを含めるには、各コメント行を番号（#）記号で始めます。例:

```
#This text is for informational purposes only.
```

パラメータ:

- **rule**。ソーステーブルの複合メッセージキーを生成するためのルールを定義します。各ルールでは、最初にソーステーブルのスキーマとテーブル名を特定します。スキーマを変更するか、ターゲットのテーブルの名前変更ルールを定義する場合は、ターゲットのスキーマまたは名前が変更されたテーブルの名前を使用します。次に、メッセージキーを構成する 1 つ以上のテーブルカラムの名前を指定します。テーブルにカラムが定義されていることを確認します。カラムが定義されていないと、データベース取り込みジョブが失敗します。SQL Server ソースの場合は、データベースの名前も *database* の形式で含めます。*schema*。*tablename*。

  同じルール構成ファイルに複数のルールを定義できます。

  メッセージキーの生成時、一括取り込みデータベースは、各カラム値の文字表現とそれに続く区切り文字を使用します。各カラムの値と区切り文字は、ルール定義にカラムが表示される順序で複合キー値に追加されます。その後、複合キーはレコードの Kafka メッセージキーとして使用されます。メッセージキーに空の値または null 値があるカラムの位置は、区切り文字のみで表されます。

- **delimiter**。*オプション*。生成されたメッセージキーの各キーカラムの値の後に区切り文字として使用される単一の文字を指定します。このパラメータは、ルール構成ファイルに 1 回だけ指定できます。

  デフォルトはセミコロン（;）です。

- **tableNotFound**。*オプション*。このパラメータを ABORT に設定すると、データベース取り込みジョブはソーステーブルのデータの処理を停止し、ルール構成ファイルにテーブルのルール定義がないと失敗します。各ソーステーブルには、複合メッセージキーを適切に生成させるためのルール定義が必要です。このパラメータは構成ファイルに 1 回だけ指定できます。

  このパラメータを指定せず、ルール構成ファイルにテーブルが見つからない場合は、ターゲットメッセージングシステムパラメータのデフォルトのルールによって、レコードに使用するキーが決定されます。

- **trace**。*オプション*。ルール定義に基づくメッセージキー生成のトレースを有効または無効にします。有効な値は以下のとおりです。

  - **true**。ルール定義に基づくメッセージキー生成のトレースを有効にします。

  - **false**。ルール定義に基づくメッセージキー生成のトレースを無効にします。

  このパラメータは、ルール構成ファイルに 1 回だけ指定できます。

  デフォルトは **false** です。

サンプルルール:

```
rule=(testdb.ABC.DEPT,DEPTNO,DNAME)
tableNotFound=ABORT
trace=true
delimiter=;
```

このルールに基づいて生成されたキー出力の例:

```
1234;HR;
```

データベース取り込みタスクの設定

Kafka ターゲットを持つデータベース取り込み増分ロードタスクを作成する場合、次のオプションを設定して、カスタムメッセージキーの生成を有効にする必要があります。

- タスクウィザードの［**ターゲット**］ページで、［**テーブル名をトピック名として使用**］チェックボックスがオフになっていることを確認します。次に、トピック名を［**トピック名**］フィールドに入力します。
- ［**出力形式**］フィールドで、［**Avro**］を選択します。［**Avro 形式**］フィールドで任意の Avro 形式を選択できます。
- ［**カスタムプロパティ**］で **captureColumnValuesFile** プロパティに、Secure Agent システム上で作成したルール構成ファイルを指すパス値を指定します。

Microsoft Azure Synapse Analytics ターゲット

次のリストは、Microsoft Azure Synapse Analytics ターゲットを準備および使用する際の考慮事項を示しています。

- Microsoft Azure Synapse Analytics ターゲットを使用してデータベース統合タスクをデプロイして実行するには、ターゲット接続でターゲットデータベースに対する CONTROL 権限を持つデータベースユーザーを指定する必要があります。ユーザーに CONTROL 権限を付与するには、次の SQL 文を使用します。

  ```
  USE database_name;
  GRANT CONTROL TO user_name;
  ```

  この権限は、初期ロード、増分ロード、および初期ロードと増分ロードの組み合わせジョブに必要です。この権限により、一括取り込みデータベースは、ターゲットテーブルと、外部データソース、外部ファイル形式、データベーススコープの資格情報オブジェクト（データベースに存在しない場合）などのデータベースオブジェクトを作成できます。この権限は、外部データソースおよびデータベーススコープの資格情報オブジェクトを作成するために特に必要です。

  注: マスターキーを手動で作成する必要があります。マスターキーを作成するには、データベースに対する CONTROL 権限が必要です。
- データベース統合ジョブは、データを Microsoft Azure Synapse Analytics ターゲットテーブルに書き込む前に、まず Microsoft Azure Data Lake Storage Gen2 ステージングファイルにデータを送信します。ステージングファイルは、フィールド区切り文字として 16 進数の x1d 区切り文字を使用します。データがターゲットに書き込まれた後、ステージングファイルを含むテーブル固有のディレクトリのコンテンツ全体が削除されます。
- Synapse Analytics 接続で Microsoft Azure Data Lake Storage Gen2 を使用する場合は、Microsoft Azure Data Lake Storage の［**階層型名前空間**］オプションを有効にする必要があります。この設定では、blob ストレージを使用することは推奨されません。
- データベース統合ジョブが Microsoft Azure Synapse Analytics ターゲットにプロパゲートできるソーステーブルのカラム数は、508 カラムを超えてはなりません。
- データベース統合増分ロードジョブと、初期ロードジョブと増分ロードジョブの組み合わせにより、ターゲット上に INFORMATICA_CDC_RECOVERY という名前のリカバリテーブルが生成され、障害後に再開されたジョブが以前に処理されたデータを再度プロパゲートするのを防ぐ内部サービス情報が格納されます。このリカバリテーブルは、ターゲットテーブルと同じスキーマで生成されます。
- データベース統合ジョブが外部テーブルを使用して Microsoft Azure Synapse Analytics ターゲットにデータを読み込むと、ジョブの再開時にこれらのテーブルが再作成される場合でも、ジョブはターゲット上に作成されたログテーブルと外部テーブルを削除しません。

Microsoft SQL Server および Azure SQL データベースのターゲット

Microsoft SQL Server または Microsoft Azure SQL Database ターゲットは、初期ロード、増分ロード、および初期ロードと増分ロードの組み合わせジョブで使用できます。SQL Server ターゲットには、オンプレミス、RDS、および Azure SQL Managed Instance ターゲットが含まれます。

次のリストは、Microsoft SQL Server ターゲットを準備および使用する際の考慮事項を示しています。

- 一括取り込みデータベースでは、PostgreSQL、または Teradata ソースからデータをレプリケートするジョブのターゲットとして Microsoft SQL Server はサポートされていません。
- SQL Server JDBC ドライバは、一括取り込みデータベースとともに提供されます。個別にインストールする必要はありません。
- 一括取り込みデータベースユーザーがターゲットテーブルを作成し、それらのテーブルにデータを書き込むには、少なくとも次のデータベースロールが必要です。

  - db_datareader

  - db_datawriter

  - db_ddladmin
- Administrator で、SQL Server ターゲットに接続するための SQL Server 接続を定義するときに、次の必須の SQL Server プロパティを入力します。

  - SQL Server のバージョン。**[SQL Server 2017]** または **[SQL Server 2019]** を選択します。

  - 認証モード。**[SQL Server 認証]** または **[Windows 認証 v2]** を選択します。

  - ユーザー名

  - パスワード

  - ホスト

  - ポート

  - データベース名

  - スキーマ

  - コードページ

  他の SQL Server プロパティはサポートされていません。
- SQL Server ターゲットを持つデータベース統合増分ロードジョブ、および初期ロードと増分ロードの組み合わせジョブは、ターゲットテーブルスキーマに基づいて、いくつかの追加のメタデータカラムとともに LOG テーブルを生成します。LOG テーブルは、変更データがターゲットにフラッシュされる直前に作成されます。受信 DML データは、ローカル CSV ファイルを SQL Server ドライバの Bulk Copy API に提供することにで LOG テーブルに挿入されます。LOG テーブルの情報に基づいて merge apply 文が生成され、DML 操作が実際のターゲットテーブルに適用されます。DML の変更が適用されると、LOG テーブルは削除されます。
  複数のジョブ、または複数のテーブルを含むジョブを実行している場合、LOG テーブルによって、顧客データベースインスタンスの追加スペースまたはサイズ要件が一時的に急増する可能性があります。LOG テーブルに必要なスペースとサイズは、フラッシュサイクルの一部として受信される行数によって異なります。
- データベース統合増分ロードジョブ、または初期および増分ロードジョブが SQL Server ターゲットにプロパゲートできるソーステーブル内のカラム数は、508 カラムを超えることはできません。ソーステーブルに 508 を超えるカラムが含まれている場合、ジョブは LOG テーブルの作成中に失敗します。
- データベース取り込み増分ロードジョブ、および初期ロードジョブと増分ロードジョブの組み合わせの SQL Server ターゲットには、スキーマドリフトオプションはサポートされていません。

- データベース統合増分ロードジョブと、初期ロードジョブと増分ロードジョブの組み合わせにより、ターゲット上に INFORMATICA_CDC_RECOVERY という名前のリカバリテーブルが生成され、障害後に再開されたジョブが以前に処理されたデータを再度プロパゲートするのを防ぐ内部サービス情報が格納されます。このリカバリテーブルは、ターゲットテーブルと同じスキーマで生成されます。

Oracle ターゲット

次のリストは、Oracle ターゲットを準備および使用する際の考慮事項を示しています。

ターゲットの準備

- 一括取り込みデータベースでは、データを Oracle ターゲットデータベースにロードするために、ユーザーに特定の権限が必要です。オンプレミス Oracle ターゲットの場合、Oracle ターゲットに接続する一括取り込みデータベースユーザー（*cmid_user*）に、次のユーザー特権を付与します。

```
GRANT CREATE SESSION TO cmid_user;

GRANT SELECT ON "PUBLIC".V$DATABASE TO cmid_user;
GRANT SELECT ON "PUBLIC".V$CONTAINERS TO cmid_user;

GRANT SELECT ON DBA_USERS TO cmid_user;
GRANT SELECT ON DBA_TABLES TO cmid_user;
GRANT SELECT ON DBA_OBJECT_TABLES TO cmid_user;
GRANT SELECT ON DBA_INDEXES TO cmid_user;
GRANT SELECT ON DBA_OBJECTS TO cmid_user;

GRANT CREATE TABLE <schema.table> TO cmid_user; <--Unless you grant on ANY TABLE
GRANT SELECT ON ALL_CONSTRAINTS TO cmid_user;
GRANT SELECT ON ALL_OBJECTS TO cmid_user;

GRANT SELECT ON SYS.TAB$ TO cmid_user;
GRANT SELECT ON SYS.RECYCLEBIN$ TO cmid_user;
GRANT SELECT ON SYS.COL$ TO cmid_user; <-- If cmid_user is the owner of the target schema
GRANT SELECT ON SYS.CCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CDEF$ TO cmid_user;
GRANT SELECT ON SYS.OBJ$ TO cmid_user;
GRANT SELECT ON SYS.COLTYPE$ TO cmid_user;
GRANT SELECT ON SYS.ATTRCOL$ TO cmid_user;
GRANT SELECT ON SYS.IDNSEQ$ TO cmid_user;
GRANT SELECT ON SYS.ATTRCOL$ TO cmid_user;
GRANT SELECT ON SYS.IDNSEQ$ TO cmid_user;
GRANT SELECT ON SYS.IND$ TO cmid_user;

-- Grant the following if cmid_user is NOT the owner of the target schema. If you prefer, you
-- can grant to individual target tables and indexes instead of to ANY TABLE or ANY INDEX.
GRANT ALTER SESSION TO cmid_user;
GRANT RESOURCE TO cmid_user;
GRANT SELECT ANY TABLE TO cmid_user;
GRANT SELECT ANY DICTIONARY TO <cmid_user>;
GRANT ALTER ANY TABLE TO cmid_user;
GRANT CREATE ANY TABLE TO cmid_user;
GRANT DROP ANY TABLE TO cmid_user;
GRANT INSERT ANY TABLE TO cmid_user;
GRANT UPDATE ANY TABLE TO cmid_user;
GRANT DELETE ANY TABLE TO cmid_user;
GRANT CREATE ANY INDEX TO cmid_user;
GRANT ALTER ANY INDEX TO cmid_user;
GRANT DROP ANY INDEX TO cmid_user;
```

- Amazon RDS for Oracle ターゲットの場合、RDS にマスタユーザーとしてログインし、Oracle ターゲットに接続する一括取り込みデータベースユーザー（*cmid_user*）に次のユーザー特権を付与します。

```
GRANT CREATE SESSION TO cmid_user;

GRANT SELECT on "PUBLIC".V$DATABASE TO cmid_user;

GRANT SELECT on DBA_USERS TO cmid_user;
```

```
GRANT SELECT on DBA_TABLES TO cmid_user;
GRANT SELECT on DBA_INDEXES TO cmid_user;

GRANT CREATE TABLE <schema.table> TO cmid_user;
GRANT SELECT on SYS.TAB$ TO cmid_user;
GRANT SELECT on SYS.COL$ TO cmid_user;
GRANT SELECT on SYS.OBJ$ TO cmid_user;
GRANT SELECT on SYS.IND$ TO cmid_user;

-- Grant the following if cmid_user is NOT the owner of the target schema. If you prefer, you
-- can grant to individual target tables and indexes instead of to ANY TABLE or INDEX.
GRANT ALTER SESSION TO cmid_user;
GRANT RESOURCE TO cmid_user;
GRANT SELECT ANY TABLE TO cmid_user;
GRANT SELECT ANY DICTIONARY TO <cmid_user>;
GRANT ALTER ANY TABLE TO cmid_user;
GRANT CREATE ANY TABLE TO cmid_user;
GRANT DROP ANY TABLE TO cmid_user;
GRANT INSERT ANY TABLE TO cmid_user;
GRANT UPDATE ANY TABLE TO cmid_user;
GRANT DELETE ANY TABLE TO cmid_user;
GRANT CREATE ANY INDEX TO cmid_user;
GRANT ALTER ANY INDEX TO cmid_user;
GRANT DROP ANY INDEX TO cmid_user;
```

また、次の Amazon RDS プロシージャを実行して、追加の SELECT 権限を *cmid_user* に付与します。

```
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$CONTAINERS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_OBJECT_TABLES',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_OBJECTS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_CONSTRAINTS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_OBJECTS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'RECYCLEBIN$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
```

```
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'CCOL$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'CDEF$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'COLTYPE$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ATTRCOL$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'IDNSEQ$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
```

使用に関する考慮事項:

- デフォルトでは、パフォーマンスを最適化するために、一括取り込みデータベースでは Oracle ターゲットテーブルのロギングが無効になっています。データベース取り込みタスクウィザードの **[ターゲット]** ページで writerOracleNoLogging カスタムプロパティを false に設定すると、ロギングを有効にできます。
- Oracle Database Ingestion 接続プロパティで SSL の使用が有効になっている場合、Oracle ターゲットを持つデータベース取り込み初期ロードジョブは失敗します。

PostgreSQL ターゲット

Amazon Aurora PostgreSQL は、Db2 for i または Oracle ソースを持つ初期ロードジョブ、増分ロードジョブ、および初期ロードと増分ロードの組み合わせジョブでターゲットとして使用できます。

次のリストは、PostgreSQL ターゲットを準備および使用する際の考慮事項を示しています。

- PostgreSQL ソースを含むデータベース取り込みタスクをデプロイして実行するには、ターゲット接続で必要な特権を持つデータベースユーザーを指定する必要があります。
  ユーザーには、接続で指定されたデータベースに対する CONNECT 特権と TEMPORARY 特権、およびターゲットプロパティで指定されたターゲットスキーマに対する USAGE 特権と CREATE 特権が必要です。

  次の SQL 文を使用して、これらの特権をユーザーロールに付与し、そのロールをユーザーに割り当てます。

  ```
  CREATE ROLE dbmi_role;
  GRANT CONNECT ON DATABASE database TO dbmi_role;
  GRANT TEMPORARY ON DATABASE database TO dbmi_role;
  GRANT CREATE ON SCHEMA schema TO dbmi_role;
  ```

```
GRANT USAGE ON SCHEMA schema TO dbmi_role;
CREATE USER dbmi_user with PASSWORD 'password';
GRANT dbmi_role to dbmi_user;
```

注: ジョブの実行時に生成されるターゲットテーブルに対する特権は、ジョブを実行するユーザーに付与されます。

- PostgreSQL ターゲットを使用したデータベース統合増分ロードジョブでは、ターゲットテーブルスキーマに基づいて LOG テーブルが生成され、いくつかのメタデータカラムが追加されます。LOG テーブルは、変更データがターゲットにフラッシュされる直前に作成されます。受信 DML データは、ローカル CSV ファイルを PostgreSQL ドライバの Bulk Copy API に提供することにで LOG テーブルに挿入されます。LOG テーブルの情報に基づいて一連の merge apply 文が生成され、DML 操作が実際のターゲットテーブルに適用されます。DML の変更が適用されると、LOG テーブルは削除されます。
  複数のジョブ、または複数のテーブルを含むジョブを実行している場合、LOG テーブルによって、顧客データベースインスタンスの追加スペースまたはサイズ要件が一時的に急増する可能性があります。LOG テーブルに必要なスペースとサイズは、フラッシュサイクルの一部として受信される行数によって異なります。
- データベース統合増分ロードジョブが PostgreSQL ターゲットにプロパゲートできるソーステーブル内のカラム数は、796 カラムを超えることはできません。ソーステーブルに 796 を超えるカラムが含まれている場合、ジョブは LOG テーブルの作成中に失敗します。
- PostgreSQL でサポートするソースオブジェクト識別子の最大長は 63 文字です。PostgreSQL ターゲットを持つデータベース統合タスクのデプロイは、ソーステーブル名またはカラム名の長さが 63 文字を超えると、検証中に失敗します。
- スキーマドリフトオプションは、データベース取り込みジョブの PostgreSQL ターゲットではサポートされていません。
- データベース統合増分ロードジョブにより、ターゲット上に INFORMATICA_CDC_RECOVERY という名前のリカバリテーブルが生成され、障害後に再開されたジョブが以前に処理されたデータを再度プロパゲートするのを防ぐ内部サービス情報が格納されます。このリカバリテーブルは、ターゲットテーブルと同じスキーマで生成されます。

Snowflake ターゲット

ターゲットの準備

ターゲットの準備は、Snowflake ターゲットテーブルへのデータの高パフォーマンスストリーミングに Superpipe 機能を使用するか、中間ステージファイルにデータを書き込むかによって異なります。

Superpipe を使用

Superpipe 機能を使用する場合は、次の手順を実行します。

1. 一括取り込みユーザーを作成します。次の SQL 文を使用します。

   ```
   create user INFACMI_User password 'Xxxx@xxx';
   ```

2. 新しいユーザーロールを作成し、一括取り込みユーザーに付与します。以下の SQL 文を使用します。

   ```
   create role INFACMI_superpipe;
   grant role INFACMI_superpipe to user INFACMI_User;
   ```

3. Snowflake 仮想ウェアハウスの使用権限を新しいロールに付与します。次の SQL 文を使用します。

   ```
   grant usage on warehouse warehouse_name to role INFACMI_superpipe;
   ```

4. Snowflake データベースの使用権限を新しいロールに付与します。次の SQL 文を使用します。

   ```
   grant usage on database INFACMI_DB1 to role INFACMI_superpipe;
   ```

5. 新しいスキーマを作成します。以下の SQL 文を使用します。

   ```
   use database INFACMI_DB1;
   create schema sh_superpipe;
   ```

6. 新しい Snowflake スキーマに対する create stream、create view、および create table 特権を新しいロールに付与します。次の SQL 文を使用します。

   ```
   grant create stream, create view, create table, usage on schema INFACMI_DB1.sh_superpipe to role INFACMI_superpipe;
   ```

7. 新しく作成されたユーザーのデフォルトのロールを設定します。次の SQL 文を使用します。

   ```
   alter user INFACMI_User set default_role=INFACMI_superpipe;
   ```

8. ターゲットへの Snowflake Data Cloud 接続を定義します。認証方法として **[KeyPair]** オプションを使用する必要があります。「コネクタと接続」>「Snowflake Data Cloud 接続プロパティ」を参照してください。

9. OpenSSL バージョン 3.x.x と PBE-SHA1-2DES または PBE-SHA1-3DES 暗号を使用してプライベートキーを生成します。次の openssl コマンドを使用し、プライベートキーを生成してフォーマットします。

   ```
   openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -v1 PBE-SHA1-3DES -out rsa_key.p8
   ```

10. パブリックキーを生成します。次の openssl コマンドを使用し、-in オプションで暗号化されたプライベートキーを含むファイル（rsa_key.p8）を参照します。

    ```
    openssl rsa -in rsa_key.p8 -pubout -out rsa_key.pub
    ```

11. Snowflake で、パブリックキーを Snowflake ユーザーに割り当てます。次の SQL コマンドを使用します。

    ```
    alter user INFACMI_User set rsa_public_key='key_value';
    ```

次の手順: 取り込みタスクを作成するときに、タスクウィザードの **[ターゲット]** ページで **[Superpipe]** オプションを選択します。必要に応じて、変更データ行がマージされて Snowflake ターゲットテーブルに適用される頻度を制御する **[マージ頻度]** 値を指定することもできます。

Superpipe を使用しない

Snowflake ターゲットに Superpipe 機能を使用しない場合は、ACCOUNTADMIN ユーザーとして次の手順を実行します。

1. 一括取り込みユーザーを作成します。次の SQL 文のいずれかを使用します。

   ```
   create user INFACMI_User password 'Xxxx@xxx';
   ```

   または

   ```
   replace user INFACMI_User password 'Xxxx@xxx';
   ```

2. 新しいロールを作成し、そのロールを一括取り込みユーザーに付与します。以下の SQL 文を使用します。

   ```
   create role INFA_CMI_Role;
   grant role INFA_CMI_Role to user INFACMI_User;
   ```

3. Snowflake 仮想ウェアハウスの使用権限を新しいロールに付与します。次の SQL 文を使用します。

   ```
   grant usage on warehouse CMIWH to role INFA_CMI_Role;
   ```

4. Snowflake データベースの使用権限を新しいロールに付与します。次の SQL 文を使用します。

   ```
   grant usage, CREATE SCHEMA on database CMIDB to role INFA_CMI_Role;
   ```

5. 新しく作成されたユーザーのデフォルトのロールを設定します。次の SQL 文を使用します。

   ```
   alter user INFACMI_User set default_role=INFA_CMI_Role;
   ```

また、INFACMI_User として新しいスキーマを作成します。

```
create schema CMISchema;
```

注: ユーザーのデフォルトロールが取り込みタスクに使用され、必要な権限がない場合、実行時に次のエラーが発行されます。

```
SQL compilation error: Object does not exist, or operation cannot be performed.
```

使用に関する考慮事項:

- 一括取り込みは、Snowflake Data Cloud ターゲットにデータを移動する代替方法を提供しています。
  - 取り込みタスクを定義するときに［**Superpipe**］オプションを選択した場合、取り込みジョブは Snowpipe Streaming API を使用して、短い待ち時間でデータ行をターゲットテーブルに直接ストリーミングします。この方法は、すべてのロードタイプで使用できます。［**KeyPair**］認証を使用する必要があります。
  - Superpipe を使用しない場合、取り込みジョブはまず、タスク定義で指定した名前を持つ内部ステージのデータファイルにデータを書き込みます。
- Superpipe を使用せず、取り込みジョブのターゲットプロパティで指定した内部ステージが存在しない場合、一括取り込みデータベースは、次の SQL コマンドを実行してステージを自動的に作成します。

  `Create stage if not exists "Schema"."Stage_Bucket""`

  コマンドを正常に実行するには、次の特権をユーザーロールに付与する必要があります。

  `GRANT CREATE STAGE ON SCHEMA "Schema" TO ROLE <your_role>;`
- Snowflake Data Cloud ターゲットの接続を定義するときは、［**JDBC URL の追加パラメータ**］フィールドに database= *target_database_name* と設定する必要があります。それ以外の場合は、データベース統合タスクウィザードでターゲットを定義しようとすると、スキーマのリストを取得できないことを示すエラーメッセージが表示されます。
- ［**KeyPair**］オプションを認証方法として使用して Snowflake ターゲットの接続を定義し、OpenSSL 3.x.x バージョンでプライベートキーを生成する場合は、プライベートキーの生成時に PBE-SHA1-2DES または PBE-SHA1-3DES の暗号を使用します。以下のコマンドのいずれかを実行します。

  `openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -v1 PBE-SHA1-3DES -out rsa_key.p8`

  または

  `openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -v1 PBE-SHA1-2DES -out rsa_key.p8`

  PBE-SHA1-2DES 暗号または PBE-SHA1-3DES 暗号なしで汎用コマンドを使用すると、データベース統合タスクウィザードのターゲット定義手順でターゲットスキーマを取得中に、無効またはサポート対象外のプライベートキーに関するエラーメッセージが表示されることがあります。

  OpenSSL 1.1.1 を使用してプライベートキーを生成すると、エラーメッセージは表示されません。
- データベース統合増分ロードジョブと、初期ロードジョブと増分ロードジョブの組み合わせにより、ターゲット上に INFORMATICA_CDC_RECOVERY という名前のリカバリテーブルが生成され、障害後に再開されたジョブが以前に処理されたデータを再度プロパゲートするのを防ぐ内部サービス情報が格納されます。このリカバリテーブルは、ターゲットテーブルと同じスキーマで生成されます。
- Snowflake ターゲットの場合、NUMBER フィールドのスケールを変更したり、既存のフィールドのデータ型を別のデータ型に変更したりすることはできません。Snowflake ではこれらの操作がサポートされていないためです。

Snowflake へのプライベート接続の設定

AWS または Azure プライベートリンクエンドポイントを使用して Snowflake にアクセスできます。

AWS または Azure プライベートリンクの設定によって、Snowflake への接続が AWS または Azure 内部ネットワークを使用して確立され、パブリックインターネットを介して行われないようにします。

プライベート AWS ネットワーク経由で Snowflake アカウントに接続するには、
AWS Private Link and Snowflake を参照してください。

プライベート Azure ネットワーク経由で Snowflake アカウントに接続するには、
Azure Private Link and Snowflake を参照してください。

Amazon S3、Google Cloud Storage、および Azure Data Lake Storage Gen2 ターゲット上の CDC ファイルのデフォルトのディレクトリ構造

データベース取り込みジョブは、Amazon S3、Google Cloud Storage、および Microsoft Azure Data Lake Storage Gen2 ターゲットにディレクトリを作成して、変更データ処理に関する情報を格納します。

次のディレクトリ構造がデフォルトでターゲットに作成されます。

```
Bucket
└──────connection_folder
    └──────job_folder
        ├──────cycle
        │    ├──────completed
        │    │    ├──────completed_cycle_folder
        │    │    │    └──────Cycle-timestamp.csv
        │    │    │         ...
        │    │    └──────completed_cycle_folder
        │    │         └──────Cycle-timestamp.csv
        │    └──────contents
        │         ├──────cycle_folder
        │         │    └──────Cycle-contents-timestamp.csv
        │         │         ...
        │         └──────cycle_folder
        │              └──────Cycle-contents-timestamp.csv
        └──────data
             └──────table_name
                  ├──────data
                  │    ├──────cycle_folder
                  │    │    └──────table_name_timestamp.csv
                  │    │         ...
                  │    └──────cycle_folder
                  │         └──────table_name_timestamp.csv
                  └──────schema
                       └──────V1
                            └──────table_name.schema
```

次の表に、デフォルト構造に含まれるディレクトリを示します。

フォルダ	説明
connection_folder	一括取り込みデータベースのオブジェクトが含まれています。このフォルダは、Amazon S3 接続プロパティの **［フォルダパス］** フィールドまたは Microsoft Azure Data Lake Storage Gen2 接続プロパティの **［ディレクトリパス］** フィールドで指定されています。 **注**: このフォルダは、Google Cloud Storage ターゲット用には作成されていません。
job_folder	ジョブ出力ファイルが含まれています。このフォルダは、データベース取り込みタスクウィザードの **［ターゲット］** ページの **［ディレクトリ］** フィールドで指定されています。
cycle/completed	各完了 CDC サイクルのサブフォルダが含まれています。各完了サイクルサブフォルダには、完了サイクルファイルが含まれています。
cycle/contents	各 CDC サイクルのサブフォルダが含まれます。各サイクルサブフォルダには、サイクルコンテンツファイルが含まれています。
data	各テーブルの出力データファイルとスキーマファイルが含まれています。

フォルダ	説明
data/*table_name*/schema/V1	スキーマファイルが含まれています。 **注**: 出力ファイルが Parquet 形式を使用している場合、一括取り込みデータベースはスキーマファイルをこのフォルダに保存しません。
data/*table_name*/data	出力データファイルを生成する各 CDC サイクルのサブフォルダが含まれています。

サイクルディレクトリ

一括取り込みデータベースは、次のパターンを使用してサイクルディレクトリに名前を付けます。

[dt=]*yyyy-mm-dd-hh-mm-ss*

データベース取り込みタスクウィザードの［**ターゲット**］ページの［**ディレクトリタグの追加**］チェックボックスを選択すると、サイクルフォルダ名に「dt=」プレフィックスが追加されます。

サイクルコンテンツファイル

サイクルコンテンツファイルは cycle/contents/*cycle_folder* サブディレクトリにあります。サイクルコンテンツファイルには、サイクル中に DML イベントが発生した各テーブルのレコードが含まれています。サイクル内のテーブルで DML 操作が発生しなかった場合、そのテーブルはサイクルコンテンツファイルに表示されません。

一括取り込みデータベースは、次のパターンを使用してサイクルコンテンツファイルに名前を付けます。

Cycle-contents-*timestamp*.csv

サイクルコンテンツの csv ファイルには、次の情報が含まれています。

- テーブル名
- サイクル名
- テーブルのサイクルフォルダへのパス
- テーブルの開始シーケンス
- テーブルの終了シーケンス
- 挿入操作の数
- 更新操作の数
- 削除操作の数
- *複合ロードジョブのみ*: 切り捨て操作の数
- *複合ロードジョブのみ*: 初期ロードフェーズ中に発生した挿入操作の数
- *複合ロードジョブのみ*: 初期ロードフェーズ中に発生した削除操作の数
- スキーマバージョン
- スキーマバージョンのスキーマファイルへのパス

  注: 出力データファイルが Parquet 形式を使用している場合、一括取り込みデータベースは、サイクルコンテンツファイルで指定されたパスにスキーマファイルを保存しません。代わりに、データベース取り込みタスクウィザードの［**ターゲット**］ページの［**Avro スキーマディレクトリ**］フィールドで指定されているフォルダ内のスキーマファイルを使用できます。

完了サイクルファイル

完了サイクルファイルは cycle/completed/*completed_cycle_folder* サブディレクトリにあります。データベース取り込みジョブは、サイクルが完了した後、このサブディレクトリにサイクルファイルを作成します。このファイルが存在しない場合、サイクルはまだ完了していません。

一括取り込みデータベースは、次のパターンを使用して完了サイクルファイルに名前を付けます。

`Cycle-`*`timestamp`*`.csv`

完了サイクルの csv ファイルには、次の情報が含まれています。

- サイクル名
- サイクル開始時刻
- サイクル終了時刻
- サイクル終了時の現在のシーケンス番号
- サイクルコンテンツファイルへのパス
- サイクル終了の理由
  有効な理由の値は、以下のとおりです。
  - NORMAL_COMMIT。サイクルが DML 制限に達した後、またはサイクル間隔の終了後に、コミット操作が発生しました。サイクルはコミット境界でのみ終了できます。
  - NORMAL_EXPIRY。サイクル間隔が経過したため、サイクルが終了しました。最後の操作はコミットでした。
  - *複合初期ロードジョブのみ*: BACKLOG_COMPLETED。CDC バックログ処理が完了したため、サイクルが終了しました。CDC バックログは、ジョブの組み合わせの初期ロードフェーズ中にキャプチャされたイベントで構成されます。バックログには、初期ロードフェーズの開始時または終了時、および初期ロードフェーズからメインの CDC 増分処理への移行中にキャプチャされた可能性のある DML 変更が含まれます。
  - *複合ロードジョブのみ:* INITIAL_LOAD_COMPLETED。初期ロードが完了したため、サイクルが終了しました。
  - *複合ロードジョブのみ:* RESYNC_STARTED。テーブルの再同期が開始されたため、サイクルが終了しました。

出力データファイル

データファイルには、次の情報を含むレコードが含まれています。

- 操作タイプ。有効な値は以下のとおりです。
  - I: 挿入操作
  - U: 更新操作
  - D: 削除操作
  - *複合ロードジョブのみ:* T: 切り詰め操作
  - *複合ロードジョブのみ:* X: 複合ロードジョブの初期ロードフェーズ中に発生した削除操作
  - *複合ロードジョブのみ:* Y: 複合ロードジョブの初期ロードフェーズ中に発生した挿入操作
- ソート可能なシーケンス番号。初期ロードジョブと増分ロードジョブの組み合わせでは、ソート可能なシーケンス番号には 20 桁のプレフィックスが含まれており、これを使用して行を再同期バージョンおよびロードジョブに合わせることができます。プレフィックスは次の属性を組み合わせたものです。
  1. インカネーション。この 9 桁の数字は、テーブルが再同期されるたびに増加します。初期値は 1 です。
  2. スキーマバージョン。この 9 桁の数字は、テーブルにスキーマドリフトの変更がプロパゲートされるたびに増加します。初期値は 1 です。

3. フェーズ。この 2 桁の数字は、アンロードからバックログ、CDC への移行が実行されると変化します。有効な値は以下のとおりです。

   •00: 切り詰め（初期ロードまたは再同期中に最初に書き込まれたデータレコード）

   •01: 初期ロードまたは再同期中の通常の挿入

   •02: 初期ロード中に検出された変更

   •03: 初期ロードまたは再同期が完了した後、メインの CDC フェーズに戻る前に検出された変更

   •04: 通常の CDC フェーズで検出された変更

- 日付カラム

  注: レコードの挿入と削除には、操作後のイメージのみが含まれています。レコードの更新には、更新前後のイメージが含まれています。

Amazon S3、Google Cloud Storage、フラットファイル、Microsoft Fabric OneLake、および ADLS Gen2 ターゲット上の出力ファイルのカスタムディレクトリ構造

デフォルトの構造を使用しない場合は、初期ロードジョブ、増分ロードジョブ、または初期ロードと増分ロードの組み合わせジョブが Amazon S3、Google Cloud Storage、フラットファイル、または Microsoft Azure Data Lake Storage（ADLS）Gen2、または Microsoft Fabric OneLake ターゲットに書き込む出力ファイルのカスタムディレクトリ構造を設定できます。

初期ロード

デフォルトでは、初期ロードジョブは出力ファイルを親ディレクトリの下の *tablename_timestamp* サブディレクトリに書き込みます。Amazon S3、フラットファイル、および ADLS Gen2 ターゲットについては、タスクウィザードの **［ターゲット］**ページで **［親としての接続ディレクトリ］**チェックボックスが選択されている場合、親ディレクトリはターゲット接続プロパティで指定されます。

- Amazon S3 接続では、この親ディレクトリは **［フォルダパス］**フィールドで指定されます。
- フラットファイル接続では、この親ディレクトリは **［ディレクトリ］**フィールドで指定されます。
- ADLS Gen2 接続では、この親ディレクトリは **［ディレクトリパス］**フィールドで指定されます。

Google Cloud Storage ターゲットでは、親ディレクトリは、タスクウィザードの **［ターゲット］**ページにある **［バケット］**フィールドで指定されたバケットコンテナです。

Microsoft Fabric OneLake ターゲットの場合、親ディレクトリは、Microsoft Fabric OneLake 接続プロパティの **［レイクハウスのパス］**フィールドで指定されたパスです。

ニーズに合うようにディレクトリ構造をカスタマイズできます。例えば、初期ロードの場合、環境に合わせてファイルを整理したり、ファイルを見つけやすくしたりするために、ルートディレクトリや、接続プロパティで指定された親ディレクトリとは異なるディレクトリパスに出力ファイルを書き込むことができます。または、すべてのファイルの自動処理を容易にするために、タイムスタンプ付きのサブディレクトリにファイルを別々に書き込むのではなく、テーブルのすべての出力ファイルをテーブル名の付いたディレクトリに直接統合できます。

ディレクトリ構造を設定するには、取り込みタスクウィザードの **［ターゲット］**ページにある **［データディレクトリ］**フィールドを使用する必要があります。デフォルト値は{TableName}_{Timestamp}です。これにより、出力ファイルが親ディレクトリの下の *tablename_timestamp* サブディレクトリに書き込まれます。大文字と小文字を区別しないプレースホルダとディレクトリ名の任意の組み合わせで構成されるディレクトリパターンを作成することにより、カスタムディレクトリパスを設定できます。プレースホルダは次のとおりです。

- ターゲットテーブル名は{TableName}

- 初期ロードジョブがターゲットへのデータの転送を開始した日付と時刻は{Timestamp}（yyyymmdd_hhmissms 形式）
- ターゲットスキーマ名は{Schema}
- 2 桁の年は{YY}
- 4 桁の年は{YYYY}
- 2 桁の月の値は{MM}
- 月の 2 桁の日は{DD}

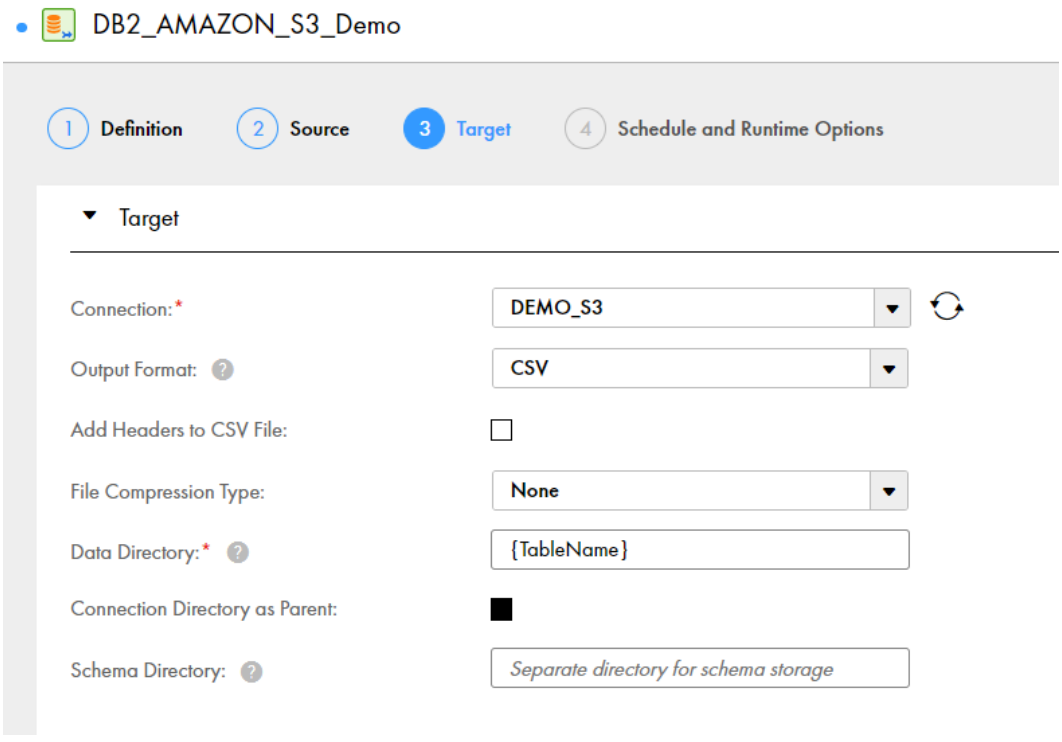
パターンには、次の関数を含めることもできます。

- toLower()は、かっこ内のプレースホルダで表される値を小文字にするのに使用します
- toUpper()は、かっこ内のプレースホルダで表される値を大文字にするのに使用します

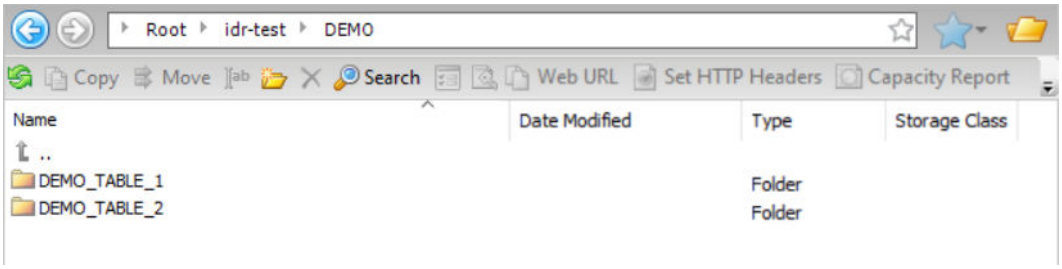
デフォルトでは、ターゲットスキーマもデータディレクトリに書き込まれます。スキーマに別のディレクトリを使用する場合は、**［スキーマディレクトリ］**フィールドでディレクトリパターンを定義できます。

例 1:

Amazon S3 ターゲットを使用していて、出力ファイルとターゲットスキーマを、接続プロパティの**［フォルダパス］**フィールドで指定された親ディレクトリの下にある同じディレクトリに書き込みたいとします。この場合、親ディレクトリは idr-test/DEMO/です。テーブルのすべての出力ファイルを、タイムスタンプなしでテーブル名と一致する名前を持つディレクトリに書き込みたいと考えています。**［データディレクトリ］**フィールドに入力し、**［親としての接続ディレクトリ］**チェックボックスを選択する必要があります。次の画像は、タスクウィザードの**［ターゲット］**ページにあるこの設定を示しています。

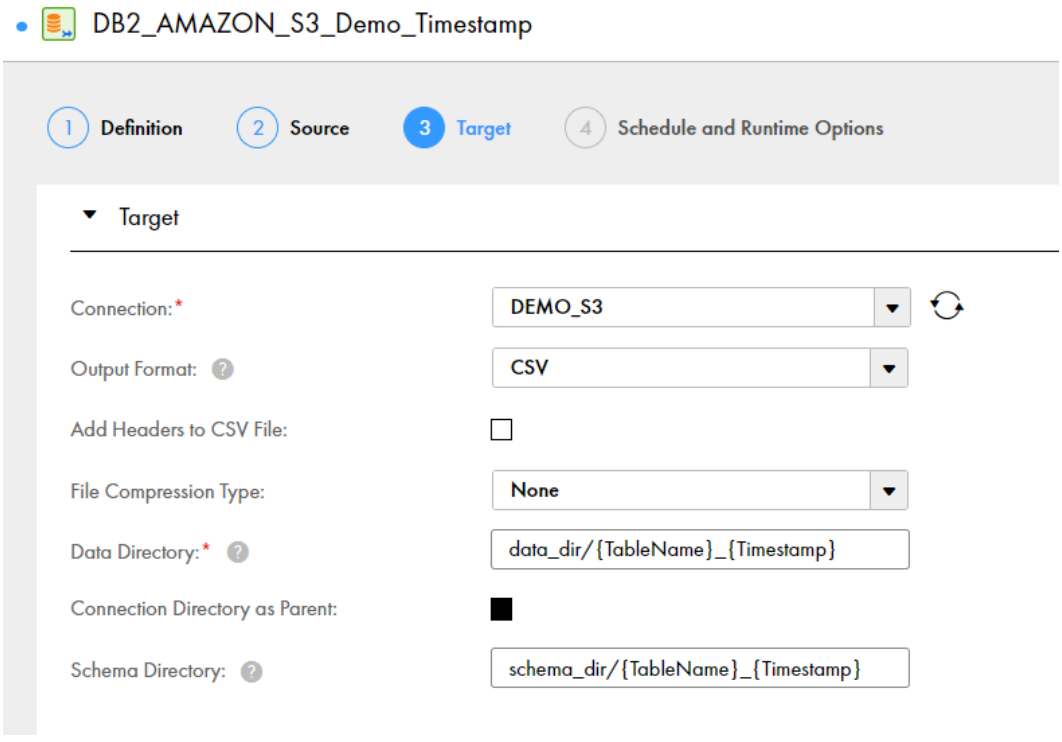


この設定に基づき、結果として得られるディレクトリ構造は次のようになります。

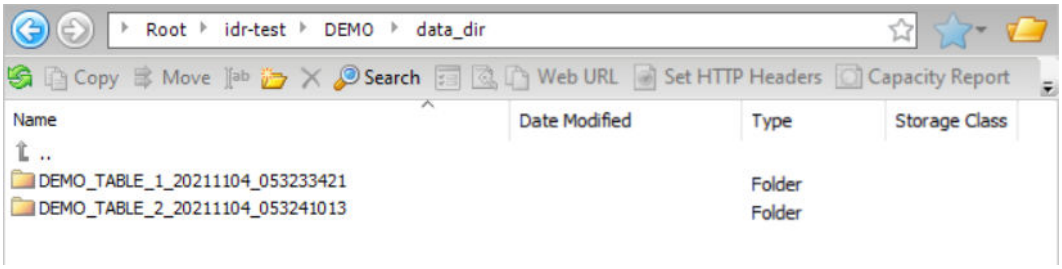


例 2

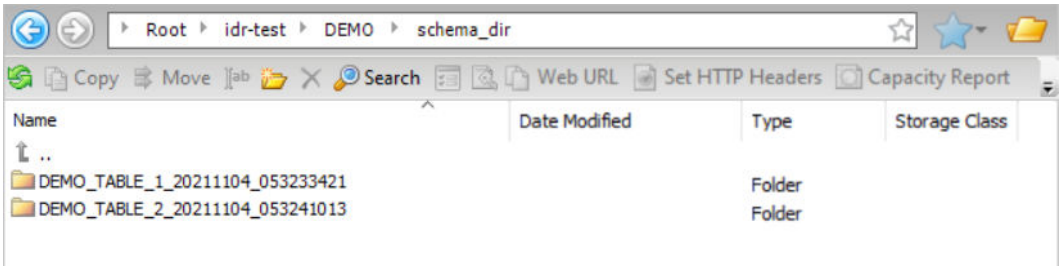
Amazon S3 ターゲットを使用していて、出力データファイルをカスタムディレクトリパスに書き込み、ターゲットスキーマを別のディレクトリパスに書き込みたいとします。Amazon S3 接続プロパティの **[フォルダパス]** フィールドで指定されたディレクトリをデータディレクトリとスキーマディレクトリの親ディレクトリとして使用するには、**[親としての接続ディレクトリ]** を選択します。この場合、親ディレクトリは idr-test/DEMO/です。**[データディレクトリ]** フィールドと **[スキーマディレクトリ]** フィールドで、data_dir や schema_dir などの特定のディレクトリ名を使用して、その後にデフォルトの{TableName}_{Timestamp}プレースホルダ値を指定することによって、ディレクトリパターンを定義します。プレースホルダは書き込み先の *tablename_timestamp* ディレクトリを作成します。次の画像は、タスクウィザードの **[ターゲット]** ページにあるこの設定を示しています。



この設定に基づき、結果として得られるデータディレクトリ構造は次のようになります。



結果として得られるスキーマディレクトリ構造は次のようになります。



増分ロード、および初期ロードと増分ロードの組み合わせ

デフォルトでは、増分ロードジョブ、および初期ロードジョブと増分ロードジョブの組み合わせは、サイクルファイルとデータファイルを親ディレクトリの下のサブディレクトリに書き込みます。ただし、カスタムディレクトリ構造を作成して、組織の要件に最も合うようにファイルを整理できます。

この機能は、Amazon S3、Google Cloud Storage、または Microsoft Azure Data Lake Storage（ADLS）Gen2 ターゲットを持つデータベース取り込み増分ロードタスク、および初期ロードタスクと増分ロードタスクの組み合わせに適用されます。また、Salesforce ソースとこれらのターゲットタイプのいずれかを持つアプリケーション取り込み増分ロードジョブにも適用されます。

Google Cloud Storage を除くすべてのターゲットについては、タスクウィザードの **[ターゲット]** ページで **[親としての接続ディレクトリ]** チェックボックスがオンになっている場合、親ディレクトリはターゲット接続プロパティで設定されます。

- Amazon S3 接続では、親ディレクトリは **[フォルダパス]** フィールドで指定されます。
- ADLS Gen2 接続では、この親ディレクトリは **[ディレクトリパス]** フィールドで指定されます。

Google Cloud Storage ターゲットでは、親ディレクトリは、タスクウィザードの **[ターゲット]** ページにある **[バケット]** フィールドで指定されたバケットコンテナです。

ニーズに合うようにディレクトリ構造をカスタマイズできます。例えば、データファイルとサイクルファイルは、接続プロパティで指定された親ディレクトリではなく、タスクのターゲットディレクトリに書き込むことができます。または、1）テーブル名を含むサブディレクトリの下にテーブル固有のデータとスキーマファイルを統合する、2）CDC サイクルごとにデータファイルとサマリコンテンツおよび完了したファイルをパーティション化する、または 3）リテラル値とプレースホルダを含むパターンを定義することによって完全にカスタマイズされたディレクトリ構造を作成することができます。例えば、SQL タイプの式を実行して時間に基づいてデータを処理する場合は、CDC サイクルごとにパーティション化せずに、すべてのデータファイルをタイムスタンプサブディレクトリに直接書き込むことができます。

増分ロードタスクのカスタムディレクトリ構造を設定するには、取り込みタスクウィザードの**［ターゲット］**ページで、次のオプションフィールドのいずれかのパターンを定義します。

フィールド	説明	デフォルト
タスクターゲットディレクトリ	増分ロードタスクの出力ファイルを保存するために使用するルートディレクトリの名前。 **［親としての接続ディレクトリ］**オプションを選択した場合でも、必要に応じてタスクターゲットディレクトリを指定できます。これは親ディレクトリに追加され、データ、スキーマ、サイクル完了、およびサイクルコンテンツディレクトリのルートを形成します。 このフィールドは、次のディレクトリフィールドのいずれかのパターンで{TaskTargetDirectory}プレースホルダが指定されている場合は必須です。	なし
親としての接続ディレクトリ	接続プロパティで指定された親ディレクトリを使用するには、このチェックボックスを選択します。	選択済み
データディレクトリ	データファイルを含むサブディレクトリへのパス。 データファイルとスキーマファイルが CDC サイクルごとにパーティション化されて*いない*場合、ディレクトリパスでは、{TableName}プレースホルダが必要です。	{TaskTargetDirectory}/data/{TableName}/data
スキーマディレクトリ	スキーマファイルをデータディレクトリに保存しない場合は、スキーマファイルを保存するサブディレクトリへのパス。 データファイルとスキーマファイルが CDC サイクルごとにパーティション化されていない場合、ディレクトリパスでは、{TableName}プレースホルダが必要です。	{TaskTargetDirectory}/data/{TableName}/schema
サイクル完了ディレクトリ	サイクル完了ファイルが含まれているディレクトリへのパス。	{TaskTargetDirectory}/cycle/completed
サイクルコンテンツディレクトリ	サイクルコンテンツファイルが含まれているディレクトリへのパス。	{TaskTargetDirectory}/cycle/contents
データディレクトリにサイクルのパーティション化を使用してください	各データディレクトリの下に、CDC サイクルごとにタイムスタンプサブディレクトリが作成されます。 このオプションが選択されていない場合、別のディレクトリ構造を定義しない限り、個々のデータファイルがタイムスタンプなしで同じディレクトリに書き込まれます。	選択済み

フィールド	説明	デフォルト
サマリディレクトリにサイクルのパーティション化を使用してください	サマリコンテンツサブディレクトリおよび完了サブディレクトリの下に、CDC サイクルごとにタイムスタンプサブディレクトリが作成されます。	選択済み
コンテンツ内の個々のファイルを一覧表示します	コンテンツサブディレクトリの下にある個々のデータファイルを一覧表示します。 **[サマリディレクトリにサイクルのパーティション化を使用する]** がオフの場合、このオプションはデフォルトで選択されています。タイムスタンプや日付などのプレースホルダを使用してカスタムサブディレクトリを設定できる場合を除き、コンテンツサブディレクトリ内の個々のファイルがすべて一覧表示されます。 **[データディレクトリにサイクルのパーティション化を使用する]** が選択されている場合でも、必要に応じてこのチェックボックスを選択して、個々のファイルを一覧表示し、CDC サイクルごとにグループ化できます。	**[サマリディレクトリにサイクルのパーティション化を使用する]** が選択されている場合は選択されません。 **[サマリディレクトリにサイクルのパーティション化を使用する]** をオフにした場合は選択されます。

ディレクトリパターンは、中括弧{ }で示される大文字と小文字を区別しないプレースホルダと特定のディレクトリ名の任意の組み合わせで構成されます。次のプレースホルダがサポートされています。

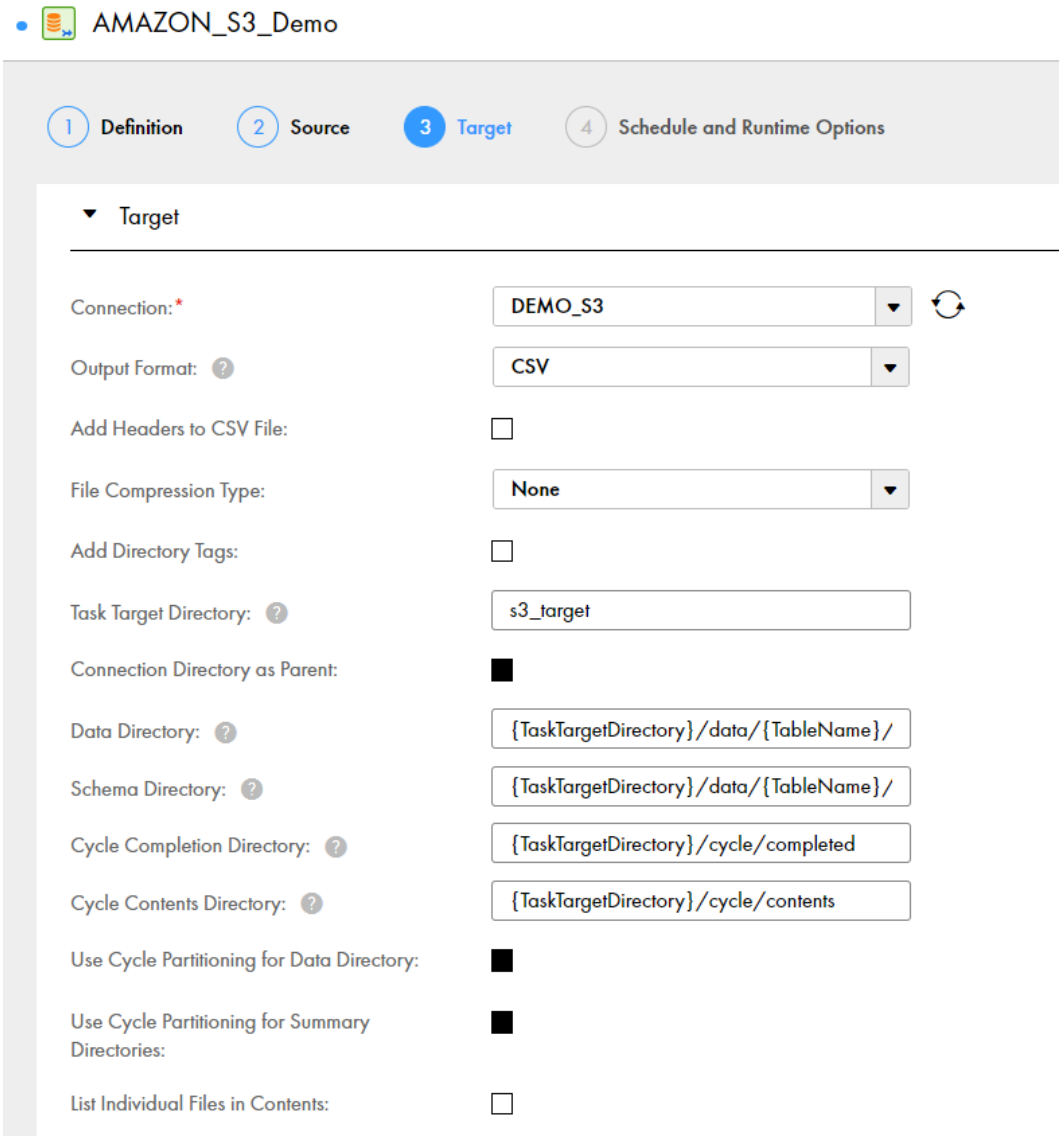
- 接続プロパティのディレクトリの代わりに使用するターゲット上のタスク固有のベースディレクトリを表す{TaskTargetDirectory}
- ターゲットテーブル名は{TableName}
- 日付と時刻を表す{Timestamp}、形式は yyyymmdd_hhmissms
- ターゲットスキーマ名は{Schema}
- 2 桁の年は{YY}
- 4 桁の年は{YYYY}
- 2 桁の月の値は{MM}
- 月の 2 桁の日は{DD}

注: タイムスタンプ、年、月、および日のプレースホルダは、データ、コンテンツ、および完了ディレクトリのパターンで指定した場合は CDC サイクルの開始時点、スキーマディレクトリパターンで指定した場合は CDC ジョブの開始時点を表します。

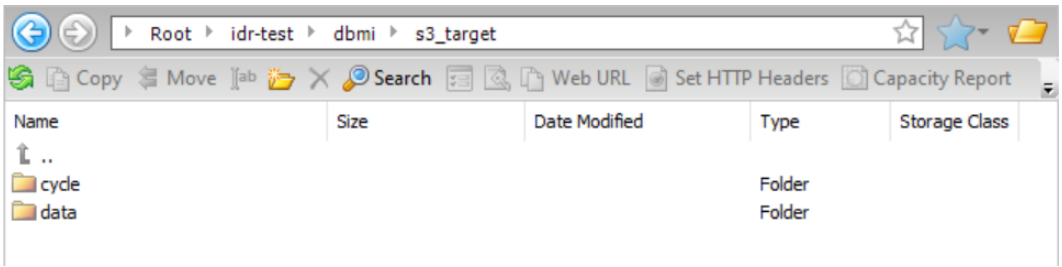
例 1:

タスクウィザードに表示される増分ロードジョブ、または初期ロードジョブと増分ロードジョブの組み合わせのデフォルトのディレクトリ設定を使用するつもりだとします。ターゲットタイプは Amazon S3 です。**[親としての接続ディレクトリ]** チェックボックスがデフォルトで選択されているため、Amazon S3 接続プロパティの **[フォルダパス]** フィールドで指定されている親ディレクトリパスが使用されます。この親ディレクトリは `idr-test/dbmi/` です。{TaskTargetDirectory}プレースホルダは、後続のディレクトリフィールドのデフォルトパターンで使用されるため、タスクターゲットディレクトリ名（この場合は s3_target）も指定する必要があります。{TableName}プレースホルダはデフォルトのパターンに含まれているため、データディレクトリとスキーマディレクトリ内のファイルはテーブル名でグループ化されます。また、サイクルのパーティション化が有効になっているため、データディレクトリ、スキーマディレクトリ、およびサイクルサマリディレクトリ内

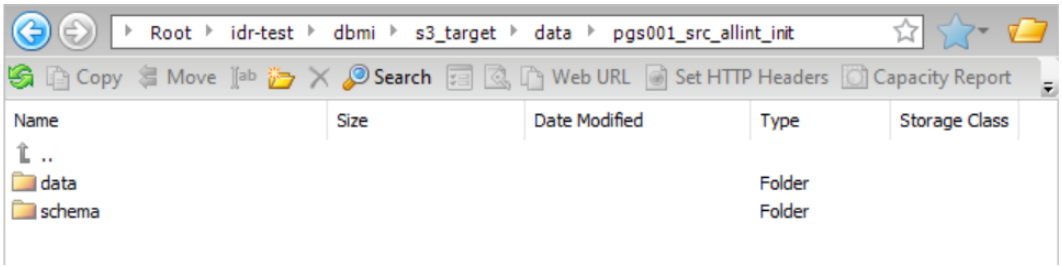
のファイルは、CDC サイクルごとにさらに分割されます。次の画像は、指定されたタスクターゲットディレクトリ名を除く、タスクウィザードの［**ターゲット**］ページのデフォルトの設定を示しています。



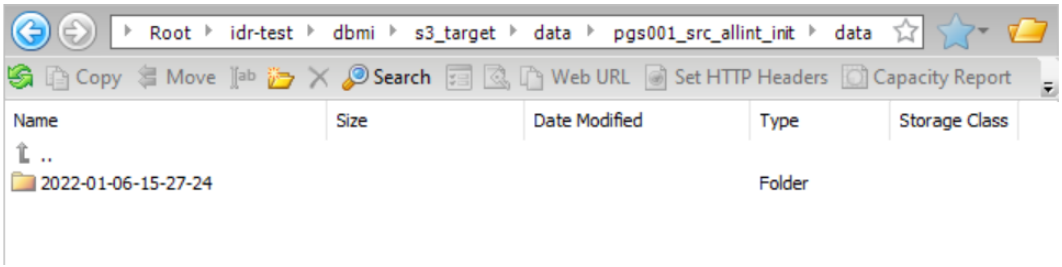
この設定に基づき、結果として得られるデータディレクトリ構造は次のようになります。



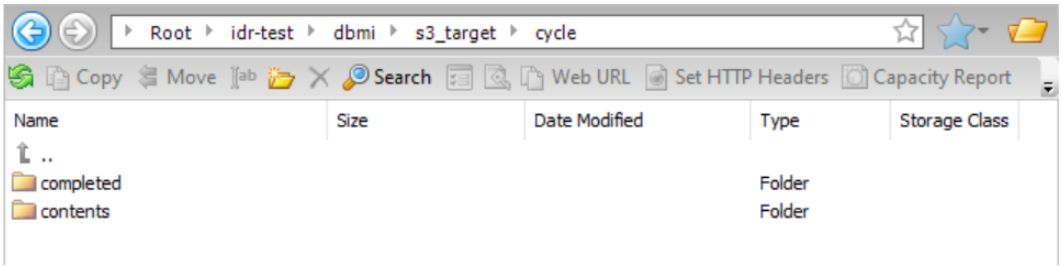
データフォルダをドリルダウンしてから、そのフォルダ内のテーブル（pgs001_src_allint_init）をドリルダウンすると、データサブディレクトリとスキーマサブディレクトリにアクセスできます。



データフォルダをドリルダウンすると、データファイルのタイムスタンプディレクトリにアクセスできます。



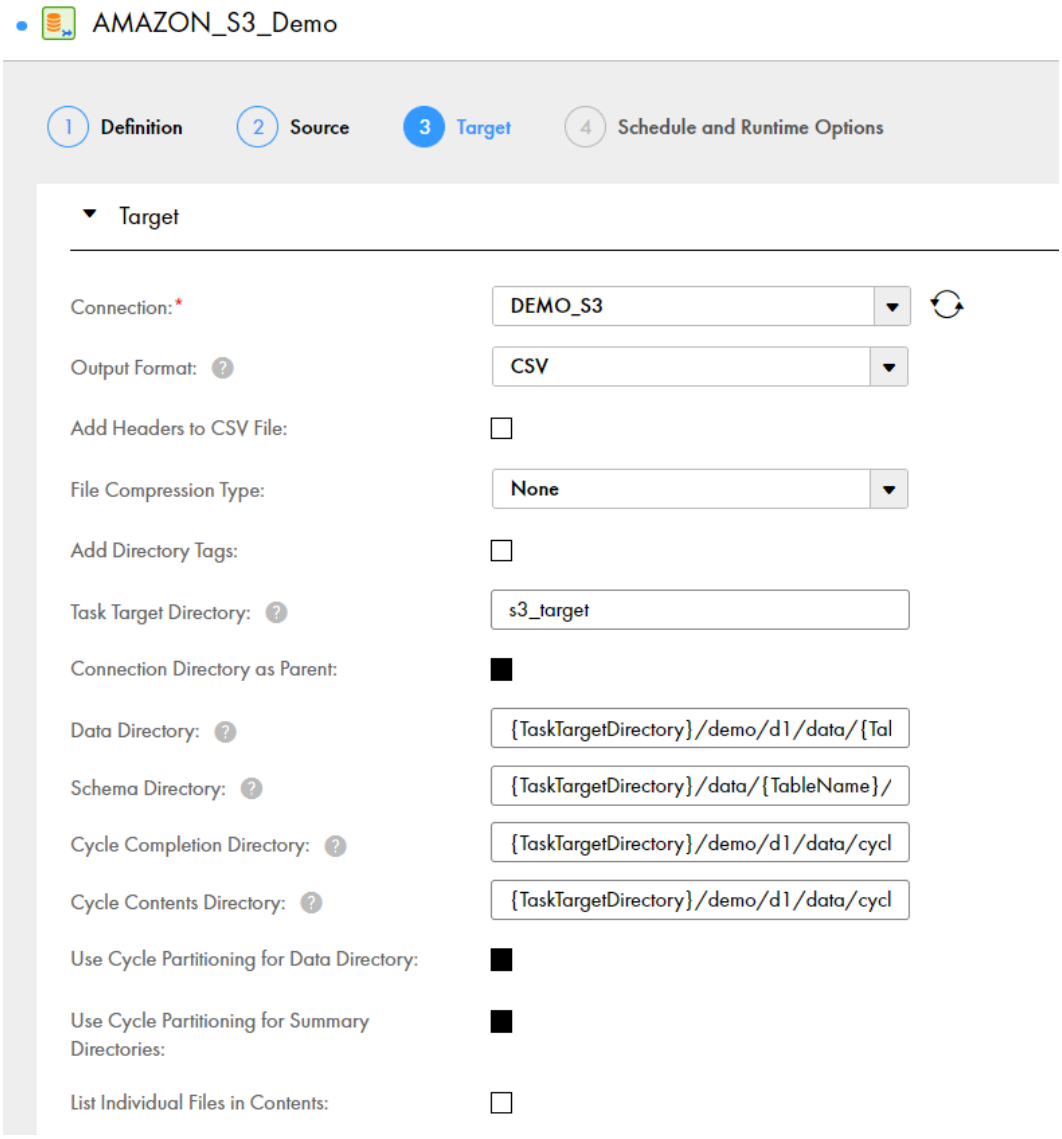
サイクルをドリルダウンすると、サマリコンテンツサブディレクトリと完了サブディレクトリにアクセスできます。



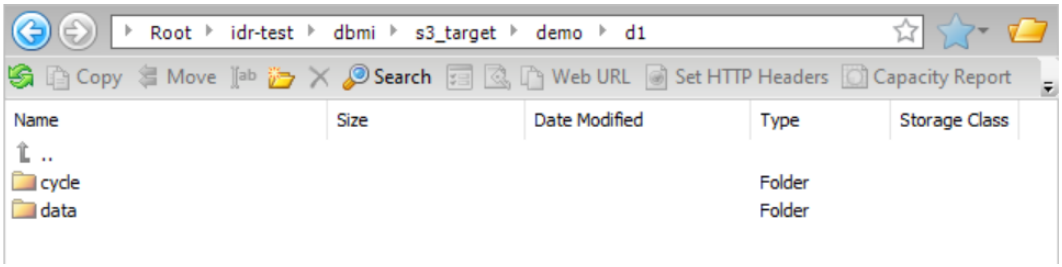
例 2

スキーマディレクトリを除くすべてのディレクトリパスにサブディレクトリ「demo」と「d1」を追加する増分ロードジョブ、または初期ロードジョブと増分ロードジョブの組み合わせ用のカスタムディレクトリ構造を作成して、デモ用のファイルを簡単に見つけられるようにしたいと考えています。**［親としての接続ディレクトリ］**チェックボックスが選択されているため、Amazon S3 接続プロパティの**［フォルダパス］**フィールドで指定されている親ディレクトリパス（idr-test/dbmi/）が使用されます。{TaskTargetDirectory}プレースホルダは、後続のディレクトリフィールドのパターンで使用されるため、タスクターゲットディレクトリも指定する必要があります。データディレクトリとスキーマディレクトリ内のファイルは、テーブル名でグループ化されます。また、サイクルのパーティション化が有効になっているため、データディレクトリ、スキーマディレク

トリ、およびサイクルサマリディレクトリ内のファイルは、CDC サイクルごとにさらに分割されます。次の画像は、タスクウィザードの［**ターゲット**］ページにあるカスタム設定を示しています。



この設定に基づき、結果として得られるデータディレクトリ構造は次のようになります。



サポートされている Avro データ型

一括取り込みデータベースは、Avro スキーマが提供するプリミティブデータ型と論理データ型の一部をサポートします。これらのデータ型は、Avro または Parquet 出力形式をサポートするターゲット型に関連しています。

プリミティブデータ型は、単一のデータ値を表します。論理データ型は、派生型を表す追加の属性を持つ Avro プリミティブデータ型または複合データ型です。

次の表に、一括取り込みデータベースがサポートするプリミティブ Avro データ型を示します。

プリミティブデータ型	説明
INT	32 ビット符号付き整数
LONG	64 ビット符号付き整数
FLOAT	単精度（32 ビット）IEEE 754 浮動小数点数
DOUBLE	倍精度（64 ビット）IEEE 754 浮動小数点数
BYTES	8 ビットの符号なしバイトのシーケンス
STRING	Unicode 文字シーケンス

次の表に、一括取り込みデータベースがサポートする論理 Avro データ型を示します。

論理データ型	説明
DECIMAL	スケーリングされていない形式の任意精度の符号付き 10 進数×10 *-scale*
DATE	時刻またはタイムゾーンへの参照を含まない日付。
TIME	タイムゾーンまたは日付への参照を含まない、1 ミリ秒または 1 マイクロ秒の精度の時刻。
TIMESTAMP	特定のカレンダーまたはタイムゾーンへの参照を含まない、1 ミリ秒または 1 マイクロ秒の精度の日時値。

Databricks Delta ターゲットの場合、一括取り込みデータベースは中間 Parquet ファイルで次のデータ型を使用しません。

- ミリ秒の精度の TIMESTAMP
- ミリ秒またはマイクロ秒の精度の TIME

スキーマドリフト処理

一括取り込みデータベースは、ソーススキーマの変更を自動的に検出し、ターゲットに対してこれらの変更を処理するように設定できます。このプロセスは、スキーマドリフトと呼ばれます。

一括取り込みデータベースは次のタイプのソーススキーマの変更を検出できます。

- カラムの追加

- カラムの変更
- カラムの削除
- カラム名の変更

タスクを定義するときに、データベース取り込みタスクウィザードの**［スケジュールおよびランタイムオプション］**ページで、サポートされているタイプのスキーマ変更の処理方法を設定できます。例えば、スキーマドリフトオプションを設定して、変更を無視したり、変更をレプリケートしたり、スキーマの変更が発生したときにジョブまたはサブタスクを停止したりできます。詳細については、「スケジュールとランタイムオプションの設定」 (ページ 157)を参照してください。スキーマ変更のタイプが異なると、ターゲットのタイプに応じてデフォルト設定が異なる場合があります。

スキーマドリフトオプションは、次のソースとターゲットの組み合わせおよびロードタイプでサポートされています。

ソース	ロードタイプ	ターゲット
Db2 for i	差分 初期と増分の組み合わせ	Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Kafka（増分ロードのみ）、Microsoft Azure Data Lake Storage、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、または Snowflake
DB2 for z/OS（Db2 11 を除く）	差分 初期と増分の組み合わせ	Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Kafka（増分ロードのみ）、Microsoft Azure Data Lake Storage、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、または Snowflake
Microsoft SQL Server	差分 初期と増分の組み合わせ	Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Kafka（増分ロードのみ）、Microsoft Azure Data Lake Storage、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、または Snowflake
Oracle	差分 初期と増分の組み合わせ	Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Kafka（増分ロードのみ）、Microsoft Azure Data Lake Storage、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、または Snowflake
PostgreSQL	差分 初期と増分の組み合わせ	**増分ロード**: Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Kafka（増分ロードのみ）、Microsoft Azure Data Lake Storage、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、または Snowflake **初期と増分の組み合わせ**: Oracle または Snowflake

考慮事項:

- ターゲットでサポートされていないタイプのスキーマ変更をレプリケートしようとすると、データベース取り込みジョブはエラーが発生して終了します。
- 一括取り込みデータベースは、プライマリキーまたは一意キーの制約を追加、削除、または変更するソースの変更をレプリケートしません。ソースでこれらのタイプの変更が発生した場合は、ターゲットテーブルを再同期する必要があります。

- 一括取り込みデータベースがスキーマの変更を検出したときにジョブを停止するスキーマドリフトオプションを設定している場合は、**［オプションを指定して再開］**コマンドを使用して、オーバーライドスキーマドリフトオプションを指定してジョブを再開することができます。
- 一括取り込みデータベースがソーステーブルのスキーマ変更を検出するのは、変更されたソーステーブルで DML 操作が発生した後のみです。DML 操作に干渉せずに複数のスキーマ変更が発生した場合、一括取り込みデータベースは DML 操作が発生すると、スキーマのすべての変更を一度に検出します。一括取り込みデータベースがサポートされているすべてのスキーマ変更を正しく検出するようにするために、Informatica では、スキーマ変更をソーステーブルに 1 つずつ適用し、その後に少なくとも 1 つの DML 変更を適用することをお勧めします。
- Oracle または SQL Server ソースを持つタスクの［カラムの追加］オプションを［レプリケート］に設定した場合、一括取り込みデータベースは、テーブル定義の最後に新しいカラムが追加されると想定します。テーブル定義の途中に新しいカラムがある場合、一括取り込みデータベースは、カラムが削除されて再度追加されたかのように変更を処理する場合があります。この状況では、次の警告によって、同じカラムが削除されて追加されたと報告されることがあります。

  `Column <column_name2> has been added. Column <column_name2> has been dropped.`
- Microsoft Azure Synapse Analytics ターゲットを持つデータベース取り込みタスクは、ソースカラムの名前変更操作をレプリケートできません。**［レプリケート］**オプションは使用できません。
- Snowflake ターゲットを持つデータベース取り込みタスクは、次の制限付きでソースカラムの変更操作をサポートします。

  - Snowflake ターゲットは、NUMBER カラムのスケールを変更できません。

  - Snowflake ターゲットは、既存のカラムのデータ型を別のデータ型に変更することはサポートしていません。
- Google BigQuery ターゲットを持つデータベース取り込みタスクは、ソースカラムの名前変更または変更操作をレプリケートできません。これらの操作のスキーマドリフトオプションは使用できません。
- **［カラムの追加］**オプションを **［レプリケート］**に設定してから、デフォルト値を持つカラムを DB2 for i ソーステーブルに追加すると、一括取り込みデータベースは、ターゲットに新たに追加されたテーブル行にデフォルト値を追加します。ただし、ターゲットの既存の行は更新されず、デフォルト値が反映されません。既存のターゲット行に移入されたデフォルト値を取得するには、別の初期ロードを実行してターゲットを再実体化します。

すべてのソーステーブル DML 変更の監査履歴のターゲットテーブルへの適用

ソーステーブルに対して行われたすべての DML 変更操作の監査証跡をターゲットに書き込むために、Google BigQuery、Oracle、または Snowflake ターゲットを持つデータベース取り込み増分ロードタスク、および初期ロードタスクと増分ロードタスクの組み合わせを設定できます。ソーステーブルの各 DML 変更の行が、選択した監査カラムとともに、生成されたターゲットテーブルに書き込まれます。監査カラムには、DML 操作タイプ、時刻、所有者、トランザクション ID、生成された昇順シーケンス番号、前のイメージなどの変更に関するメタデータが含まれています。

タスクを定義するときに、**［ターゲット］**ページの **［適用モード］**フィールドで **［監査］**を選択します。**［適用モード］**フィールドは、新しいタスクまたはデプロイされていないタスクで使用できます。

追加する監査メタデータカラムを指定するには、**［ターゲット］**ページの**［詳細］**の下にある次のチェックボックスを 1 つ以上選択します:

- **最終レプリケート時刻を追加**。ターゲットテーブルでレコードが挿入または最後に更新された時点のタイムスタンプを記録するカラムを追加します。初期ロードの場合、ロードされたすべてのレコードのタイムスタンプは同じですが、Superpipe オプションを使用する Snowflake ターゲットに限り、秒または分がわずかに異なる可能性があります。増分ロード、および初期ロードと増分ロードの組み合わせの場合、このカラムには、ターゲットに適用された最後の DML 操作のタイムスタンプが記録されます。

  注: このオプションは、Google BigQuery および Snowflake ターゲットでのみ使用できます。
- **操作の追加 <*metadata_type*>**。DML 操作タイプ、時刻、所有者、トランザクション ID、生成された昇順シーケンス番号など、変更操作のメタデータを含むカラムが追加されます。カラムには、データがターゲットテーブルにロードされるときに入力されます。
- **前のイメージを追加**。更新の操作前のイメージデータを含む_OLD カラムが追加されます。テーブルの以前のカラムの値と新しいカラムの値を比較できます。

  注: このオプションは Oracle ターゲットでは使用できません。
- **メタデータカラムのプレフィックス**。追加された監査カラムの名前にプレフィックスを追加して、他のテーブルカラムと区別できるようにします。デフォルトは INFA_です。

これらのフィールドはオプションです。**［操作タイプの追加］**チェックボックスのみデフォルトで選択されており、次の DML 操作タイプを示すカラムが追加されます: I（挿入）、D（削除）、または U（更新）。

タスクに関連付けられたジョブを初めて実行すると、ジョブは、選択した監査メタデータカラムを含むターゲットテーブルを生成します。ターゲットテーブルにインデックス以外の制約が存在しないことを確認します。

Databricks Delta ターゲットや Snowflake ターゲットなど、**［論理削除］**モードをサポートするターゲットの場合、論理削除として処理された各削除操作に対して、操作タイプ「D」が INFA_OPERATION_TYPE カラムに表示され、選択した他のメタデータカラムに値が書き込まれます。ただし、更新操作と挿入操作では、INFA_OPERATION_TYPE カラムと選択されている他のすべてのメタデータカラムは NULL です。

監査モードの例

たとえば、ソーステーブルに対して次の DML 変更操作が次の順序で発生するとします。

```
Insert into tableA pkey = 1
Update tableA where pkey=1
Update tableA where pkey=1
Delete from tableA where pkey = 1
```

次のすべての行がターゲットテーブルに表示され、すべてのソース DML 変更の監査証跡が提供されます。

```
opType=I, pkey=1....
opType=U, pkey=1...
opType=U, pkey=1...
opType=D, pkey=1...
```

この例では、選択されている監査カラムは opType のみです。

タスクの適用モードが［標準］の場合、最後の DML 操作は削除であり、以前の変更が上書きされるため、これらの行はターゲットテーブルに表示されません。

注: 初期ロードと増分ロードの組み合わせジョブで、初期アンロードフェーズ中に増分挿入変更レコードをキャプチャした場合、ジョブは同じ行に対して削除を実行して、初期アンロードで取得された可能性のある重複を削除します。この意図的に作成されたアクティビティは、監査適用モードで反映されます。

ターゲットに対する削除を論理削除として適用する機能

サポートされているソースタイプと Databricks Delta および Snowflake ターゲットを持つデータベース取り込み増分ロードジョブおよび初期ロードと増分ロードの組み合わせジョブの場合、ターゲットでソースの削除操作を論理削除として処理するようにタスクを設定できます。

論理削除では、削除された行をデータベースから実際には削除せずに、削除済みとしてマークします。この行は、生成された INFA_OPERATION_TYPE メタデータカラムに「D」という値を設定してターゲットに適用されます。

重要: 論理削除は、すべてのソースオブジェクトにプライマリキーがあり、行が最初に作成された後は行のプライマリキー値を変更することがソースで許可されていない場合にのみ使用できます。プライマリキーの値が変更されると、重複した行がターゲットに書き込まれ、ターゲットが破損する可能性があります。

シナリオ例: 組織は、データウェアハウスで論理削除を使用して、監査目的で行を保持しながら、ソースで削除された行をマークする必要があるとします。

論理削除を有効にするには、取り込みタスクを設定するときに、タスクウィザードの **[ターゲット]** ページの **[適用モード]** フィールドを **[論理削除]** に設定します。

注: 初期ロードと増分ロードの組み合わせジョブで、初期アンロードフェーズ中に増分挿入変更レコードをキャプチャした場合、ジョブは同じ行に対して削除を実行して、初期アンロードで取得された可能性のある重複を削除します。この意図的に作成されたアクティビティは、論理削除適用モードで反映されます。

データベース統合タスクの設定

一括取り込みで、データベース統合タスクウィザードを使用してデータベース統合タスクを設定します。

ウィザードページで、次の設定タスクを完了します。

1. タスク名、プロジェクトの場所、ランタイム環境、ロードタイプなどの基本的なタスク情報を定義します。
2. ソースを設定します。
3. ターゲットを設定します。
4. ランタイムオプションを設定します。

[次へ] または **[戻る]** をクリックして別のページに移動します。いつでも **[保存]** をクリックして、これまでに入力した情報を保存できます。

すべてのウィザードページを完了したら、情報を保存し、**[デプロイ]** をクリックして、タスクを実行可能ジョブとして Secure Agent で使用できるようにします。

始める前に

開始する前に、Administrator で次の要件タスクを完了してください。

- 組織に一括取り込みデータベースおよび DBMI パッケージのライセンスがあることを確認してください。
- ランタイム環境の Secure Agent が実行されていること、および一括取り込みサービスにアクセスできることを確認します。
- ソース接続およびターゲット接続を定義します。

また、Oracle ソースを使用して増分ロード操作を実行する場合は、Secure Agent システムで ORACLE_HOME 環境変数が定義されていることを確認してください。

基本的なタスク情報の定義

データベース統合タスクの定義を開始するには、最初に、タスク名、プロジェクトまたはプロジェクトフォルダの場所、ロード操作の種類など、タスクに関するいくつかの基本情報を入力する必要があります。

1. 一括取り込み［一括取り込み］で、**［新規］** > **［データベース取り込みタスク］**をクリックします。

一括取り込みデータベースタスクウィザードの**［定義］**ページが表示されます。

2. 以下のプロパティを設定します。

プロパティ	説明
名前	データベース統合タスクの名前。 タスク名には、ラテン英数字、スペース、ピリオド（.）、コンマ（,）、アンダースコア（_）、プラス記号（+）、およびハイフン（-）を含めることができます。タスク名に他の特殊文字を含めることはできません。 タスク名では大文字と小文字は区別されません。 最大長は 50 文字です。 **注**: データベース取り込みタスク名にスペースを含めると、タスクをデプロイした後、対応するジョブ名にスペースが表示されなくなります。
場所	タスク定義を含むプロジェクトまたはプロジェクト\フォルダ。デフォルトは、Explore で現在選択されているプロジェクトまたはプロジェクトのサブフォルダです。プロジェクトまたはプロジェクトのサブフォルダが選択されていない場合、デフォルトは**［デフォルト］**プロジェクトになります。
ランタイム環境	タスクを実行するランタイム環境。 ランタイム環境は、1 つ以上の Secure Agent で構成される Secure Agent グループである必要があります。Secure Agent は、タスクを実行し、安全な通信を可能にする軽量のプログラムです。 Hosted Agent またはサーバーレスランタイム環境を使用することはできません。 **ヒント**: **［更新］**アイコンをクリックして、ランタイム環境のリストを更新します。

プロパティ	説明
説明	タスクのオプションの説明。 最大長は 4,000 文字です。
ロードタイプ	データベース統合タスクを実行するロード操作のタイプ。次のオプションがあります。 - **初期ロード**。特定の時点で読み取られたデータを、バッチ操作でソーステーブルからターゲットにロードします。初期ロードを実行して、増分変更データの送信先となるターゲットをマテリアライズできます。 - **増分ロード**。ソースデータの変更を継続的に、またはジョブが停止または終了するまでターゲットにプロパゲートします。ジョブは、ジョブが最後に実行されてから、または最初のジョブ実行の特定の開始点から発生した変更をプロパゲートします。 - **初期ロードと増分ロード**。ターゲットへのポイントインタイムデータの初期ロードを実行してから、同じソーステーブルに対して継続的に行われた増分データ変更のプロパゲートに自動的に切り替わります。 **注**: 初期アンロードロードフェーズ中に変更レコードがキャプチャされた場合、その変更レコードはアンロードフェーズが完了するまで適用処理の対象から外されます。アンロードフェーズ中にキャプチャされた挿入行は、削除操作と挿入操作のペアに変換され、アンロードされたデータとキャプチャされた変更データの両方で挿入が発生した場合は、1 つの挿入行のみがターゲットに適用されるようになります。

3. **［次へ］**をクリックします。

ソースの設定

データベース統合タスクウィザードの**［ソース］**ページでソースを設定します。

注: MongoDB ソースの場合のみ、タスクウィザードはスキーマの代わりにデータベースを表示し、テーブルの代わりにコレクションを表示します。ただし、このドキュメントでは、単純化するため、すべてのソースタイプを網羅するようにスキーマとテーブルという用語を使用しています。

1. **［接続］**リストで、ソースシステムの接続を選択します。接続タイプは、接続名の後の括弧内に表示されます。

   組織が使用するランタイム環境の接続は、管理者で事前定義する必要があります。

   リストには、**［定義］**ページで選択されたロードタイプに有効な接続タイプのみが含まれます。ロードタイプを選択しなかった場合、接続は一覧表示されません。

   ロードタイプを変更し、選択した接続が無効になると、警告メッセージが発行され、**［接続］**フィールドがクリアされます。更新されたロードタイプに有効な別の接続を選択する必要があります。

   注: 取り込みタスクをデプロイした後は、最初に関連する取り込みジョブをデプロイ解除せずに接続を変更することはできません。接続を変更した後、タスクを再度デプロイする必要があります。

2. **［スキーマ］**リストで、ソーステーブルを含むソーススキーマを選択します。

   リストには、指定されたソース接続でアクセスされるデータベースで使用可能なスキーマのみが含まれます。
   Oracle、Microsoft SQL Server、Netezza、または PostgreSQL ソースを持つタスクを作成する場合、接続プロパティで指定されたスキーマ名がデフォルトで表示されます。

3. 増分ロードタスク用に DB2 for i ソースを定義する場合は、**［ジャーナル名］**フィールドで、ソーステーブルに加えられた変更を記録するジャーナルの名前を選択します。

4. 増分ロードタスク、または初期ロードと増分ロードの組み合わせタスクの PostgreSQL ソースを定義する場合は、次のフィールドに入力します。

フィールド	説明
レプリケーションスロット名	PostgreSQL レプリケーションスロットの一意の名前。 スロット名には、小文字のラテン英数字とアンダースコア（_）文字を含めることができます。 最大長は 63 文字です。 **重要**: 各データベース取り込みタスクは、異なるレプリケーションスロットを使用する必要があります。
レプリケーションプラグイン	PostgreSQL レプリケーションプラグイン。次のオプションがあります。 - pgoutput。このオプションは、PostgreSQL バージョン 10 以降でのみ選択できます。 - wal2json
パブリケーション	レプリケーションプラグインとして pgoutput を選択した場合は、このプラグインが使用するパブリケーション名を指定します。 **注**: レプリケーションプラグインとして wal2json を選択した場合、このフィールドは表示されません。

5. Db2 for LUW、Oracle、または SQL Server ソースを持つ増分ロードタスク、または初期ロードと増分ロードの組み合わせタスクを定義する場合は、**［変更データキャプチャメソッド］**で使用するキャプチャメソッドを選択します。
   a. **［CDC メソッド］**フィールドで、次のオプションのいずれかを選択して、ソースの変更をキャプチャするために使用するメソッドを指定します。

方法	サポートされるソース	説明
CDC テーブル	SQL Server のみ	SQL Server CDC テーブルからデータ変更を直接読み取ります。
ログベース	Oracle および SQL Server	データベーストランザクションログを読み取ることによって、挿入、更新、削除、およびカラムの DDL 変更を近似リアルタイムでキャプチャします。 Oracle ソースの場合、データ変更は Oracle REDO ログから読み取られます。 SQL Server ソースの場合、データ変更は SQL Server トランザクションログと有効な SQL Server CDC テーブルから読み取られます。
クエリベース	Db2 for LUW、Oracle、および SQL Server	CDC クエリカラムを指す SQL WHERE 句を使用して、挿入と更新をキャプチャします。クエリカラムは、CDC 間隔の開始以降にソーステーブルに加えられた変更を含む行の識別に使用されます。 増分ロードジョブおよび初期ロードと増分ロードの組み合わせジョブの Db2 for LUW ソースの場合、このキャプチャメソッドのみ使用できます。

b. **［クエリベース］**オプションを選択した場合は、次の追加フィールドに入力します。

- **CDC クエリカラムタイプ**。ソーステーブルにおける CDC クエリカラムのカラムタイプです。利用可能な唯一のオプションは**［タイムスタンプ］**です。

  注:「タイムスタンプ」は、日付と時刻を組み合わせたカラムデータ型を表します。Oracle の場合、クエリカラムでサポートされるデータ型は TIMESTAMP です。SQL Server の場合、クエリカラムでサポートされるデータ型は DATETIME と DATETIME2 です。

- **CDC クエリカラム名**。ソーステーブルにおける CDC クエリカラムの、大文字小文字が区別された名前です。カラムはソーステーブルに存在する必要があります。最大長は 70 文字です。
- **CDC 間隔**。日数、時間数、分数で表される、クエリベースの変更データキャプチャサイクルの頻度です。少なくとも 1 つの間隔フィールドに正の数値を入力する必要があります。入力しない場合、タスクを保存しようとすると、エラーが発生します。デフォルト値は 5 分です。

6. **［テーブルの選択］**で、以下のいずれかの方法を使用してソーステーブルを選択します。

- データレプリケーション用のスキーマ内のすべてのテーブルとカラムを選択するには、**［すべて選択］**を選択します。**［選択されたテーブル］**フィールドには、選択されたすべてのテーブルの数が表示されます。**［テーブルビュー］**での選択内容を後で編集することはできません。

  注: すべてのテーブルの情報を取得するには、時間がかかる場合があります。

- **［ルール］**で、レプリケートするソーステーブルのサブセットを指定するルールを定義します。ルールを使用すると、**［テーブルビュー］**でテーブルや、選択したテーブルのカラムを個別に選択または選択解除できます。また、文字データのスペースを切り捨てるオプションを設定することもできます。ルールを追加するには、8 に進みます。

7. データベースビューをソースとして含める場合は、［更新］アイコンの右側にある**［ビューを含める］**チェックボックスを選択します。このチェックボックスは、Db2 for i、Db2 for LUW、Microsoft SQL Server、MySQL、Oracle、PostgreSQL、または Teradata ソースを持つ初期ロードタスクでのみ使用できます。

   ビューが取得され、**［選択されたテーブル］**のカウント、およびテーブル名のリストに含まれます。

8. テーブル選択ルールを追加するには、まず**［すべて選択］**チェックボックスがオフになっていることを確認します。次に、以下のサブステップを実行します。

   a. **［ルール］**で、最初のテーブルの上にある［ルールの追加］（+）アイコンをクリックします。テーブルに行が追加されます。

   b. **［テーブルルール］**カラムで**［含める］**または**［除外する］**を選択して、包含ルールまたは除外ルールを作成します。

   c. **［条件］**カラムに、テーブル名、または 1 つ以上のワイルドカードを含むテーブル名マスクを入力して、テーブル選択に含める、またはテーブル選択から除外するソーステーブルを特定します。次のガイドラインを使用します。

   - マスクには、次のワイルドカードの 1 つまたは両方を含めることができます: 1 つ以上の文字を表すアスタリスク（*）ワイルドカードと単一の文字を表す疑問符（?）ワイルドカード。ワイルドカードは、マスク値内で複数回使用することができ、値内のどこでも使用できます。
   - タスクウィザードでは大文字と小文字が区別されます。テーブルが定義された際の指定どおりに大文字小文字を区別してテーブル名またはマスクを入力します。
   - ソースデータベースで使用されている場合でも、引用符や括弧などの区切り文字は含めないでください。例えば、Oracle では、名前を小文字または大文字と小文字の混在で格納されるように、小文字の名前または大文字と小文字が混在する名前を引用符で囲む必要があります。ただし、タ

スクウィザードでは、小文字または大文字と小文字が混在する名前を引用符なしで入力する必要があります。

- テーブル名にバックスラッシュ（\）、アスタリスク（*）、ドル記号（$）、キャレット（^）、疑問符（?）などの特殊文字が含まれている場合は、ルールを入力するときに名前の各特殊文字をバックスラッシュ（\）でエスケープします。

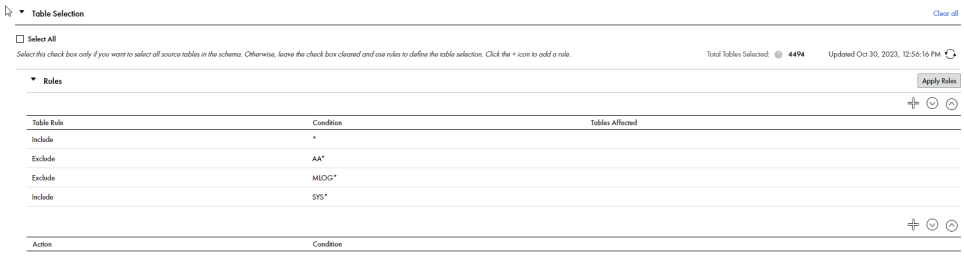
d. 必要に応じて追加のルールを定義します。

含めるルールと除外ルールを複数定義すると、一覧表示されている順序で上から下に処理されます。矢印アイコンを使用して順序を変更します。複数のルールを使用する例については、「「ソーステーブルを選択するルールの例」 (ページ 115)」を参照してください。

e. 終了したら、**［ルールの適用］**をクリックします。

［選択されたテーブルの総数］と**［テーブルビュー］**の数が更新されます。［更新］アイコンをクリックすると、各ルールの**［影響を受けるテーブル］**の数が表示されます。

次の図は、**［ソース］**ページで定義した複数のルールを示しています。



ルールの適用後にルールを追加、削除、または変更する場合は、**［ルールの適用］**を再度クリックする必要があります。［更新］アイコンをクリックして、テーブル数を更新します。**［ルールの適用］**をクリックせずにすべてのルールを削除した場合、デプロイ時に検証エラーが発生します。**［テーブルビュー］**リストには引き続きテーブルが表示されます。**［すべて選択］**に切り替えると、ルールは無効になり、表示されなくなります。

9. ルールに基づいて選択されたソーステーブルの文字カラムに対して切り捨てアクションを実行するには、**［ルール］**にある 2 番目の［アクション］テーブルでカラムアクションルールを作成します。

注: MongoDB ソースに対してカラムアクションルールを作成することはできません。

a. 2 番目のテーブルの上にある［ルールの追加］（+）アイコンをクリックします。

b. **［アクション］**カラムで、次のいずれかのオプションを選択します。

- **LTRIM**.文字カラム値の左側のスペースを切り捨てます。
- **RTRIM**.文字カラム値の右側のスペースを切り捨てます。
- **TRIM**.文字カラム値の左側と右側のスペースを切り捨てます。

c. **［条件］**カラムに、カラム名または 1 つ以上のアスタリスク（*）または疑問符（?）を含むカラム名マスクを入力します。ワイルドカード。値は、アクションが適用されるカラムを識別するために、選択したソーステーブルのカラムと照合されます。

注: 異なるアクションタイプに対して、あるいは条件が異なる同じアクションタイプに対して複数のルールを定義できます。ルールは、リストされている順序で上から下に処理されます。矢印アイコンを使用して順序を変更します。

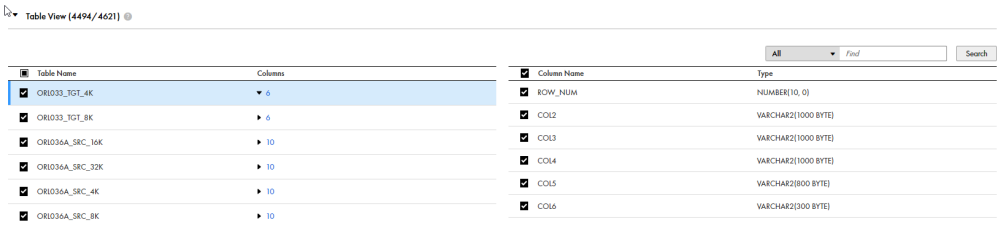
10. **［テーブルビュー］**で、選択したソーステーブルとカラムのセットを表示または編集します。

［すべて選択］を選択した場合、テーブルとカラムのリストは表示専用です。

ルールを適用した場合は、個々のテーブルの横にあるチェックボックスをクリックして、選択したテーブルのセットを絞り込むことができます。レプリケートしないテーブルの選択を解除するか、レプリケートする追加の項目を選択します。［更新］アイコンをクリックして、選択したテーブルの数を更新します。

Oracle および SQL Server ソースの場合のみ、選択したソーステーブルのカラムを個別に選択解除または再選択することもできます。選択したテーブルのデータのレプリケート元のカラムを表示または変更するには、**[カラム]** カラムにある、強調表示されたカラム数をクリックします。カラム名とデータ型が右側に表示されます。デフォルトでは、選択したソーステーブルのすべてカラムが選択されています。カラムを選択解除または再選択するには、カラム名の横にあるチェックボックスをクリックします。プライマリキーカラムを選択解除することはできません。

次の図は、選択したテーブルと、最初のテーブルで選択したカラムを示しています。



注:

- テーブルやカラムを検索するには、［カラム］リストの上のドロップダウンリストで **[テーブル名]**、**[カラム]**、または **[すべて]** を選択し、［検索］ボックスに検索文字列を入力して **[検索]** をクリックします。文字列の先頭または末尾にアスタリスク（*）ワイルドカードを 1 つ含めることができます。
- テーブルまたはカラムのチェックボックス設定を初めて変更すると、ルールは無効になります。**[テーブルビュー]** の選択内容が優先されます。ただし、［ルールの追加］（+）アイコンを再度クリックすると、個別に選択解除または選択したテーブルが、新しいルールとして［ルール］リストに反映され、ルールが再び優先されます。**[テーブルビュー]** リストに戻るには、**[ルールの適用]** を再度クリックします。
- **[テーブルビュー]** セクションには、文字カラムの長さがバイト単位で示されます。文字あたりのバイト数はデータベースで使用される文字セットエンコーディングによって異なるため、一部のソースでは、実際のカラムの長さの文字数が **[テーブルビュー]** に表示されるバイト数と異なる場合があります。
- カラムを個別に選択した場合、結果として得られるカラムのセットはスキーマドリフトの設定に関係なく固定で、スキーマの変更によって更新されることはありません。たとえば、ソースカラムが追加または名前変更された場合、そのカラムは選択されたカラムのリストに含まれていないため、CDC 処理からサイレントに除外されます。ただし、選択したカラムがソースで削除された場合、スキーマドリフトの［カラムの削除］オプションによって処理方法が制御されます。削除されたカラムの操作は、ルールを再度適用するまでカラムのリストに反映されません。
- 初期ロードと増分ロードの組み合わせジョブの場合、カラム選択を追加または変更すると、再同期操作が自動的にトリガされます。再同期は、ターゲットテーブルがソースと同じ値を持つようにするために必要です。再同期オプションがない増分ロードジョブの場合、カラム選択を変更するとエラーになります。デプロイしたジョブのカラム選択を変更してから再デプロイすると、ソースとターゲットは一致しなくなります。

  注: 選択したカラムを削除しても、再同期操作はトリガされず、エラーは報告されません。選択したカラムの削除は、カラム削除イベントと同じように扱われ、タスクのカラム削除スキーマのドリフト設定がトリガされます。

11. DB2 for i、DB2 for z/OS、Microsoft SQL Server、Oracle、PostgreSQL、SAP HANA、または SAP HANA Cloud ソースを持つ増分ロードタスク、または初期ロードタスクと増分ロードタスクの組み合わせを定義しようとしていて、選択したソーステーブルの 1 つ以上で変更データキャプチャが有効になっていない場合は、CDC を有効にするためのスクリプトを生成してから、スクリプトを実行またはダウンロードできます。

注: Db2 for LUW、Oracle、または SQL Server ソースの CDC メソッドとして［**クエリベース**］を選択した場合、［**CDC スクリプト**］フィールドはクエリベースの変更キャプチャメソッドには適用できないため、使用できません。

a. ［**CDC スクリプト**］フィールドで、次のオプションのいずれかを選択します。

- **すべてのカラムの CDC を有効化**。選択されたソーステーブルのすべてのカラムに対して CDC を有効にします。このオプションは、DB2 for i、DB2 for z/OS、PostgreSQL、SAP HANA、SAP HANA Cloud、または SQL Server ソースの唯一の有効なオプションです。

  注: 一意のインデックスを持つテーブルを含め、プライマリキーのないソーステーブルの場合、選択されているオプションに関係なく、デフォルトですべてのカラムの CDC が有効になっています。

- **プライマリキーカラムの CDC を有効化**。選択されたソーステーブルのプライマリキーカラムに対してのみ CDC を有効にします。このオプションは、DB2 for i、DB2 for z/OS、PostgreSQL、SQL Server ソース、または Google BigQuery ターゲットを持つタスクには使用しないでください。

スクリプトは、ソースタイプに応じて、次の方法で CDC を有効にします。

- DB2 for i ソースの場合、スクリプトはソーステーブルでのジャーナルを有効にします。
- Db2 for z/OS ソースの場合、スクリプトはソーステーブルと CDC に必要な特定の Db2 カタログテーブルに対して DATA CAPTURE CHANGES を設定します。1 つのジョブに DATA CAPTURE CHANGES が設定されると、他のすべてのジョブは、Db2 で必要なカタログテーブルに対してその属性が有効になったことを認識します。これは、Db2 カタログテーブルが Db2 のすべてのユーザーによって共有されるテーブルのセットであるためです。
- Microsoft SQL Server ソースの場合、スクリプトはストアドプロシージャ sys.sp_cdc_enable_db および sys.sp_cdc_enable_table を実行して、ソースデータベースおよびテーブルで CDC を有効にします。RDS for SQL Server の場合、スクリプトは msdb.dbo.rds_cdc_enable_db プロシージャを実行してソースデータベースで CDC を有効にし、sys.sp_cdc_enable_table スクリプトを実行してテーブルの CDC を追跡します。
- Oracle ソースの場合、スクリプトは、選択したソーステーブルのすべてまたはプライマリキーカラムの補足ログを有効にして、REDO ログに追加情報を記録します。
- PostgreSQL ソースの場合、スクリプトは選択されたソーステーブルに REPLICATION IDENTITY FULL を設定して、すべてのカラム値を WAL ファイルに書き込みます。このスクリプトは、pgoutput または wal2json タイプのレプリケーションスロットも作成します。スロットタイプが pgoutput の場合、スクリプトはパブリケーションも作成し、そこにテーブルを追加します。
- SAP HANA および SAP HANA Cloud ソースの場合、スクリプトは必要な PKLOG、PROCESSED、および_CDC シャドーテーブルを作成します。このスクリプトは、選択されたソーステーブルごとに 3 つのトリガと 1 つのシーケンスも作成します。

b. スクリプトを実行するには、［**実行**］をクリックします。

スクリプトを実行できるデータベースロールまたは特権がない場合は、［ダウンロード］アイコンをクリックしてスクリプトをダウンロードします。スクリプトファイル名の形式は次のとおりです。`cdc_script_`*`taskname_number`*`.txt` 次に、データベース管理者にスクリプトの実行を依頼します。

データベース取り込みタスクを実行する前に、スクリプトが実行されていることを確認してください。

注: 後で［**CDC スクリプト**］オプションに変更して、スクリプトを再度実行すると、スクリプトは最初に元のカラムセットに対する CDC を削除し、次に現在のカラムセットに対して CDC を有効にします。SAP HANA ソースの場合、PROCESSED テーブルと PKLOG テーブルがすでに存在するときは、それらは新しいスクリプトから省略されます。シャドー_CDC テーブルとトリガが、選択されたいずれかのテーブルにすでに存在する場合、それらのオブジェクトを作成する SQL 文は、新しいスクリプトではコメントアウトされます。

12. Microsoft SQL Server ソースの場合は、次のフィールドに入力します。
    - **［キャプチャファイルグループ］**フィールドに、キャプチャ用に作成される変更テーブルに使用するファイルグループの名前を入力します。このフィールドを空のままにすると、変更テーブルはデータベースのデフォルトのファイルグループに配置されます。
    - **［ゲートロール］**フィールドに、データを変更するためのアクセスをゲートするために使用されるデータベースロールの名前を入力します。このフィールドを空のままにすると、データベースはゲートロールを使用しません。
13. テーブル選択条件に一致するソーステーブルのリストを作成してダウンロードするには、次のサブ手順を実行します。
    a. ルールベースのテーブル選択を使用した場合は、**［ルールタイプ別のテーブルのリスト］**で、使用する選択ルールのタイプを選択します。次のオプションがあります。
        - **含めるルールのみ**
        - **除外ルールのみ**
        - **含めるルールと除外ルール**
    b. 使用したテーブル選択方法に関係なく、リストにカラムを含めるには、**［カラムを含める］**チェックボックスを選択します。

        注: このオプションは MongoDB ソースでは使用できません。
    c. ［ダウンロード］アイコンをクリックします。

        カラムを含むダウンロードしたリストの形式は次のとおりです。

        status,*schema_name*,*table_name*,*object_type*,*column_name*,*comment*

        次の表に、ダウンロードしたリストに表示される情報を示します。

フィールド	説明
status	一括取り込みデータベースにサポートされていないタイプがあるために、ソーステーブルまたはカラムを処理から除外するかどうかを示します。有効な値は以下のとおりです。 - E。オブジェクトは、除外ルールによって処理から除外されます。 - I: オブジェクトは処理に含まれます。 - X。このオブジェクトはサポートされていないタイプのオブジェクトであるため、処理から除外されます。例えば、サポートされていないタイプのオブジェクトには、サポートされていないデータ型のカラムと、サポートされていないカラムのみを含むテーブルが含まれます。コメントフィールドに、サポートされていないタイプの詳細が示されます。
schema_name	ソーススキーマの名前を指定します。
table_name	ソーステーブルの名前を指定します。
object_type	ソースオブジェクトのタイプを指定します。有効な値は以下のとおりです。 - C。カラム。 - T。テーブル。

フィールド	説明
column_name	ソースカラムの名前を指定します。この情報は、**[カラム]** チェックボックスを選択した場合にのみ表示されます。
comment	サポートされていないタイプのソースオブジェクトが、選択ルールに一致していても処理から除外される理由を指定します。

14. **[詳細]** で、ソースタイプとロードタイプに応じて使用できる詳細プロパティを設定します。

プロパティ	ソースとロードタイプ	説明
フラッシュバックの無効化	Oracle ソース - 初期ロード	データベースからデータを取得するときに一括取り込みデータベースが Oracle Flashback を使用できないようにするには、このチェックボックスを選択します。 Oracle Flashback を使用するには、ユーザーに EXECUTE ON DBMS_FLASHBACK 権限を付与する必要があります。これは、初期ロードには必要ありません。 このチェックボックスは、新しい初期ロードタスクに対してデフォルトで選択されています。既存の初期ロードタスクの場合、このチェックボックスはデフォルトでクリアされているため、Oracle Flashback は有効のままになります。パーティション化が有効なタスクの場合、このチェックボックスは自動的に選択され、編集できません。

LOB を含める	Oracle ソース:
	- Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Microsoft Azure Data Lake Storage Gen 2、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、Snowflake、または SQL Server ターゲットへの初期ロード。PostgreSQL ターゲットではサポートされていません。
	PostgreSQL ソース:
	- Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Microsoft Azure Data Lake Storage Gen2、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、または Snowflake ターゲットへの初期ロードと増分ロード。
	- Azure Event Hubs への増分ロード。
	- Snowflake ターゲットへの初期ロードと増分ロードの組み合わせ。
	SQL Server ソース:
	- Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Microsoft Azure Data Lake Storage Gen 2、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、Snowflake、または SQL
		ターゲットにデータをレプリケートするラージオブジェクト（LOB）カラムがソースに含まれている場合は、このチェックボックスを選択します。
		LOB データ型は次のとおりです。
		- Oracle の場合: BLOB、CLOB、および NCLOB
		- PostgreSQL の場合: BYTEA、TEXT、および XML 加えて、JSON、JSONB など大きくなる可能性があるその他の型
		- SQL Server の場合: GEOGRAPHY、GEOMETRY、IMAGE、NTEXT、NVARCHAR(MAX)、TEXT、VARBINARY(MAX)、VARCHAR(MAX)、および XML
		ターゲットタイプに固有の切り詰めポイントによっては、LOB データが切り詰められる場合があります。
		初期ロードの場合:
		- BLOB、BYTEA、GEOGRAPHY、GEOMETRY、IMAGE、または VARBINARY(MAX)カラムは、ターゲットの BINARY カラムに書き込まれる前に切り詰められます。
		  - Amazon S3、Databricks Delta、Google Cloud Storage、Microsoft Azure Data Lake Storage Gen2、Oracle、Oracle Cloud Object Storage、および SQL Server ターゲットの場合、データは 16777216 バイトに切り詰められます。
		  - Amazon Redshift ターゲットの場合、データは 1024000 バイトに切り詰められます。
		  - Microsoft Azure Synapse Analytics ターゲットの場合、データは 1000000 バイトに切り詰められます。
		  - Google BigQuery および Snowflake ターゲットの場合、データは 8388608 バイトに切り詰められます。
		- CLOB、NCLOB、TEXT、NTEXT、NVARCHAR（MAX）、VARCHAR（MAX）、または XML カラムは、ターゲットの VARCHAR カラムに書き込まれる前に切り詰められます。
		  - Amazon S3、Databricks Delta、Google Cloud Storage、Microsoft Azure Data Lake Storage Gen2、Oracle、Oracle Cloud Object Storage、および Snowflake ターゲットの場合、データは 16777216 バイトに切り詰められます。
		  - Amazon Redshift ターゲットの場合、データは 65535 バイトに切り詰められます。
		  - Google BigQuery ターゲットの場合、データは 8388608 バイトに切り詰められます。

プロパティ	ソースとロードタイプ	説明
	Server ターゲットへの初期ロード。 - Azure Event Hubs、Databricks Delta、Snowflake、SQL Server ターゲットへの増分ロード。 - Databricks Delta、Snowflake、および SQL Server ターゲットへの初期ロードと増分ロードの組み合わせ。 **注**: CDC メソッドとして **[クエリベース]** を選択した場合、このフィールドは無効になります。	- Microsoft Azure Synapse Analytics ターゲットの場合、データは 500000 バイトに切り詰められます。 - SQL Server ターゲットの場合、CLOB、TEXT、および VARCHAR（MAX）データは 16777216 バイトに、NCLOB、NTEXT、および NVARCHAR（MAX）データは 33554432 バイトに、XML データは 33554442 バイトに切り詰められます。 増分ロードと複合ロードの PostgreSQL ソースの場合、ラージオブジェクトカラムに 1 MB を超えるデータが含まれていると、データは 1 MB に切り詰められます。 Azure Event Hubs ターゲットの場合、全体のレコードの最大サイズは 1 MB です。レコードサイズが 1 MB を超えると、Azure Event Hubs はエラーを生成し、タスクは失敗します。
永続ストレージの有効化	Db2 for LUW（クエリベースの CDC）、MongoDB、Oracle（クエリベースの CDC）、PostgreSQL、SAP HANA、SAP HANA Cloud、および SQL Server（クエリベースの CDC）を除くすべてのソース - 増分ロード、および初期ロードと増分ロードの組み合わせ。 クエリベースの CDC メソッドを使用する Db2 for LUW、Oracle、および SQL Server ソースの場合、このフィールドは表示されません。これは、永続ストレージがデフォルトで有効になっていて変更できないためです。 MongoDB、PostgreSQL、SAP HANA、および SAP HANA Cloud 変更データソースの場合、永続ストレージがデフォルトで有効になっていて変更できないため、このフィールドは表示されません。	ターゲットへのデータの書き込みが遅い場合や遅延している場合でもデータを継続的に使用できるようにディスクバッファへのトランザクションデータの永続ストレージを有効にするには、このチェックボックスを選択します。 永続ストレージを使用する利点は、ソーストランザクションログの消費が高速になり、ログアーカイブやバックアップへの依存度が低くなるため、データベース取り込みジョブを再開した後もディスクストレージ内の永続データに引き続きアクセスできることです。

プロパティ	ソースとロードタイプ	説明
パーティション化の有効化	Oracle ソース - 初期ロード、および初期ロードと増分ロードの組み合わせ SQL Server ソース - 初期ロード、および初期ロードと増分ロードの組み合わせ	ソースオブジェクトのパーティション化を有効にする場合は、このチェックボックスを選択します。オブジェクトがパーティション化されると、データベース統合ジョブは、各パーティションから読み取ったレコードを並列処理します。 Oracle ソースの場合、一括取り込みデータベースは ROWID をパーティションキーとして使用して、パーティションの範囲を決定します。また、**[パーティション化の有効化]** チェックボックスを選択すると、**[フラッシュバックの無効化]** チェックボックスが自動的に選択されます。 SQL Server ソースの場合、パーティション化はプライマリキーに基づきます。 **注**: 初期ロードと増分ロードを組み合わせた場合、ソースオブジェクトのパーティション化は初期ロードフェーズでのみ行われます。
パーティションの数	Oracle ソース - 初期ロード、および初期ロードと増分ロードの組み合わせ SQL Server ソース - 初期ロード、および初期ロードと増分ロードの組み合わせ	ソースオブジェクトのパーティション化を有効にする場合、作成するパーティションの数を入力します。デフォルト数は 5 です。最小値は 2 です。

プロパティ	ソースとロードタイプ	説明

増分ロード操作の当初の開始点	すべてのソース - 増分ロード	ソースログ内の位置をカスタマイズする場合は、このフィールドを設定します。データベース統合ジョブは、最初に実行されたときに変更レコードの読み取りをこの位置から開始します。
		次のオプションがあります。
		- **利用可能で最も早い時間**。変更が保存されているデータベースログまたはデータベース構造内で利用可能な最も早い位置。
		  - DB2 for i の場合、現在のジャーナルの先頭。
		  - Db2 for LUW の場合、このオプションは使用できません。
		  - DB2 for z/OS の場合、トランザクションログで利用可能な最も古いレコード。
		  - MongoDB の場合、このオプションは使用できません。
		  - MySQL の場合、最初の binlog ファイルで使用可能な最も古いレコード。
		  - Oracle の場合、このオプションは使用できません。
		  - PostgreSQL の場合、レプリケーションスロットで利用可能な最も古いレコード。
		  - SAP HANA および SAP HANA Cloud の場合、PKLOG テーブルで使用可能な最も古いレコード。
		  - SQL Server の場合、アクティブなトランザクションログで利用可能な最も古いレコード。**［メソッド］**が**［クエリベースの** CDC］に設定されている場合、このオプションは使用できません。
		- **使用可能な最新**。データベースログまたはデータベース構造内で利用可能な最新の位置。
		- **位置**。データベース取り込みジョブが変更レコードの取得を開始する変更ストリーム内の位置。位置の値は、Db2 for i タイムスタンプ、Db2 for z/OS LRSN、Oracle SCN、PostgreSQL LSN、SAP HANA シーケンス値、または SQL Server LSN です。この値は、現在の位置の値と同じかそれ以下にする必要があります。値が無効な場合、ジョブは失敗します。デフォルトでは値 0 が表示され、それが次の開始位置になります。
		  - Db2 for i の場合、デフォルト値の 0 を使用しないでください。
		  - Db2 for LUW の場合、このオプションは使用できません。
		  - Db2 for z/OS の場合、値 0 を指定すると、使用可能な最新のポイントが使用されます。
		  - MongoDB と MySQL の場合、このオプションは使用できません。
		  - Oracle の場合、値 0 を指定すると、使用可能な最新のポイントが使用されます。**［メソッド］**が**［クエリベースの** CDC］に設定されている場合、このオプションは使用できません。
		  - PostgreSQL の場合、値 0 を指定すると、使用可能な最も古いポイントが使用されます。
		  - SAP HANA の場合、値 0 を指定すると、使用可能な最も古いポイントが使用されます。

プロパティ	ソースとロードタイプ	説明
		- SQL Server の場合、値 0 を指定すると、使用可能な最も古いポイントが使用されます。アクティブなトランザクションログの開始より前のゼロ以外の LSN の場合、データはトランザクションログからではなく CDC テーブルから読み取られます。**［メソッド］**が**［クエリベースの** CDC］に設定されている場合、このオプションは使用できません。 - **特定の日付と時刻**。一括取り込みデータベースが、変更レコードの取得を開始する変更ストリーム内の位置を決定するために使用する、MM/DD/YYYY hh:mm AM|PM の形式の日付と時刻。一括取り込みデータベースはこの日時より後に開始された変更のみを取得します。使用可能なアーカイブログに最も早い日付と時刻より前の日付と時刻を入力すると、ジョブは失敗します。 MySQL ソースの場合、このオプションは使用できません。 デフォルトは**［使用可能な最新］**です。 **注意事項**: - **［増分ロード操作の当初の開始点］**オプションは、ジョブの最初の実行にのみ関連します。その後、停止または強制終了されたジョブを再開すると、ジョブは最後に中断したところからソースデータのプロパゲートを開始します。 - トランザクションログでログベースの CDC を使用する SQL Server の増分ロードジョブの場合、要求された LSN が使用可能なときは、トランザクションログのアクティブな部分からデータ変更が読み取られます。LSN がアクティブなトランザクションログよりも前の場合、データの変更は以前に有効化されていた CDC テーブルから読み取られます。ソーステーブルで SQL Server CDC が有効になっていることを確認してください。 - 初期ロードと増分ロードの組み合わせジョブの場合、初期ロードは、変更データの増分処理が現在のトランザクションログの末尾に到達するまで実行されません。
取得サイズ	MongoDB - 初期ロードと増分ロード	MongoDB ソースの場合、データベース統合ジョブがソースから一度に読み取る必要があるレコードの数。有効な値は 1〜2147483647 です。デフォルトは 5000。

15. **［カスタムプロパティ］**で、特別な要件を満たすために Informatica が提供するカスタムプロパティを指定できます。プロパティを追加するには、**［プロパティの作成］**フィールドに、プロパティの名前と値を入力します。次に、**［プロパティの追加］**をクリックします。

    これらのプロパティを指定する場合は、Informatica グローバルカスタマサポートにお問い合わせください。通常、これらのプロパティは、固有の環境または特別な処理のニーズに対応します。必要に応じて、複数のプロパティを指定できます。プロパティ名には、英数字と次の特殊文字のみを含めることができます: ピリオド（.）、ハイフン（-）、およびアンダースコア（_）。

    ヒント: プロパティを削除するには、リストのプロパティ行の右端にある［削除］アイコンをクリックします。

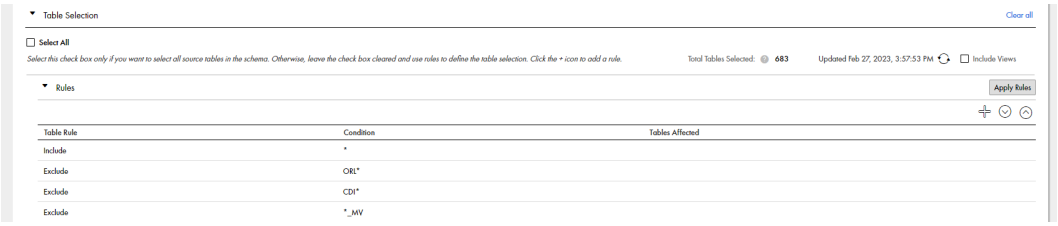
16. **［次へ］**をクリックします。

ソーステーブルを選択するルールの例

データベース統合タスクのソースを定義する場合、必要に応じてテーブル選択ルールを定義して、指定したスキーマのソーステーブルのサブセットを選択できます。この簡単な例は、選択ルールを使用して必要なテーブルを選択する方法を示しています。

4,362 個のテーブルがソーススキーマにあると仮定します。データをレプリケートする必要のないテーブルを除外したいとします。

次のルールを示されている順序で定義します。



ルールは上から下に処理されます。

- ルール 1 では、スキーマ内のすべてのソーステーブルを含めます。
- ルール 2 は、名前が「ORL」で始まるソーステーブルを除外します。
- ルール 3 は、名前が「CDI」で始まるソーステーブルを除外します。
- ルール 4 は、「_MV」で終わるソーステーブルを除外します。

［更新］アイコンをクリックした後、**［選択されたテーブル］**フィールドには 683 個のテーブルが表示されます。これは、3679 個のテーブルが除外されたことを示します。

ターゲットの設定

データベース統合タスクウィザードの**［ターゲット］**ページでターゲットを設定します。

1. **［接続］**リストで、ターゲットタイプの接続を選択します。接続タイプは、接続名の後の括弧内に表示されます。

   ランタイム環境の接続を管理者で事前に定義しておく必要があります。

   リストには、**［定義］**ページで選択されたロードタイプに有効な接続タイプのみが含まれます。ロードタイプを選択しなかった場合、接続は一覧表示されません。

   ロードタイプを変更し、選択した接続が無効になると、警告メッセージが発行され、**［接続］**フィールドがクリアされます。更新されたロードタイプに有効な別の接続を選択する必要があります。

   注: 取り込みタスクをデプロイした後は、最初に関連する取り込みジョブをデプロイ解除せずに接続を変更することはできません。その後、タスクを再度デプロイする必要があります。
2. **［ターゲット］**セクションで、ターゲットタイプに関連するプロパティを構成します。

   これらのプロパティの説明については、次のトピックを参照してください。

   - 「Amazon Redshift ターゲットのプロパティ 」 (ページ 119)
   - 「Amazon S3 ターゲットのプロパティ」 (ページ 120)
   - 「Databricks Delta ターゲットのプロパティ」 (ページ 124)
   - 「フラットファイルターゲットのプロパティ」 (ページ 126)
   - 「Google BigQuery ターゲットのプロパティ」 (ページ 129)
   - 「Google Cloud Storage ターゲットのプロパティ」 (ページ 132)

- 「Kafka ターゲットのプロパティ」 (ページ 136)
- 「Microsoft Azure Data Lake Storage ターゲットのプロパティ」 (ページ 139)
- 「Microsoft Azure Synapse Analytics ターゲットのプロパティ」 (ページ 143)
- 「Microsoft Fabric OneLake ターゲットプロパティ」 (ページ 144)
- 「Microsoft SQL Server ターゲットのプロパティ」 (ページ 146)
- 「Oracle ターゲットのプロパティ」 (ページ 147)
- 「Oracle Cloud Object Storage ターゲットのプロパティ」 (ページ 149)
- 「PostgreSQL ターゲットプロパティ」 (ページ 153)
- 「Snowflake Data Cloud ターゲットのプロパティ」 (ページ 153)

3. 選択したソーステーブルに関連付けられているターゲットオブジェクトの名前を変更する場合は、テーブルの名前変更ルールを定義します。

   例えば、TGT_などのプレフィックスを追加できます。詳細については、「ターゲットのテーブルの名前変更」 (ページ 116)を参照してください。

4. ソースデータ型からターゲットデータ型へデフォルトのマッピングをオーバーライドする場合は、データ型ルールを定義します。

   この機能は、MySQL（初期ロード）ソースまたは Oracle（いずれかのロードタイプ）ソースと SQL ベースのターゲットタイプを持つタスクに対してのみサポートされています。詳細については、「データ型マッピングのカスタマイズ」 (ページ 117)を参照してください。

5. **［カスタムプロパティ］**で、特別な要件を満たすために Informatica が提供するカスタムプロパティを指定できます。プロパティを追加するには、**［プロパティの作成］**フィールドに、プロパティの名前と値を入力します。次に、**［プロパティの追加］**をクリックします。

   これらのプロパティを指定する場合は、Informatica グローバルカスタマサポートにお問い合わせください。通常、これらのプロパティは、固有の環境または特別な処理のニーズに対応します。必要に応じて、複数のプロパティを指定できます。プロパティ名には、英数字と次の特殊文字のみを含めることができます: ピリオド（.）、ハイフン（-）、およびアンダースコア（_）。

   ヒント: プロパティを削除するには、リストのプロパティ行の右端にある［削除］アイコンボタンをクリックします。

6. ある場合は**［次へ］**をクリックするか、または**［保存］**をクリックします。

ターゲットのテーブルの名前変更

既存のスキーマを使用してターゲットを設定する場合、オプションで、選択したソーステーブルに対応するターゲットテーブルの名前を変更するためのルールを定義できます。

Apache Kafka などのターゲットメッセージングシステムの場合、ルールは出力メッセージのテーブル名を変更します。

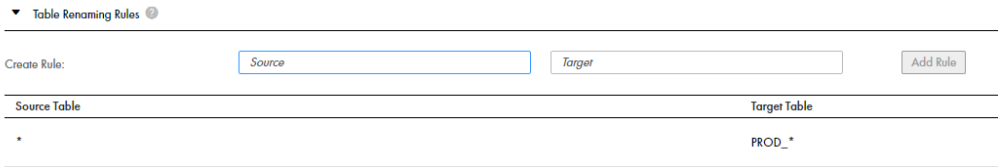
テーブルの名前を変更するためのルールを作成するには、次の手順を実行します。

1. **［テーブルの名前変更ルール］**の**［ルールの作成］**フィールドに、ソーステーブル名または 1 つ以上のワイルドカードを含むテーブル名マスクを入力します。次に、対応するターゲットテーブル名またはテーブル名マスクを入力します。

注意事項:

- ソースの場合、アスタリスク（*）ワイルドカードのみを入力すると、**［ソース］**ページのテーブル選択基準に一致するすべてのソーステーブルを選択できます。あるいは、特定のソーステーブル名、または 1 つ以上の文字を表すアスタリスク（*）ワイルドカードか単一の文字を表す疑問符（?）ワイルドカードを 1 つ以上含むテーブル名マスクを入力できます。
- ターゲットにワイルドカード文字を含むテーブル名マスクを使用するには、ソースでもワイルドカード文字を使用する必要があります。ワイルドカード文字を含むターゲットテーブルマスクで特定のソーステーブル名を使用すると、タスクのデプロイは失敗します。
- テーブル名にバックスラッシュ（\）、アスタリスク（*）、ドット（.）、疑問符（?）などの特殊文字が含まれている場合は、名前の各特殊文字をバックスラッシュ（\）でエスケープします。
- Windows では、ターゲットテーブルの名前変更条件を入力することによってターゲットテーブル名の長さが 232 文字を超える場合、名前は 222 文字に切り詰められます。一括取り込みデータベースでは、名前に日時の yyyyMMddHHmmss 値（14 文字）が追加されるため、名前は Windows の上限である 255 文字を超えます。名前を変更するターゲットテーブルの名前が 232 文字を超えないようにしてください。

2. **［ルールの追加］**をクリックします。
ルールがルールリストに表示されます。



複数のテーブルルールを定義できます。テーブルが複数のルールに一致していない限り、処理方法においてはルールの順序は重要ではありません。この場合、最後に一致するルールによってテーブルの名前が決まります。

ルールを削除するには、ルール行の右端にある［削除］アイコンをクリックします。

例:

選択したすべてのソーステーブルに対応するターゲットテーブルの名前にプレフィックス「PROD_」を追加するとします。次の値を入力します。

- ソースには、アスタリスク（*）ワイルドカード文字のみを入力して、選択されたすべてのソーステーブルを指定します。
- ターゲットには、「PROD_*」と入力して、名前でソーステーブルと一致するすべてのターゲットテーブルの名前にこのプレフィックスを追加します。

データ型マッピングのカスタマイズ

データベース取り込みタスクのターゲットを設定する場合、必要に応じてデータ型マッピングルールを定義して、ソースデータ型からターゲットデータ型へのデフォルトのマッピングをオーバーライドできます。

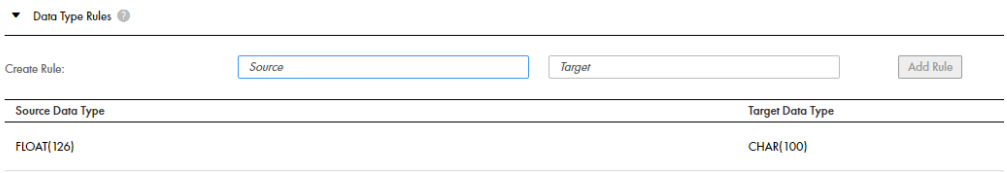
デフォルトのマッピングについては、「「デフォルトデータ型のマッピング」 (ページ 169)」を参照してください。

この機能は、MySQL（初期ロード）ソースまたは Oracle（任意のロードタイプ）ソース、および SQL をサポートするターゲットタイプ（Databricks Delta、Google BigQuery、Microsoft Azure Synapse Analytics、Oracle、SQL Server、Snowflake など）を持つタスクでサポートされています。この機能は、他のソースタイプでは認定されていません。

例えば、Snowflake VARCHAR(255)データ型へのデフォルトのマッピングを使用する代わりに、精度のない Oracle NUMBER カラムを、同じく精度のない Snowflake ターゲットの NUMBER()カラムにマッピングするデータ型ルールを作成できます。

データ型マッピングルールを作成するには、次の手順を実行します。

1. **［データ型ルール］**を展開します。
2. **［ルールの作成］**フィールドに、ソースデータ型とそれをマッピングするターゲットデータ型を入力します。
   *［ソース］*フィールドでのみ、例えば、NUMBER(%,4)、NUMBER(8,%)、または NUMBER(%)のように、パーセント（%）ワイルドカードを含めて、データ型の精度、スケール、またはサイズを表すことができます。同じデータ型だが精度、スケール、またはサイズ値が異なるソースカラムのそれぞれを個別に指定するのではなく、ワイルドカードを使用して、そのようなすべてのソース列をカバーします。例えば、FLOAT(16)、FLOAT(32)、FLOAT(84)をカバーするには、FLOAT(%)と入力します。ターゲットデータ型に%ワイルドカードを入力することはできません。%ワイルドカードを使用するソースデータ型は、特定の精度、スケール、またはサイズ値を使用するターゲットデータ型にマッピングする必要があります。例えば、ソースデータ型 FLOAT(%)を NUMBER(38,10)などのターゲットデータ型の仕様にマッピングできます。
3. **［ルールの追加］**をクリックします。
   ルールがルールリストに表示されます。



ルールを削除するには、ルール行の右端にある［削除］アイコンをクリックします。

カスタムマッピングルールを使用してタスクをデプロイした後は、タスクがデプロイ解除されるまでルールを編集できません。

使用上の注意:

- 一般に、バイナリデータ型は文字データ型にマッピングできません。
- 同じ長さまたは同じ精度とスケール値である同じソースデータ型に対して複数のデータ型ルールを定義すると、データベース取り込みタスクを保存できなくなります。
- 同じソースデータ型に対して複数のデータ型ルールを定義したが、%ワイルドカードを使用して、1 つのルールで長さまたは精度とスケール値を表し、2 番目のルールで特定の長さまたは精度とスケール値を表す場合、%ワイルドカードを使用したルールの前に、特定の値を含むルールが最初に処理されます。例えば、ソースデータ型 FLOAT(84)と FLOAT(%)をマッピングする場合、FLOAT(84)ルールが最初に処理され、次に FLOAT(%)ルールが処理されて、サイズの異なる他の FLOAT ソースカラムがカバーされます。
- ソースデータ型に長さまたは精度とスケール値が必要な場合は、%ワイルドカードまたは特定の値（例えば、VARCHAR(%)または VARCHAR(10)）を使用して必要な属性を設定してください。
- 無効なマッピングを定義すると、エラーメッセージがログに書き込まれます。その後、必要に応じて DBA のサポートを受けて、マッピングエラーを修正できます。
- Oracle ソースの場合、ソースオブジェクトの次のクエリによって返されるデータ型を使用する必要があります。

  ```
  select dbms_metadata.get_ddl('TABLE', 'YOUR_TABLE_NAME','TABLE_OWNER_NAME') from dual;
  ```
- 一括取り込みデータベースは、データ型マッピングルールの BYTE および CHAR セマンティクスをサポートしていません。
- ソースデータ型にデフォルトの精度がある場合は、ルールで指定する必要があります。例えば、TIMESTAMP の代わりに TIMESTAMP(6)を使用する必要があります。

Amazon Redshift ターゲットのプロパティ

Amazon Redshift ターゲットのあるデータベース統合タスクを定義する場合、タスクウィザードの**［ターゲット］**タブでターゲットのいくつかのプロパティを入力する必要があります。

次の表は、**［ターゲット］**に表示される Amazon Redshift ターゲットのプロパティについて説明しています。

プロパティ	説明
ターゲット作成	利用可能なただ 1 つのオプションは、**［ターゲットテーブルを作成する］**であり、これによりソーステーブルをベースにしてターゲットテーブルを生成します。 **注**: ターゲットテーブルが作成された後、一括取り込みデータベースは、後続のジョブ実行でターゲットテーブルをインテリジェントに処理します。一括取り込みデータベースは、特定の状況に応じて、ターゲットテーブルを切り詰めたり再作成したりする場合があります。
スキーマ	一括取り込みデータベースがターゲットテーブルを作成するターゲットスキーマを選択します。
バケット	Amazon Redshift に読み込むデータオブジェクトへのアクセスを保存、整理、制御する Amazon S3 のバケットコンテナの名前を指定します。
データディレクトリまたはタスクターゲットディレクトリ	一括取り込みデータベースがタスクに関連付けられたジョブの出力ファイルを格納するサブディレクトリを指定します。このフィールドは、初期ロードジョブの場合は**［データディレクトリ］**、増分ロードジョブ、または初期ロードと増分ロードの組み合わせジョブの場合は**［タスクターゲットディレクトリ］**と呼ばれます。

次の表は、**［詳細］**に表示されるターゲットの詳細プロパティについて説明しています。

プロパティ	説明
大文字と小文字の変換を有効にする	デフォルトでは、ターゲットテーブル名およびカラム名は、対応するソース名と同じ大文字と小文字で生成されます。ただし、ターゲットのクラスタレベルまたはセッションレベルのプロパティがこの大文字と小文字を区別する動作をオーバーライドしている場合を除きます。ターゲット名の大文字と小文字を制御する場合は、このチェックボックスを選択します。次に、**［大文字と小文字の変換ストラテジ］**オプションを選択します。
大文字と小文字の変換ストラテジ	**［大文字と小文字の変換を有効にする］**を選択した場合は、以下のいずれかのオプションを選択して、生成されたターゲットテーブル（またはオブジェクト）名およびカラム（またはフィールド）名の大文字と小文字の処理方法を指定します。 - **ソースと同じ**。ソーステーブル（またはオブジェクト）名およびカラム（またはフィールド）名と同じ大文字と小文字を使用します。 - UPPERCASE。すべて大文字を使用します。 - lowercase。すべて小文字を使用します。 デフォルト値は、**［ソースと同じ］**です。 **注**: 選択したストラテジは、大文字と小文字の制御に関するターゲットのクラスタレベルまたはセッションレベルのプロパティをオーバーライドします。

Amazon S3 ターゲットのプロパティ

Amazon S3 ターゲットのあるデータベース統合タスクを定義する場合、タスクウィザードの［**ターゲット**］タブでターゲットのいくつかのプロパティを入力する必要があります。

［**ターゲット**］では、次の Amazon S3 ターゲットのプロパティを入力できます。

プロパティ	説明
出力形式	出力ファイルの形式を選択します。次のオプションがあります。 - CSV - AVRO - PARQUET デフォルトの値は［CSV］です。 **注**: CSV 形式の出力ファイルでは、各フィールドの区切り文字として二重引用符（""）が使用されます。
CSV ファイルへのヘッダーの追加	［CSV］が出力形式として選択されている場合は、このチェックボックスをオンにして、ソース列名を含むヘッダーを出力 CSV ファイルに追加します。
Avro 形式	出力形式として［AVRO］を選択した場合、ソーステーブルごとに作成される Avro スキーマの形式を選択します。次のオプションがあります。 - Avro-Flat。すべての Avro フィールドを 1 つのレコードに一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Generic。ソーステーブルのすべてのカラムを Avro フィールドの単一の配列に一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Nested。各タイプの情報を個別のレコードに編成する場合に、この Avro スキーマ形式を使用します。 デフォルト値は［Avro-Flat］です。
Avro シリアル化形式	出力形式として［AVRO］が選択されている場合は、Avro 出力ファイルのシリアル化形式を選択します。次のオプションがあります。 - **なし** - Binary - JSON デフォルト値は［Binary］です。
Avro スキーマディレクトリ	出力形式として［AVRO］が選択されている場合は、一括取り込みデータベースが各ソーステーブルの Avro スキーマ定義を格納しているローカルディレクトリを指定します。スキーマ定義ファイルには、次の命名パターンがあります。 *schemaname_tablename*.txt **注**: このディレクトリが指定されていない場合、Avro スキーマ定義ファイルは作成されません。
ファイル圧縮タイプ	CSV または AVRO 出力形式の出力ファイルのファイル圧縮タイプを選択します。次のオプションがあります。 - **なし** - Deflate - Gzip - Snappy デフォルト値は［**なし**］、これは圧縮が使用されないことを意味します。

プロパティ	説明
Avro 圧縮タイプ	[AVRO] が出力形式としてが選択されている場合は、Avro 圧縮タイプを選択します。次のオプションがあります。 - **なし** - Bzip2 - Deflate - Snappy デフォルト値は **[なし]**、これは圧縮が使用されないことを意味します。
Parquet 圧縮タイプ	[PARQUET] 出力形式が選択されている場合、Parquet でサポートされている圧縮タイプを選択できます。次のオプションがあります。 - **なし** - Gzip - Snappy デフォルト値は **[なし]**、これは圧縮が使用されないことを意味します。
Deflate 圧縮レベル	[Deflate] が [Avro **圧縮タイプ**] フィールドで選択されている場合、圧縮レベルとして 0〜9 を指定します。デフォルトは 0 です。
ディレクトリタグの追加	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、Hive パーティショニングの命名規則と互換性を持たせるために適用サイクルディレクトリの名前に「dt=」プレフィックスを追加するには、このチェックボックスをオンにします。このチェックボックスはデフォルトでオフになっています。
タスクターゲットディレクトリ	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、出力データファイル、スキーマファイル、および CDC サイクルのコンテンツと完了ファイルを保持する他のディレクトリのルートディレクトリ。このフィールドを使用して、タスクのカスタムルートディレクトリを指定できます。**[親としての接続ディレクトリ]** オプションを有効にしている場合は、必要に応じて、接続プロパティで指定された親ディレクトリで使用するタスクターゲットディレクトリを指定できます。 このフィールドは、次のディレクトリフィールドのいずれかのパターンで {TaskTargetDirectory}プレースホルダが指定されている場合は必須です。
親としての接続ディレクトリ	ターゲット接続プロパティで指定されたディレクトリ値を、タスクターゲットプロパティで指定されたカスタムディレクトリパスの親ディレクトリとして使用するようにするには、このチェックボックスをオンにします。初期ロードタスクの場合、親ディレクトリは、**データディレクトリ**と**スキーマディレクトリ**で使用されます。増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、親ディレクトリは**データディレクトリ**、**スキーマディレクトリ**、**サイクル完了ディレクトリ**、および**サイクルコンテンツディレクトリ**で使用されます。 このチェックボックスはデフォルトで選択されています。オフにしたとき、初期ロードの場合は、**[データディレクトリ]** フィールドで出力ファイルへのフルパスを定義します。増分ロードの場合は、必要に応じて **[タスクターゲットディレクトリ]** でタスクのルートディレクトリを指定します。

プロパティ	説明

データディレクトリ	初期ロードタスクの場合、一括取り込みデータベースが出力データファイルとオプションでスキーマを保存するディレクトリのディレクトリ構造を定義します。ディレクトリパターンを定義するには、次のタイプのエントリを使用できます。
	- プレースホルダ{SchemaName}、{TableName}、{Timestamp}、{YY}、{YYYY}、{MM}、および{DD}（{YY}、{YYYY}、{MM}、および{DD}は日付要素です）。{Timestamp}値の形式は yyyymmdd_hhmissms です。ディレクトリパスに生成された日付と時刻は、初期ロードジョブがターゲットへのデータの転送を開始した日付と時刻を表します。
	- 特定のディレクトリ名。
	- toUpper()および toLower()関数。これは、関連付けられた(*placeholder*)の値を強制的に大文字または小文字に変換します。
	注: プレースホルダの値の大文字と小文字は区別されません。
	例:
	`myDir1/{SchemaName}/{TableName}`
	`myDir1/myDir2/{SchemaName}/{YYYY}/{MM}/{TableName}_{Timestamp}`
	`myDir1/{toLower(SchemaName)}/{TableName}_{Timestamp}`
	デフォルトのディレクトリパターンは`{TableName}_{Timestamp}`です。
	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合は、cdc-data データファイルを含むサブディレクトリへのカスタムパスを定義します。ディレクトリパターンを定義する場合は、次のタイプのエントリを使用します。
	- プレースホルダ{TaskTargetDirectory}、{SchemaName}、{TableName}、{Timestamp}、{YY}、{YYYY}、{MM}、および{DD}（{YY}、{YYYY}、{MM}、および{DD}は日付要素です）。{Timestamp}値の形式は yyyymmdd_hhmissms です。ディレクトリパスに生成された日付と時刻は、CDC サイクルが開始された日付と時刻を表します。
	toUpper または toLower 関数を含める場合は、前の例に示すように、プレースホルダ名を丸かっこで囲み、関数とプレースホルダの両方を中かっこで囲みます。
	- 特定のディレクトリ名。
	デフォルトのディレクトリパターンは`{TaskTargetDirectory}/data/{TableName}/data` です。
	注: Amazon S3、フラットファイル、Microsoft Azure Data Lake Storage Gen2、および Oracle Cloud Object Store ターゲットでは、**[親としての接続ディレクトリ]** が選択されている場合、一括取り込みデータベースは、ターゲット接続プロパティで指定されたディレクトリをデータディレクトリパスのルートとして使用します。Google Cloud Storage ターゲットの場合、一括取り込みデータベースは取り込みタスクのターゲットプロパティで指定した**バケット**名を使用します。

スキーマディレクトリ	デフォルトディレクトリ以外のディレクトリにスキーマファイルを保存する場合は、スキーマファイルを保存するカスタムディレクトリを指定できます。初期ロードの場合、便利になるように、以前に使用した値を使用できる場合はそれがドロップダウンリストに表示されます。このフィールドはオプションです。 初期ロードの場合、デフォルトでは、スキーマはデータディレクトリに保存されます。増分ロード、および初期ロードと増分ロードの組み合わせの場合、スキーマファイルのデフォルトディレクトリは`{TaskTargetDirectory}/data/{TableName}/schema` です。 **[データディレクトリ]** フィールドと同じプレースホルダを使用できます。プレースホルダは必ず中かっこ{}で囲んでください。 toUpper または toLower 関数を含める場合は、`{toLower(SchemaName)}`のように、プレースホルダ名を丸かっこで囲み、関数とプレースホルダの両方を中かっこで囲みます。 **注**: スキーマは、CSV 形式で出力データファイルにのみ書き込まれます。Parquet および Avro 形式のデータファイルには、独自の埋め込みスキーマが含まれています。
サイクル完了ディレクトリ	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、サイクル完了ファイルを含むディレクトリへのパス。デフォルトは`{TaskTargetDirectory}/cycle/completed` です。
サイクルコンテンツディレクトリ	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、サイクルコンテンツファイルを含むディレクトリへのパス。デフォルトは`{TaskTargetDirectory}/cycle/contents` です。

プロパティ	説明
データディレクトリにサイクルのパーティション化を使用する	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、各データディレクトリの下に、CDC サイクルごとにタイムスタンプサブディレクトリが作成されます。 このオプションが選択されていない場合、別のディレクトリ構造を定義しない限り、個々のデータファイルがタイムスタンプなしで同じディレクトリに書き込まれます。
サマリディレクトリにサイクルのパーティション化を使用する	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合は、サマリコンテンツサブディレクトリおよび完了サブディレクトリの下に CDC サイクルごとにタイムスタンプサブディレクトリが作成されます。
コンテンツ内の個々のファイルを一覧表示する	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合は、コンテンツサブディレクトリの下に個々のデータファイルが一覧表示されます。 **[サマリディレクトリにサイクルのパーティション化を使用する]** がオフの場合は、このオプションがデフォルトでオンになります。タイムスタンプや日付などのプレースホルダを使用してカスタムサブディレクトリを設定できる場合を除き、コンテンツサブディレクトリ内の個々のファイルがすべて一覧表示されます。 **[データディレクトリにサイクルのパーティション化を使用する]** が選択されている場合でも、必要に応じてこのチェックボックスを選択して、個々のファイルを一覧表示し、CDC サイクルごとにグループ化することができます。

[詳細] では、次の Amazon S3 ターゲット詳細プロパティを入力できます。これらのプロパティは主に増分ロードに適用されます。

フィールド	説明
操作タイプの追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイプを記録するメタデータカラムを追加するには、このチェックボックスを選択します。 増分ロードの場合、ジョブは、挿入を表す「I」、更新を表す「U」、または削除を表す「D」を書き込みます。初期ロードの場合、ジョブは常に、挿入を表す「I」を書き込みます。 デフォルトでは、このチェック ボックスは、増分ロードジョブ、初期および増分ロードジョブの場合はオンになっており、初期ロードジョブの場合はオフになっています。
操作時間の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスをオンにします。 初期ロードの場合、ジョブは常に現在の日付と時刻を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。
操作所有者の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作の所有者を記録するメタデータカラムを追加するには、このチェックボックスを選択します。 初期ロードの場合、ジョブは常に所有者として「INFA」を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。 **注**: このプロパティは、MongoDB ソースまたは PostgreSQL ソースを持つジョブでは使用できません。

フィールド	説明
操作トランザクション ID の追加	ジョブが SQL 操作のターゲットにプロパゲートする出力にソーストランザクション ID を含むメタデータカラムを追加するには、このチェックボックスを選択します。 初期ロードの場合、ジョブは常に ID として「1」を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。
前のイメージを追加	ジョブがターゲットに書き込む出力に UNDO データを含めるには、このチェックボックスを選択します。 初期ロードの場合、ジョブは null を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。

Databricks Delta ターゲットのプロパティ

Databricks Delta ターゲットのあるデータベース統合タスクを定義する場合、タスクウィザードの **[ターゲット]** タブでターゲットのいくつかのプロパティを入力する必要があります。

次の表は、**[ターゲット]** に表示される Databricks Delta ターゲットのプロパティについて説明しています。

プロパティ	説明
ターゲット作成	利用可能なただ 1 つのオプションは、**[ターゲットテーブルを作成する]** であり、これによりソーステーブルをベースにしてターゲットテーブルを生成します。 **注**: ターゲットテーブルが作成された後、一括取り込みデータベースは、後続のジョブ実行でターゲットテーブルをインテリジェントに処理します。一括取り込みデータベースは、特定の状況に応じて、ターゲットテーブルを切り詰めたり再作成したりする場合があります。
スキーマ	一括取り込みデータベースがターゲットテーブルを作成するターゲットスキーマを選択します。

プロパティ	説明
適用モード	増分ロードジョブ、および初期ロードと増分ロードの組み合わせジョブの場合に、挿入、更新、削除といったソース DML の変更がターゲットにどのように適用されるかを示します。次のオプションがあります。 - **標準**。1 回の適用サイクルの間の変更を累積し、それをターゲットに適用する前に、より少ない SQL 文になるようにそれらをインテリジェントにマージします。例えば、ソース行で更新とそれに続く削除が発生した場合、ターゲットに行は適用されません。同じカラムまたはフィールドで複数の更新が発生した場合、最後の更新のみがターゲットに適用されます。異なるカラムまたはフィールドで複数の更新が発生した場合、更新はターゲットに適用される前に 1 つの更新レコードにマージされます。 - **論理削除**。ソース削除操作を論理削除としてターゲットに適用します。論理削除では、削除された行をデータベースから実際には削除せずに、削除済みとしてマークします。例えば、ソースで削除を行うと、ターゲットの変更レコードの INFA_OPERATION_TYPE カラムに「D」が表示されます。 処理を完了するために論理的に削除されたデータが必要となる、長期にわたるビジネスプロセスがある場合、誤って削除したデータを復元する必要がある場合、または削除された値を監査目的で追跡する必要がある場合は、論理削除の使用を検討してください。 **注**: **［論理削除］**モードを使用する場合は、ソーステーブルのプライマリキーに対して更新を実行しないでください。そうしないと、ターゲットでデータ破損が発生する可能性があります。 デフォルト値は **［標準］**です。 **注**: タスクウィザードの **［ソース］**ページで手法として **［クエリベースの** CDC**］**を選択した場合、このフィールドは表示されません。
データディレクトリまたはタスクターゲットディレクトリ	一括取り込みデータベースがタスクに関連付けられたジョブの出力ファイルを格納するサブディレクトリを指定します。このフィールドは、初期ロードジョブの場合は **［データディレクトリ］**、増分ロードジョブ、または初期ロードと増分ロードの組み合わせジョブの場合は **［タスクターゲットディレクトリ］**と呼ばれます。

次の表は、**［詳細］**に表示されるターゲットの詳細プロパティについて説明しています。

注: ［操作の追加...］メタデータフィールドと **［メタデータカラムのプレフィックス］**フィールドは、**［適用モード］**を **［論理削除］**に設定した場合にのみ表示されます。

プロパティ	説明
操作タイプの追加	ジョブがターゲットにレプリケートする出力にソース SQL 操作タイプを含むメタデータカラムを追加します。デフォルトでは、このカラムは INFA_OPERATION_TYPE という名前です。 このチェックボックスは、**［適用モード］**オプションが **［論理削除］**に設定されている場合にのみ表示されます。 論理削除モードでは、ジョブは削除操作の場合は「D」を書き込み、INFA_OPERATION_TYPE カラムへの挿入と更新の場合は NULL を書き込みます。操作タイプが NULL の場合、他の［操作の追加...］メタデータカラムも NULL です。操作タイプが「D」の場合にのみ、他のメタデータカラムに NULL 以外の値が含まれます。 デフォルトでは、このチェックボックスは選択されています。選択を解除することはできません。
操作時間の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスをオンにします。 デフォルトでは、このチェックボックスは選択されていません。

プロパティ	説明
操作所有者の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作の所有者を記録するメタデータカラムを追加するには、このチェックボックスを選択します。 デフォルトでは、このチェックボックスは選択されていません。 **注**: このプロパティは、MongoDB ソースまたは PostgreSQL ソースを持つジョブでは使用できません。
操作トランザクション ID の追加	ジョブが SQL 操作のターゲットにプロパゲートする出力にソーストランザクション ID を含むメタデータカラムを追加するには、このチェックボックスを選択します。 デフォルトでは、このチェックボックスは選択されていません。
メタデータカラムのプレフィックス	追加されたメタデータカラムの名前にプレフィックスを追加し、それらを簡単に識別して、既存のカラムの名前との競合を防ぐことができるようにします。 デフォルト値は INFA_です。
アンマネージドテーブルの作成	タスクで Databricks Delta ターゲットテーブルをアンマネージドテーブルとして作成する場合は、このチェックボックスを選択します。タスクをデプロイした後は、このフィールドを編集してマネージドテーブルに切り替えることはできません。 デフォルトでは、このオプションはオフになっており、マネージドテーブルが作成されます。 Databricks Delta のマネージドテーブルとアンマネージドテーブルの詳細については、Databricks Delta のドキュメントを参照してください。
非管理対象テーブルの親ディレクトリ	Databricks Delta アンマネージドテーブルを作成する場合は、キャプチャされた DML レコードの処理時にターゲットテーブルごとに生成される Parquet ファイルを保持するために、Amazon S3 または Microsoft Azure Data Lake Storage に存在する親ディレクトリを指定する必要があります。

フラットファイルターゲットのプロパティ

データベース統合タスクを定義する場合は、フラットファイルターゲットのいくつかのプロパティをタスクウィザードの **[ターゲット]** ページで入力する必要があります。

注: フラットファイルターゲットの場合、これらのプロパティは初期ロードジョブにのみ適用されます。

[ターゲット] では、次のフラットファイルターゲットのプロパティを入力できます。

プロパティ	説明
出力形式	出力ファイルの形式を選択します。次のオプションがあります。 - CSV - AVRO デフォルトの値は [CSV] です。 **注**: CSV 形式の出力ファイルでは、各フィールドの区切り文字として二重引用符（""）が使用されます。
CSV ファイルへのヘッダーの追加	[CSV] が出力形式として選択されている場合は、このチェックボックスをオンにして、ソース列名を含むヘッダーを出力 CSV ファイルに追加します。

プロパティ	説明
Avro 形式	出力形式として［AVRO］を選択した場合、ソーステーブルごとに作成される Avro スキーマの形式を選択します。次のオプションがあります。 - Avro-Flat。すべての Avro フィールドを 1 つのレコードに一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Generic。ソーステーブルのすべてのカラムを Avro フィールドの単一の配列に一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Nested。各タイプの情報を個別のレコードに編成する場合に、この Avro スキーマ形式を使用します。 デフォルト値は［Avro-Flat］です。
Avro シリアル化形式	出力形式として［AVRO］が選択されている場合は、Avro 出力ファイルのシリアル化形式を選択します。次のオプションがあります。 - **なし** - Binary - JSON デフォルト値は［Binary］です。
Avro スキーマディレクトリ	出力形式として［AVRO］が選択されている場合は、一括取り込みデータベースが各ソーステーブルの Avro スキーマ定義を格納しているローカルディレクトリを指定します。スキーマ定義ファイルには、次の命名パターンがあります。 *schemaname_tablename*.txt **注**: このディレクトリが指定されていない場合、Avro スキーマ定義ファイルは作成されません。
ファイル圧縮タイプ	CSV または AVRO 出力形式の出力ファイルのファイル圧縮タイプを選択します。次のオプションがあります。 - **なし** - Deflate - Gzip - Snappy デフォルト値は **［なし］**、これは圧縮が使用されないことを意味します。
Avro 圧縮タイプ	［AVRO］が出力形式としてが選択されている場合は、Avro 圧縮タイプを選択します。次のオプションがあります。 - **なし** - Bzip2 - Deflate - Snappy デフォルト値は **［なし］**、これは圧縮が使用されないことを意味します。
Deflate 圧縮レベル	［Deflate］が［Avro **圧縮タイプ**］フィールドで選択されている場合、圧縮レベルとして 0～9 を指定します。デフォルトは 0 です。

プロパティ	説明

データディレクトリ	初期ロードタスクの場合、一括取り込みデータベースが出力データファイルとオプションでスキーマを保存するディレクトリのディレクトリ構造を定義します。ディレクトリパターンを定義するには、次のタイプのエントリを使用できます。
	- プレースホルダ{SchemaName}、{TableName}、{Timestamp}、{YY}、{YYYY}、{MM}、および{DD}。ここで、{YY}、{YYYY}、{MM}、および{DD}は日付要素です。{Timestamp}値の形式は yyyymmdd_hhmissms です。ディレクトリパスに生成された日付と時刻は、初期ロードジョブがターゲットへのデータの転送を開始した日付と時刻を示します。
	- 特定のディレクトリ名。
	- toUpper()および toLower()関数。これは、関連付けられた(*placeholder*)の値を大文字または小文字にすることを強制します。
	注: プレースホルダの値の大文字と小文字は区別されません。
	例:
	`myDir1/{SchemaName}/{TableName}`
	`myDir1/myDir2/{SchemaName}/{YYYY}/{MM}/{TableName}_{Timestamp}`
	`myDir1/{toLower(SchemaName)}/{TableName}_{Timestamp}`
	デフォルトのディレクトリパターンは{TableName}_{Timestamp}です。
	注: Amazon S3、フラットファイルおよび Microsoft Azure Data Lake Storage Gen2 ターゲットでは、**[親としての接続ディレクトリ]** が選択されている場合、一括取り込みデータベースは、ターゲット接続プロパティで指定されたディレクトリをデータディレクトリパスのルートとして使用します。Google Cloud Storage ターゲットの場合、一括取り込みデータベースは取り込みタスクのターゲットプロパティで指定した**バケット**名を使用します。

親としての接続ディレクトリ	初期ロードタスクの場合、ターゲット接続プロパティで指定されたディレクトリ値を、タスクターゲットプロパティで指定されたカスタムディレクトリパスの親ディレクトリとして使用するようにするには、このチェックボックスを選択します。親ディレクトリは、**データディレクトリ**と**スキーマディレクトリ**で使用されます。
スキーマディレクトリ	初期ロードタスクの場合、デフォルトディレクトリ以外のディレクトリにスキーマファイルを保存する場合は、スキーマファイルを保存するカスタムディレクトリを指定できます。このフィールドはオプションです。 デフォルトでは、スキーマはデータディレクトリに保存されます。増分ロードの場合、スキーマファイルのデフォルトディレクトリは、{TaskTargetDirectory}/data/{TableName}/schema です。 **[データディレクトリ]** フィールドと同じプレースホルダを使用できます。プレースホルダが中括弧{}で囲まれていることを確認します。

[詳細] で次のような詳細ターゲットプロパティを入力できます。

フィールド	説明
操作タイプの追加	ジョブがターゲットに伝播する出力にソース SQL 操作タイプを含むメタデータカラムを追加するには、このチェックボックスを選択します。 初期ロードの場合、ジョブは常に挿入を表す「I」を書き込みます。 デフォルトでは、このチェックボックスはオフです。
操作時間の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスをオンにします。 初期ロードの場合、ジョブは常に現在の日付と時刻を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。

フィールド	説明
操作所有者の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作の所有者を記録するメタデータカラムを追加するには、このチェックボックスを選択します。 初期ロードの場合、ジョブは常に所有者として「INFA」を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。 **注**: このプロパティは、MongoDB ソースまたは PostgreSQL ソースを持つジョブでは使用できません。
操作トランザクション ID の追加	ジョブが SQL 操作のターゲットにプロパゲートする出力にソーストランザクション ID を含むメタデータカラムを追加するには、このチェックボックスを選択します。 初期ロードの場合、ジョブは常に ID として「1」を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。
前のイメージを追加	ジョブがターゲットに書き込む出力に UNDO データを含めるには、このチェックボックスを選択します。 初期ロードの場合、ジョブは null を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。

Google BigQuery ターゲットのプロパティ

Google BigQuery ターゲットのあるデータベース統合タスクを定義する場合、タスクウィザードの **[ターゲット]** タブでターゲットのいくつかのプロパティを入力する必要があります。

次の表は、**[ターゲット]** に表示される Google BigQuery ターゲットのプロパティについて説明しています。

プロパティ	説明
ターゲット作成	利用可能なただ 1 つのオプションは、**[ターゲットテーブルを作成する]** であり、これによりソーステーブルをベースにしてターゲットテーブルを生成します。 **注**: ターゲットテーブルが作成された後、一括取り込みデータベースは、後続のジョブ実行でターゲットテーブルをインテリジェントに処理します。一括取り込みデータベースは、特定の状況に応じて、ターゲットテーブルを切り詰めたり再作成したりする場合があります。
スキーマ	一括取り込みデータベースがターゲットテーブルを作成するターゲットスキーマを選択します。

プロパティ	説明
適用モード	増分ロードジョブ、および初期ロードと増分ロードの組み合わせジョブの場合に、挿入、更新、削除といったソース DML の変更がターゲットにどのように適用されるかを示します。次のオプションがあります。 - **標準**。1 回の適用サイクルの間の変更を累積し、それをターゲットに適用する前に、より少ない SQL 文になるようにそれらをインテリジェントにマージします。例えば、ソース行で更新とそれに続く削除が発生した場合、ターゲットに行は適用されません。同じカラムまたはフィールドで複数の更新が発生した場合、最後の更新のみがターゲットに適用されます。異なるカラムまたはフィールドで複数の更新が発生した場合、更新はターゲットに適用される前に 1 つの更新レコードにマージされます。 - **監査**。ソーステーブルで実行されたすべての DML 操作の監査証跡をターゲットに適用します。ソーステーブルの各 DML 変更の行が、**[詳細]** セクションで選択した監査カラムとともに、生成されたターゲットテーブルに書き込まれます。監査カラムには、DML 操作タイプ、時刻、所有者、トランザクション ID、生成された昇順シーケンス番号、前のイメージなどの変更に関するメタデータが含まれています。監査履歴を使用して、データをターゲットデータベースに書き込む前にダウンストリームの計算または処理を実行する場合、またはキャプチャされた変更に関するメタデータを調べる場合は、監査適用モードの使用を検討してください。 デフォルト値は **[標準]** です。 **注**: タスクウィザードの **[ソース]** ページで手法として **[クエリベースの** CDC**]** を選択した場合、このフィールドは表示されません。
バケット	Google Cloud Storage に読み込むデータオブジェクトへのアクセスを保存、整理、制御する既存のバケットコンテナの名前を指定します。
データディレクトリまたはタスクターゲットディレクトリ	一括取り込みデータベースがタスクに関連付けられたジョブの出力ファイルを格納するサブディレクトリを指定します。このフィールドは、初期ロードジョブの場合は **[データディレクトリ]**、増分ロードジョブ、または初期ロードと増分ロードの組み合わせジョブの場合は **[タスクターゲットディレクトリ]** と呼ばれます。

次の表は、**[詳細]** に表示されるターゲットの詳細プロパティについて説明しています。

プロパティ	説明
最終レプリケート時刻を追加	ターゲットテーブルでレコードが挿入または最後に更新された時点のタイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスをオンにします。初期ロードでは、ロードされたすべてのレコードのタイムスタンプが同じになります。増分ロード、および初期ロードと増分ロードの組み合わせの場合、このカラムには、ターゲットに適用された最後の DML 操作のタイムスタンプが記録されます。 デフォルトでは、このチェックボックスは選択されていません。
操作タイプの追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイプを記録するメタデータカラムを追加するには、このチェックボックスを選択します。 ジョブは、挿入を表す「I」、更新を表す「U」、または削除を表す「D」を書き込みます。 このフィールドは、**[適用モード]** が **[監査]** に設定されている場合にのみ使用できます。 デフォルトでは、このチェックボックスは選択されています。

プロパティ	説明
操作時間の追加	ジョブがターゲットテーブルにプロパゲートする出力にソース SQL 操作タイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスを選択します。 このフィールドは、**[適用モード]** が **[監査]** に設定されている場合にのみ使用できます。 デフォルトでは、このチェックボックスは選択されていません。
操作所有者の追加	ジョブがターゲットテーブルにプロパゲートする出力にソース SQL 操作の所有者を記録するメタデータカラムを追加するには、このチェックボックスを選択します。 このフィールドは、**[適用モード]** が **[監査]** に設定されている場合にのみ使用できます。 デフォルトでは、このチェックボックスは選択されていません。 **注**: このプロパティは、MongoDB ソースまたは PostgreSQL ソースを持つジョブでは使用できません。
操作トランザクション ID の追加	ジョブが SQL 操作のターゲットにプロパゲートする出力にソーストランザクション ID を含むメタデータカラムを追加するには、このチェックボックスを選択します。 このフィールドは、**[適用モード]** が **[監査]** に設定されている場合にのみ使用できます。 デフォルトでは、このチェックボックスは選択されていません。
操作シーケンスの追加	ジョブがターゲットテーブルに挿入する変更操作ごとに、生成された昇順のシーケンス番号を記録するメタデータカラムを追加するには、このチェックボックスを選択します。シーケンス番号には、操作の変更ストリーム位置が反映されます。 このフィールドは、**[適用モード]** が **[監査]** に設定されている場合にのみ使用できます。 デフォルトでは、このチェックボックスは選択されていません。
前のイメージを追加	ジョブがターゲットテーブルに挿入する出力に UNDO の「操作前のイメージ」データを含む _OLD カラムを追加するには、このチェックボックスを選択します。これにより、各データカラムの以前の値と現在の値を比較できるようになります。削除操作の場合、現在の値は NULL になります。 このフィールドは、**[適用モード]** が **[監査]** に設定されている場合にのみ使用できます。 デフォルトでは、このチェックボックスは選択されていません。
メタデータカラムのプレフィックス	追加されたメタデータカラムの名前にプレフィックスを追加し、それらを簡単に識別して、既存のカラムの名前との競合を防ぐことができるようにします。 プレフィックスには特殊文字を含めないようにしてください。特殊文字を含めた場合、タスクのデプロイメントが失敗します。 デフォルト値は INFA_です。
大文字と小文字の変換を有効にする	デフォルトでは、ターゲットテーブル名およびカラム名は、対応するソース名と同じ大文字と小文字で生成されます。ただし、ターゲットのクラスタレベルまたはセッションレベルのプロパティがこの大文字と小文字を区別する動作をオーバーライドしている場合を除きます。ターゲット名の大文字と小文字を制御する場合は、このチェックボックスを選択します。次に、**[大文字と小文字の変換ストラテジ]** オプションを選択します。
大文字と小文字の変換ストラテジ	**[大文字と小文字の変換を有効にする]** を選択した場合は、以下のいずれかのオプションを選択して、生成されたターゲットテーブル（またはオブジェクト）名およびカラム（またはフィールド）名の大文字と小文字の処理方法を指定します。 - **ソースと同じ**。ソーステーブル（またはオブジェクト）名およびカラム（またはフィールド）名と同じ大文字と小文字を使用します。 - UPPERCASE。すべて大文字を使用します。 - lowercase。すべて小文字を使用します。 デフォルト値は、**[ソースと同じ]** です。 **注**: 選択したストラテジは、大文字と小文字の制御に関するターゲットのクラスタレベルまたはセッションレベルのプロパティをオーバーライドします。

Google Cloud Storage ターゲットのプロパティ

Google Cloud Storage ターゲットのあるデータベース統合タスクを定義する場合、タスクウィザードの**［ターゲット］**タブでターゲットのいくつかのプロパティを入力する必要があります。

［ターゲット］では、次の Google Cloud Storage ターゲットのプロパティを入力できます。

プロパティ	説明
出力形式	出力ファイルの形式を選択します。次のオプションがあります。 - CSV - AVRO - PARQUET デフォルトの値は［CSV］です。 **注**: CSV 形式の出力ファイルでは、各フィールドの区切り文字として二重引用符（""）が使用されます。
CSV ファイルへのヘッダーの追加	［CSV］が出力形式として選択されている場合は、このチェックボックスをオンにして、ソース列名を含むヘッダーを出力 CSV ファイルに追加します。
Avro 形式	出力形式として［AVRO］を選択した場合、ソーステーブルごとに作成される Avro スキーマの形式を選択します。次のオプションがあります。 - Avro-Flat。すべての Avro フィールドを 1 つのレコードに一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Generic。ソーステーブルのすべてのカラムを Avro フィールドの単一の配列に一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Nested。各タイプの情報を個別のレコードに編成する場合に、この Avro スキーマ形式を使用します。 デフォルト値は［Avro-Flat］です。
Avro シリアル化形式	出力形式として［AVRO］が選択されている場合は、Avro 出力ファイルのシリアル化形式を選択します。次のオプションがあります。 - **なし** - Binary - JSON デフォルト値は［Binary］です。
Avro スキーマディレクトリ	出力形式として［AVRO］が選択されている場合は、一括取り込みデータベースが各ソーステーブルの Avro スキーマ定義を格納しているローカルディレクトリを指定します。スキーマ定義ファイルには、次の命名パターンがあります。 *schemaname_tablename*.txt **注**: このディレクトリが指定されていない場合、Avro スキーマ定義ファイルは作成されません。
ファイル圧縮タイプ	CSV または AVRO 出力形式の出力ファイルのファイル圧縮タイプを選択します。次のオプションがあります。 - **なし** - Deflate - Gzip - Snappy デフォルト値は**［なし］**、これは圧縮が使用されないことを意味します。

プロパティ	説明
Avro 圧縮タイプ	[AVRO] が出力形式としてが選択されている場合は、Avro 圧縮タイプを選択します。次のオプションがあります。 - **なし** - Bzip2 - Deflate - Snappy デフォルト値は **[なし]**、これは圧縮が使用されないことを意味します。
Parquet 圧縮タイプ	[PARQUET] 出力形式が選択されている場合、Parquet でサポートされている圧縮タイプを選択できます。次のオプションがあります。 - **なし** - Gzip - Snappy デフォルト値は **[なし]**、これは圧縮が使用されないことを意味します。
Deflate 圧縮レベル	[Deflate] が [Avro **圧縮タイプ**] フィールドで選択されている場合、圧縮レベルとして 0〜9 を指定します。デフォルトは 0 です。
ディレクトリタグの追加	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、Hive パーティショニングの命名規則と互換性を持たせるために適用サイクルディレクトリの名前に「dt=」プレフィックスを追加するには、このチェックボックスをオンにします。このチェックボックスはデフォルトでオフになっています。
バケット	Google Cloud Storage に読み込むデータオブジェクトへのアクセスを保存、整理、制御する既存のバケットコンテナの名前を指定します。
タスクターゲットディレクトリ	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、出力データファイル、スキーマファイル、および CDC サイクルのコンテンツと完了ファイルを保持する他のディレクトリのルートディレクトリ。このフィールドを使用して、タスクのカスタムルートディレクトリを指定できます。**[親としての接続ディレクトリ]** オプションを有効にしている場合は、必要に応じて、接続プロパティで指定された親ディレクトリで使用するタスクターゲットディレクトリを指定できます。 このフィールドは、次のディレクトリフィールドのいずれかのパターンで {TaskTargetDirectory}プレースホルダが指定されている場合は必須です。

プロパティ	説明

データディレクトリ	初期ロードタスクの場合、一括取り込みデータベースが出力データファイルとオプションでスキーマを保存するディレクトリのディレクトリ構造を定義します。ディレクトリパターンを定義するには、次のタイプのエントリを使用できます。
	- プレースホルダ{SchemaName}、{TableName}、{Timestamp}、{YY}、{YYYY}、{MM}、および{DD}（{YY}、{YYYY}、{MM}、および{DD}は日付要素です）。{Timestamp}値の形式は yyyymmdd_hhmissms です。ディレクトリパスに生成された日付と時刻は、初期ロードジョブがターゲットへのデータの転送を開始した日付と時刻を表します。
	- 特定のディレクトリ名。
	- toUpper()および toLower()関数。これは、関連付けられた(*placeholder*)の値を強制的に大文字または小文字に変換します。
	注: プレースホルダの値の大文字と小文字は区別されません。
	例:
	`myDir1/{SchemaName}/{TableName}`
	`myDir1/myDir2/{SchemaName}/{YYYY}/{MM}/{TableName}_{Timestamp}`
	`myDir1/{toLower(SchemaName)}/{TableName}_{Timestamp}`
	デフォルトのディレクトリパターンは`{TableName}_{Timestamp}`です。
	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合は、cdc-data データファイルを含むサブディレクトリへのカスタムパスを定義します。ディレクトリパターンを定義する場合は、次のタイプのエントリを使用します。
	- プレースホルダ{TaskTargetDirectory}、{SchemaName}、{TableName}、{Timestamp}、{YY}、{YYYY}、{MM}、および{DD}（{YY}、{YYYY}、{MM}、および{DD}は日付要素です）。{Timestamp}値の形式は yyyymmdd_hhmissms です。ディレクトリパスに生成された日付と時刻は、CDC サイクルが開始された日付と時刻を表します。
	toUpper または toLower 関数を含める場合は、前の例に示すように、プレースホルダ名を丸かっこで囲み、関数とプレースホルダの両方を中かっこで囲みます。
	- 特定のディレクトリ名。
	デフォルトのディレクトリパターンは`{TaskTargetDirectory}/data/{TableName}/data` です。
	注: Amazon S3、フラットファイル、Microsoft Azure Data Lake Storage Gen2、および Oracle Cloud Object Store ターゲットでは、**[親としての接続ディレクトリ]** が選択されている場合、一括取り込みデータベースは、ターゲット接続プロパティで指定されたディレクトリをデータディレクトリパスのルートとして使用します。Google Cloud Storage ターゲットの場合、一括取り込みデータベースは取り込みタスクのターゲットプロパティで指定した**バケット**名を使用します。

スキーマディレクトリ	デフォルトディレクトリ以外のディレクトリにスキーマファイルを保存する場合は、スキーマファイルを保存するカスタムディレクトリを指定できます。初期ロードの場合、便利になるように、以前に使用した値を使用できる場合はそれがドロップダウンリストに表示されます。このフィールドはオプションです。 初期ロードの場合、デフォルトでは、スキーマはデータディレクトリに保存されます。増分ロード、および初期ロードと増分ロードの組み合わせの場合、スキーマファイルのデフォルトディレクトリは`{TaskTargetDirectory}/data/{TableName}/schema` です。 **[データディレクトリ]** フィールドと同じプレースホルダを使用できます。プレースホルダは必ず中かっこ{}で囲んでください。 toUpper または toLower 関数を含める場合は、`{toLower(SchemaName)}`のように、プレースホルダ名を丸かっこで囲み、関数とプレースホルダの両方を中かっこで囲みます。 **注**: スキーマは、CSV 形式で出力データファイルにのみ書き込まれます。Parquet および Avro 形式のデータファイルには、独自の埋め込みスキーマが含まれています。
サイクル完了ディレクトリ	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、サイクル完了ファイルを含むディレクトリへのパス。デフォルトは`{TaskTargetDirectory}/cycle/completed` です。
サイクルコンテンツディレクトリ	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、サイクルコンテンツファイルを含むディレクトリへのパス。デフォルトは`{TaskTargetDirectory}/cycle/contents` です。

プロパティ	説明
データディレクトリにサイクルのパーティション化を使用する	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、各データディレクトリの下に、CDC サイクルごとにタイムスタンプサブディレクトリが作成されます。 このオプションが選択されていない場合、別のディレクトリ構造を定義しない限り、個々のデータファイルがタイムスタンプなしで同じディレクトリに書き込まれます。
サマリディレクトリにサイクルのパーティション化を使用する	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合は、サマリコンテンツサブディレクトリおよび完了サブディレクトリの下に CDC サイクルごとにタイムスタンプサブディレクトリが作成されます。
コンテンツ内の個々のファイルを一覧表示する	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合は、コンテンツサブディレクトリの下に個々のデータファイルが一覧表示されます。 **[サマリディレクトリにサイクルのパーティション化を使用する]** がオフの場合は、このオプションがデフォルトでオンになります。タイムスタンプや日付などのプレースホルダを使用してカスタムサブディレクトリを設定できる場合を除き、コンテンツサブディレクトリ内の個々のファイルがすべて一覧表示されます。 **[データディレクトリにサイクルのパーティション化を使用する]** が選択されている場合でも、必要に応じてこのチェックボックスを選択して、個々のファイルを一覧表示し、CDC サイクルごとにグループ化することができます。

[詳細] では、次の Google Cloud Storage ターゲット詳細プロパティを入力できます。これらのプロパティは主に増分ロードジョブに使用されます。

フィールド	説明
操作タイプの追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイプを記録するメタデータカラムを追加するには、このチェックボックスを選択します。 増分ロードの場合、ジョブは、挿入を表す「I」、更新を表す「U」、または削除を表す「D」を書き込みます。初期ロードの場合、ジョブは常に、挿入を表す「I」を書き込みます。 デフォルトでは、このチェック ボックスは、増分ロードジョブ、初期および増分ロードジョブの場合はオンになっており、初期ロードジョブの場合はオフになっています。
操作時間の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスをオンにします。 初期ロードの場合、ジョブは常に現在の日付と時刻を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。
操作所有者の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作の所有者を記録するメタデータカラムを追加するには、このチェックボックスを選択します。 初期ロードの場合、ジョブは常に所有者として「INFA」を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。 **注**: このプロパティは、MongoDB ソースまたは PostgreSQL ソースを持つジョブでは使用できません。

フィールド	説明
操作トランザクション ID の追加	ジョブが SQL 操作のターゲットにプロパゲートする出力にソーストランザクション ID を含むメタデータカラムを追加するには、このチェックボックスを選択します。 初期ロードの場合、ジョブは常に ID として「1」を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。
前のイメージを追加	ジョブがターゲットに書き込む出力に UNDO データを含めるには、このチェックボックスを選択します。 初期ロードの場合、ジョブは null を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。

Kafka ターゲットのプロパティ

データベース統合タスクを定義する場合は、Kafka ターゲットのいくつかのプロパティをタスクウィザードの **[ターゲット]** ページで入力する必要があります。

これらのプロパティは、増分ロード操作にのみ適用されます。

次の表は、**[ターゲット]** に表示される Kafka ターゲットのプロパティについて説明しています。

プロパティ	説明
テーブル名をトピック名として使用	一括取り込みデータベースがソースデータを含むメッセージを、ソーステーブルごとに 1 つずつ個別のトピックに書き込むか、すべてのメッセージを 1 つのトピックに書き込むかを示します。 テーブル固有のトピックを区切るメッセージを書き込むには、このチェックボックスを選択します。トピック名は、**[スキーマ名を含める]**、**[テーブルプレフィックス]**、または **[テーブルサフィックス]** プロパティに追加しない限り、トピック名はソーステーブル名に一致します。 デフォルトでは、このチェックボックスはオフです。デフォルト設定では、**[トピック名]** プロパティにすべてのメッセージが書き込まれる単一のトピックの名前を指定する必要があります。
スキーマ名を含める	**[テーブル名をトピック名として使用]** が選択されている場合、このチェックボックスが表示され、デフォルトで選択されています。この設定により、テーブル固有のトピック名にソーススキーマ名が追加されます。トピック名の形式は次のとおりです。*スキーマ名_テーブル名*。 スキーマ名を含めない場合は、このチェックボックスをオフにします。
テーブルプレフィックス	**テーブル名をトピック名として使用**を選択すると、このプロパティが表示され、オプションでプレフィックスを入力してテーブル固有のトピック名に追加できます。例えば、myprefix_を指定すると、トピック名の形式は「myprefix_*テーブル名*」になります。プレフィックスの後のアンダースコア（_）を省略すると、プレフィックスがテーブル名の前に追加されます。
テーブルサフィックス	**テーブル名をトピック名として使用**を選択すると、このプロパティが表示され、オプションでサフィックスを入力してテーブル固有のトピック名に追加できます。例えば、_mysuffix を指定すると、トピック名の形式は「*テーブル名*_mysuffix」になります。サフィックスの前のアンダースコア（_）を省略すると、サフィックスがテーブル名に追加されます。
トピック名	**[トピック名としてテーブル名を使用する]** を*選択しない*場合、ソースデータを含むすべてのメッセージが書き込まれる単一の Kafka トピックの名前を入力する必要があります。

プロパティ	説明
出力形式	出力ファイルの形式を選択します。次のオプションがあります。 - CSV - AVRO - JSON デフォルトの値は［CSV］です。 **注**: CSV 形式の出力ファイルでは、各フィールドの区切り文字として二重引用符（""）が使用されます。 Kafka ターゲットが Confluent Schema Registry を使用して増分ロードジョブのスキーマを格納する場合は、形式として［AVRO］を選択する必要があります。
JSON 形式	出力形式として［JSON］が選択されている場合は、出力の詳細レベルを選択します。次のオプションがあります。 - **簡潔**。この形式では、操作タイプやカラムの名前と値など、最も関連性の高いデータのみが出力に記録されます。 - **詳細**。この形式では、テーブル名やカラムタイプなどの詳細情報が記録されます。
Avro 形式	出力形式として［AVRO］を選択した場合、ソーステーブルごとに作成される Avro スキーマの形式を選択します。次のオプションがあります。 - Avro-Flat。すべての Avro フィールドを 1 つのレコードに一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Generic。ソーステーブルのすべてのカラムを Avro フィールドの単一の配列に一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Nested。各タイプの情報を個別のレコードに編成する場合に、この Avro スキーマ形式を使用します。 デフォルト値は［Avro-Flat］です。
Avro シリアル化形式	出力形式として AVRO が選択されている場合は、Avro 出力ファイルのシリアル化形式を選択します。次のオプションがあります。 - Binary - JSON - **なし** デフォルト値は［Binary］です。 Confluent Schema Registry を使用してスキーマを格納する Confluent Kafka ターゲットがある場合は、**［なし］**を選択します。それ以外の場合、Confluent Schema Registry はスキーマを登録しません。Confluent Scheme Registry を使用していない場合、**［なし］**は選択しないでください。
Avro スキーマディレクトリ	出力形式として［AVRO］が選択されている場合は、一括取り込みデータベースが各ソーステーブルの Avro スキーマ定義を格納しているローカルディレクトリを指定します。スキーマ定義ファイルには、次の命名パターンがあります。 *schemaname_tablename*.txt **注**: このディレクトリが指定されていない場合、Avro スキーマ定義ファイルは作成されません。 ソーススキーマの変更によってターゲットが変更されることが予想される場合、Avro スキーマ定義ファイルは、タイムスタンプを含む一意の名前で次の形式で再生成されます。 *schemaname_tablename_YYYYMMDDhhmmss*.txt この一意の命名パターンにより、古いスキーマ定義ファイルが監査目的で保持されます。

プロパティ	説明
Avro 圧縮タイプ	[AVRO] が出力形式としてが選択されている場合は、Avro 圧縮タイプを選択します。次のオプションがあります。 - **なし** - Bzip2 - Deflate - Snappy デフォルト値は **[なし]**、これは圧縮が使用されないことを意味します。
Deflate 圧縮レベル	[Deflate] が [Avro **圧縮タイプ**] フィールドで選択されている場合、圧縮レベルとして 0〜9 を指定します。デフォルトは 0 です。

[詳細] で次のような詳細ターゲットプロパティを入力できます。

プロパティ	説明
操作タイプの追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイプを含むメタデータカラムを追加するには、このチェックボックスを選択します。 ジョブは、挿入を表す「I」、更新を表す「U」、または削除を表す「D」を書き込みます。 デフォルトでは、このチェックボックスは選択されています。
操作時間の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスをオンにします。 デフォルトでは、このチェックボックスは選択されていません。
操作所有者の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作の所有者を記録するメタデータカラムを追加するには、このチェックボックスを選択します。 デフォルトでは、このチェックボックスは選択されていません。 **注**: このプロパティは、MongoDB ソースまたは PostgreSQL ソースを持つジョブでは使用できません。
操作トランザクション ID の追加	ジョブが SQL 操作のターゲットにプロパゲートする出力にソーストランザクション ID を含むメタデータカラムを追加するには、このチェックボックスを選択します。 デフォルトでは、このチェックボックスは選択されていません。
前のイメージを追加	ジョブがターゲットに書き込む出力に UNDO データを含めるには、このチェックボックスを選択します。 デフォルトでは、このチェックボックスは選択されていません。

プロパティ	説明
非同期書き込み	Kafka へのメッセージの同期配信を使用するかどうかを制御します。 - 同期配信を使用するには、このチェックボックスをオフにします。Kafka は、一括取り込みデータベースが次のメッセージを送信する前に、各メッセージの受信を確認する必要があります。このモードでは、Kafka が重複メッセージを受信する可能性はほとんどありません。ただし、パフォーマンスが低下する可能性があります。 - 非同期配信を使用するには、このチェックボックスを選択します。一括取り込みデータベースソースから変更が取得された順序に関係なく、できるだけ早くメッセージを送信します。 デフォルトでは、このチェックボックスは選択されています。

プロデューサ設定プロパティ	*key=value* ペアをカンマで区切って指定して、Apache Kafka、Confluent Kafka、Amazon Managed Streaming for Apache Kafka（MSK）、または Kafka 対応 Azure Event Hubs ターゲットの Kafka プロデューサプロパティを入力します。
	Confluent Schema Registry を使用してスキーマを格納する Confluent ターゲットがある場合は、次のプロパティを指定する必要があります。
	`schema.registry.url=url,`
	`key.serializer=org.apache.kafka.common.serialization.StringSerializer,`
	`value.serializer=io.confluent.kafka.serializers.KafkaAvroSerializer`
	Kafka プロデューサーのプロパティは、このフィールドまたは Kafka 接続の**追加の接続プロパティ**フィールドのいずれかに設定します。
	このフィールドにプロデューサープロパティを入力すると、プロパティはこのタスクにのみ関連付けられたデータベース取り込みジョブに関係します。接続のプロデューサプロパティを入力する場合、**［プロデューサ設定プロパティ］**フィールドのプロパティを指定して特定のタスクの接続レベルのプロパティをオーバーライドしない限り、プロパティは接続定義を使用するすべてのタスクのジョブに関係します。
	Kafka プロデューサプロパティの詳細については、Apache Kafka、Confluent Kafka、Amazon MSK、または Azure Event Hubs のドキュメントを参照してください。

Microsoft Azure Data Lake Storage ターゲットのプロパティ

Microsoft Azure Data Lake Storage ターゲットのあるデータベース統合タスクを定義する場合、タスクウィザードの **［ターゲット］** ページでターゲットのいくつかのプロパティを入力する必要があります。

［ターゲット］ では、次の Microsoft Azure Data Lake Storage ターゲットのプロパティを入力できます。

プロパティ	説明
出力形式	出力ファイルの形式を選択します。次のオプションがあります。 - CSV - AVRO - PARQUET デフォルトの値は［CSV］です。 **注**: CSV 形式の出力ファイルでは、各フィールドの区切り文字として二重引用符（""）が使用されます。
CSV ファイルへのヘッダーの追加	［CSV］が出力形式として選択されている場合は、このチェックボックスをオンにして、ソース列名を含むヘッダーを出力 CSV ファイルに追加します。

プロパティ	説明
Avro 形式	出力形式として［AVRO］を選択した場合、ソーステーブルごとに作成される Avro スキーマの形式を選択します。次のオプションがあります。 - Avro-Flat。すべての Avro フィールドを 1 つのレコードに一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Generic。ソーステーブルのすべてのカラムを Avro フィールドの単一の配列に一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Nested。各タイプの情報を個別のレコードに編成する場合に、この Avro スキーマ形式を使用します。 デフォルト値は［Avro-Flat］です。
Avro シリアル化形式	出力形式として［AVRO］が選択されている場合は、Avro 出力ファイルのシリアル化形式を選択します。次のオプションがあります。 - **なし** - Binary - JSON デフォルト値は［Binary］です。
Avro スキーマディレクトリ	出力形式として［AVRO］が選択されている場合は、一括取り込みデータベースが各ソーステーブルの Avro スキーマ定義を格納しているローカルディレクトリを指定します。スキーマ定義ファイルには、次の命名パターンがあります。 *schemaname_tablename*.txt **注**: このディレクトリが指定されていない場合、Avro スキーマ定義ファイルは作成されません。
ファイル圧縮タイプ	CSV または AVRO 出力形式の出力ファイルのファイル圧縮タイプを選択します。次のオプションがあります。 - **なし** - Deflate - Gzip - Snappy デフォルト値は［**なし**］、これは圧縮が使用されないことを意味します。
Avro 圧縮タイプ	［AVRO］が出力形式としてが選択されている場合は、Avro 圧縮タイプを選択します。次のオプションがあります。 - **なし** - Bzip2 - Deflate - Snappy デフォルト値は［**なし**］、これは圧縮が使用されないことを意味します。
Parquet 圧縮タイプ	［PARQUET］出力形式が選択されている場合、Parquet でサポートされている圧縮タイプを選択できます。次のオプションがあります。 - **なし** - Gzip - Snappy デフォルト値は［**なし**］、これは圧縮が使用されないことを意味します。
Deflate 圧縮レベル	［Deflate］が［Avro **圧縮タイプ**］フィールドで選択されている場合、圧縮レベルとして 0〜9 を指定します。デフォルトは 0 です。
ディレクトリタグの追加	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、Hive パーティショニングの命名規則と互換性を持たせるために適用サイクルディレクトリの名前に「dt=」プレフィックスを追加するには、このチェックボックスをオンにします。このチェックボックスはデフォルトでオフになっています。

プロパティ	説明
タスクターゲットディレクトリ	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、出力データファイル、スキーマファイル、および CDC サイクルのコンテンツと完了ファイルを保持する他のディレクトリのルートディレクトリ。このフィールドを使用して、タスクのカスタムルートディレクトリを指定できます。**［親としての接続ディレクトリ］**オプションを有効にしている場合は、必要に応じて、接続プロパティで指定された親ディレクトリで使用するタスクターゲットディレクトリを指定できます。 このフィールドは、次のディレクトリフィールドのいずれかのパターンで{TaskTargetDirectory}プレースホルダが指定されている場合は必須です。
親としての接続ディレクトリ	ターゲット接続プロパティで指定されたディレクトリ値を、タスクターゲットプロパティで指定されたカスタムディレクトリパスの親ディレクトリとして使用するようにするには、このチェックボックスをオンにします。初期ロードタスクの場合、親ディレクトリは、**データディレクトリ**と**スキーマディレクトリ**で使用されます。増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、親ディレクトリは**データディレクトリ**、**スキーマディレクトリ**、**サイクル完了ディレクトリ**、および**サイクルコンテンツディレクトリ**で使用されます。 このチェックボックスはデフォルトで選択されています。オフにしたとき、初期ロードの場合は、**［データディレクトリ］**フィールドで出力ファイルへのフルパスを定義します。増分ロードの場合は、必要に応じて**［タスクターゲットディレクトリ］**でタスクのルートディレクトリを指定します。

データディレクトリ	初期ロードタスクの場合、一括取り込みデータベースが出力データファイルとオプションでスキーマを保存するディレクトリのディレクトリ構造を定義します。ディレクトリパターンを定義するには、次のタイプのエントリを使用できます。
	- プレースホルダ{SchemaName}、{TableName}、{Timestamp}、{YY}、{YYYY}、{MM}、および{DD}（{YY}、{YYYY}、{MM}、および{DD}は日付要素です）。{Timestamp}値の形式は yyyymmdd_hhmissms です。ディレクトリパスに生成された日付と時刻は、初期ロードジョブがターゲットへのデータの転送を開始した日付と時刻を表します。
	- 特定のディレクトリ名。
	- toUpper()および toLower()関数。これは、関連付けられた(*placeholder*)の値を強制的に大文字または小文字に変換します。
	注: プレースホルダの値の大文字と小文字は区別されません。
	例:
	`myDir1/{SchemaName}/{TableName}`
	`myDir1/myDir2/{SchemaName}/{YYYY}/{MM}/{TableName}_{Timestamp}`
	`myDir1/{toLower(SchemaName)}/{TableName}_{Timestamp}`
	デフォルトのディレクトリパターンは{TableName}_{Timestamp}です。
	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合は、cdc-data データファイルを含むサブディレクトリへのカスタムパスを定義します。ディレクトリパターンを定義する場合は、次のタイプのエントリを使用します。
	- プレースホルダ{TaskTargetDirectory}、{SchemaName}、{TableName}、{Timestamp}、{YY}、{YYYY}、{MM}、および{DD}（{YY}、{YYYY}、{MM}、および{DD}は日付要素です）。{Timestamp}値の形式は yyyymmdd_hhmissms です。ディレクトリパスに生成された日付と時刻は、CDC サイクルが開始された日付と時刻を表します。
	toUpper または toLower 関数を含める場合は、前の例に示すように、プレースホルダ名を丸かっこで囲み、関数とプレースホルダの両方を中かっこで囲みます。
	- 特定のディレクトリ名。
	デフォルトのディレクトリパターンは{TaskTargetDirectory}/data/{TableName}/data です。
	注: Amazon S3、フラットファイル、Microsoft Azure Data Lake Storage Gen2、および Oracle Cloud Object Store ターゲットでは、**［親としての接続ディレクトリ］**が選択されている場合、一括取り込みデータベースは、ターゲット接続プロパティで指定されたディレクトリをデータディレクトリパスのルートとして使用します。Google Cloud Storage ターゲットの場合、一括取り込みデータベースは取り込みタスクのターゲットプロパティで指定した**バケット**名を使用します。

プロパティ	説明
スキーマディレクトリ	デフォルトディレクトリ以外のディレクトリにスキーマファイルを保存する場合は、スキーマファイルを保存するカスタムディレクトリを指定できます。初期ロードの場合、便利になるように、以前に使用した値を使用できる場合はそれがドロップダウンリストに表示されます。このフィールドはオプションです。 初期ロードの場合、デフォルトでは、スキーマはデータディレクトリに保存されます。増分ロード、および初期ロードと増分ロードの組み合わせの場合、スキーマファイルのデフォルトディレクトリは{TaskTargetDirectory}/data/{TableName}/schema です。 **[データディレクトリ]** フィールドと同じプレースホルダを使用できます。プレースホルダは必ず中かっこ{}で囲んでください。 toUpper または toLower 関数を含める場合は、{toLower(SchemaName)}のように、プレースホルダ名を丸かっこで囲み、関数とプレースホルダの両方を中かっこで囲みます。 **注**: スキーマは、CSV 形式で出力データファイルにのみ書き込まれます。Parquet および Avro 形式のデータファイルには、独自の埋め込みスキーマが含まれています。
サイクル完了ディレクトリ	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、サイクル完了ファイルを含むディレクトリへのパス。デフォルトは{TaskTargetDirectory}/cycle/completed です。
サイクルコンテンツディレクトリ	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、サイクルコンテンツファイルを含むディレクトリへのパス。デフォルトは{TaskTargetDirectory}/cycle/contents です。
データディレクトリにサイクルのパーティション化を使用する	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、各データディレクトリの下に、CDC サイクルごとにタイムスタンプサブディレクトリが作成されます。 このオプションが選択されていない場合、別のディレクトリ構造を定義しない限り、個々のデータファイルがタイムスタンプなしで同じディレクトリに書き込まれます。
サマリディレクトリにサイクルのパーティション化を使用する	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合は、サマリコンテンツサブディレクトリおよび完了サブディレクトリの下に CDC サイクルごとにタイムスタンプサブディレクトリが作成されます。
コンテンツ内の個々のファイルを一覧表示する	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合は、コンテンツサブディレクトリの下に個々のデータファイルが一覧表示されます。 **[サマリディレクトリにサイクルのパーティション化を使用する]** がオフの場合は、このオプションがデフォルトでオンになります。タイムスタンプや日付などのプレースホルダを使用してカスタムサブディレクトリを設定できる場合を除き、コンテンツサブディレクトリ内の個々のファイルがすべて一覧表示されます。 **[データディレクトリにサイクルのパーティション化を使用する]** が選択されている場合でも、必要に応じてこのチェックボックスを選択して、個々のファイルを一覧表示し、CDC サイクルごとにグループ化することができます。

[詳細] で次のような詳細ターゲットプロパティを入力できます。これらは主に増分ロードジョブと組み合わせロードジョブに関係するものです。

フィールド	説明
操作タイプの追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイプを記録するメタデータカラムを追加するには、このチェックボックスを選択します。 増分ロードの場合、ジョブは、挿入を表す「I」、更新を表す「U」、または削除を表す「D」を書き込みます。初期ロードの場合、ジョブは常に、挿入を表す「I」を書き込みます。 デフォルトでは、このチェック ボックスは、増分ロードジョブ、初期および増分ロードジョブの場合はオンになっており、初期ロードジョブの場合はオフになっています。
操作時間の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスをオンにします。 初期ロードの場合、ジョブは常に現在の日付と時刻を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。
操作所有者の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作の所有者を記録するメタデータカラムを追加するには、このチェックボックスを選択します。 初期ロードの場合、ジョブは常に所有者として「INFA」を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。 **注**: このプロパティは、MongoDB ソースまたは PostgreSQL ソースを持つジョブでは使用できません。
操作トランザクション ID の追加	ジョブが SQL 操作のターゲットにプロパゲートする出力にソーストランザクション ID を含むメタデータカラムを追加するには、このチェックボックスを選択します。 初期ロードの場合、ジョブは常に ID として「1」を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。
前のイメージを追加	ジョブがターゲットに書き込む出力に UNDO データを含めるには、このチェックボックスを選択します。 初期ロードの場合、ジョブは null を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。

Microsoft Azure Synapse Analytics ターゲットのプロパティ

データベース統合タスクを定義する場合は、Microsoft Azure Synapse Analytics ターゲットのいくつかのプロパティをタスクウィザードの **[ターゲット]** ページで入力する必要があります。

これらのプロパティは、初期ロード、増分ロード、および初期ロードと増分ロードの組み合わせ操作に適用されます。

次の表は、**[ターゲット]** に表示されるターゲットのプロパティについて説明しています。

プロパティ	説明
ターゲット作成	利用可能なただ 1 つのオプションは、**[ターゲットテーブルを作成する]** であり、これによりソーステーブルをベースにしてターゲットテーブルを生成します。 **注**: ターゲットテーブルが作成された後、一括取り込みデータベースは、後続のジョブ実行でターゲットテーブルをインテリジェントに処理します。一括取り込みデータベースは、特定の状況に応じて、ターゲットテーブルを切り詰めたり再作成したりする場合があります。
スキーマ	一括取り込みデータベースがターゲットテーブルを作成するターゲットスキーマを選択します。接続プロパティで指定されたスキーマ名がデフォルトで表示されます。このフィールドでは大文字と小文字が区別されるため、接続プロパティのスキーマ名が適切な大文字小文字表記で入力されていることを確認してください。

次の表は、**[詳細]** に表示されるターゲットの詳細プロパティについて説明しています。

プロパティ	説明
最終レプリケート時刻を追加	ターゲットテーブルでレコードが挿入または最後に更新された時点のタイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスをオンにします。初期ロードでは、ロードされたすべてのレコードのタイムスタンプが同じになります。増分ロード、および初期ロードと増分ロードの組み合わせの場合、このカラムには、ターゲットに適用された最後の DML 操作のタイムスタンプが記録されます。 デフォルトでは、このチェックボックスは選択されていません。
メタデータカラムのプレフィックス	追加されたメタデータカラムの名前にプレフィックスを追加し、それらを簡単に識別して、既存のカラムの名前との競合を防ぐことができるようにします。 プレフィックスには特殊文字を含めないようにしてください。特殊文字を含めた場合、タスクのデプロイメントが失敗します。 デフォルト値は INFA_です。

Microsoft Fabric OneLake ターゲットプロパティ

Microsoft Fabric OneLake ターゲットを持つデータベース統合タスクを定義する場合、タスクウィザードの **[ターゲット]** ページでいくつかのターゲットプロパティを入力する必要があります。

[ターゲット] では、次の Microsoft Fabric OneLake ターゲットのプロパティを入力できます。

プロパティ	説明
出力形式	出力ファイルの形式を選択します。次のオプションがあります。 - CSV - AVRO - PARQUET デフォルトの値は【CSV】です。 **注**: CSV 形式の出力ファイルでは、各フィールドの区切り文字として二重引用符（""）が使用されます。
CSV ファイルへのヘッダーの追加	【CSV】が出力形式として選択されている場合は、このチェックボックスをオンにして、ソース列名を含むヘッダーを出力 CSV ファイルに追加します。

プロパティ	説明
Avro 形式	出力形式として［AVRO］を選択した場合、ソーステーブルごとに作成される Avro スキーマの形式を選択します。次のオプションがあります。 - Avro-Flat。すべての Avro フィールドを 1 つのレコードに一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Generic。ソーステーブルのすべてのカラムを Avro フィールドの単一の配列に一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Nested。各タイプの情報を個別のレコードに編成する場合に、この Avro スキーマ形式を使用します。 デフォルト値は［Avro-Flat］です。
Avro シリアル化形式	出力形式として［AVRO］が選択されている場合は、Avro 出力ファイルのシリアル化形式を選択します。次のオプションがあります。 - **なし** - Binary - JSON デフォルト値は［Binary］です。
Avro スキーマディレクトリ	出力形式として［AVRO］が選択されている場合は、一括取り込みデータベースが各ソーステーブルの Avro スキーマ定義を格納しているローカルディレクトリを指定します。スキーマ定義ファイルには、次の命名パターンがあります。 *schemaname_tablename*.txt **注**: このディレクトリが指定されていない場合、Avro スキーマ定義ファイルは作成されません。
ファイル圧縮タイプ	CSV または AVRO 出力形式の出力ファイルのファイル圧縮タイプを選択します。次のオプションがあります。 - **なし** - Deflate - Gzip - Snappy デフォルト値は［**なし**］、これは圧縮が使用されないことを意味します。
Avro 圧縮タイプ	［AVRO］が出力形式としてが選択されている場合は、Avro 圧縮タイプを選択します。次のオプションがあります。 - **なし** - Bzip2 - Deflate - Snappy デフォルト値は［**なし**］、これは圧縮が使用されないことを意味します。
Parquet 圧縮タイプ	［PARQUET］出力形式が選択されている場合、Parquet でサポートされている圧縮タイプを選択できます。次のオプションがあります。 - **なし** - Gzip - Snappy デフォルト値は［**なし**］、これは圧縮が使用されないことを意味します。
Deflate 圧縮レベル	［Deflate］が［Avro **圧縮タイプ**］フィールドで選択されている場合、圧縮レベルとして 0 ～9 を指定します。デフォルトは 0 です。

プロパティ	説明

データディレクトリ	一括取り込みデータベースが出力データファイルとオプションでスキーマを保存するディレクトリのディレクトリ構造を定義します。ディレクトリパターンを定義するには、次のタイプのエントリを使用できます。
	- プレースホルダ{SchemaName}、{TableName}、{Timestamp}、{YY}、{YYYY}、{MM}、および{DD}（{YY}、{YYYY}、{MM}、および{DD}は日付要素です）。{Timestamp}値の形式はyyyymmdd_hhmissms です。ディレクトリパスに生成された日付と時刻は、初期ロードジョブがターゲットへのデータの転送を開始した日付と時刻を表します。
	- 特定のディレクトリ名。
	- toUpper()および toLower()関数。これは、関連付けられた(*placeholder*)の値を強制的に大文字または小文字に変換します。
	注: プレースホルダの値の大文字と小文字は区別されません。
	例:
	`myDir1/{SchemaName}/{TableName}`
	`myDir1/myDir2/{SchemaName}/{YYYY}/{MM}/{TableName}_{Timestamp}`
	`myDir1/{toLower(SchemaName)}/{TableName}_{Timestamp}`
	デフォルトのディレクトリパターンは{TableName)_{Timestamp}です。

スキーマディレクトリ	デフォルトディレクトリ以外のディレクトリにスキーマファイルを保存する場合は、スキーマファイルを保存するカスタムディレクトリを指定できます。初期ロードの場合、便利になるように、以前に使用した値を使用できる場合はそれがドロップダウンリストに表示されます。このフィールドはオプションです。 初期ロードの場合、デフォルトでは、スキーマはデータディレクトリに保存されます。増分ロード、および初期ロードと増分ロードの組み合わせの場合、スキーマファイルのデフォルトディレクトリは{TaskTargetDirectory}/data/{TableName}/schema です。 **[データディレクトリ]** フィールドと同じプレースホルダを使用できます。プレースホルダは必ず中かっこ{}で囲んでください。 toUpper または toLower 関数を含める場合は、{toLower(SchemaName)}のように、プレースホルダ名を丸かっこで囲み、関数とプレースホルダの両方を中かっこで囲みます。 **注**: スキーマは、CSV 形式で出力データファイルにのみ書き込まれます。Parquet および Avro 形式のデータファイルには、独自の埋め込みスキーマが含まれています。

Microsoft SQL Server ターゲットのプロパティ

データベース統合タスクを定義する場合は、Microsoft SQL Server ターゲットのいくつかのプロパティをタスクウィザードの **[ターゲット]** ページで入力する必要があります。

次の表は、**[ターゲット]** に表示されるターゲットのプロパティについて説明しています。

プロパティ	説明
ターゲット作成	利用可能なただ 1 つのオプションは、**[ターゲットテーブルを作成する]** であり、これによりソーステーブルをベースにしてターゲットテーブルを生成します。 **注**: ターゲットテーブルが作成された後、一括取り込みデータベースは、後続のジョブ実行でターゲットテーブルをインテリジェントに処理します。一括取り込みデータベースは、特定の状況に応じて、ターゲットテーブルを切り詰めたり再作成したりする場合があります。
スキーマ	一括取り込みデータベースがターゲットテーブルを作成するターゲットスキーマを選択します。

次の表は、**[詳細]** に表示されるターゲットの詳細プロパティについて説明しています。

プロパティ	説明
最終レプリケート時刻を追加	ターゲットテーブルでレコードが挿入または最後に更新された時点のタイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスをオンにします。初期ロードでは、ロードされたすべてのレコードのタイムスタンプが同じになります。増分ロード、および初期ロードと増分ロードの組み合わせの場合、このカラムには、ターゲットに適用された最後の DML 操作のタイムスタンプが記録されます。 デフォルトでは、このチェックボックスは選択されていません。
メタデータカラムのプレフィックス	追加されたメタデータカラムの名前にプレフィックスを追加し、それらを簡単に識別して、既存のカラムの名前との競合を防ぐことができるようにします。 プレフィックスには特殊文字を含めないようにしてください。特殊文字を含めた場合、タスクのデプロイメントが失敗します。 デフォルト値は INFA_です。

Oracle ターゲットのプロパティ

データベース統合タスクを定義する場合は、Oracle ターゲットのいくつかのプロパティをタスクウィザードの **[ターゲット]** ページで入力する必要があります。プロパティは、ロードタイプによってわずかに異なります。

次の表は、**[ターゲット]** に表示される Oracle ターゲットのプロパティについて説明しています。

プロパティ	説明
ターゲット作成	利用可能なただ 1 つのオプションは、**[ターゲットテーブルを作成する]** であり、これによりソーステーブルをベースにしてターゲットテーブルを生成します。 **注**: ターゲットテーブルが作成された後、一括取り込みデータベースは、後続のジョブ実行でターゲットテーブルをインテリジェントに処理します。一括取り込みデータベースは、特定の状況に応じて、ターゲットテーブルを切り詰めたり再作成したりする場合があります。
スキーマ	一括取り込みデータベースがターゲットテーブルを作成するターゲットスキーマを選択します。
適用モード	増分ロードジョブ、および初期ロードと増分ロードの組み合わせジョブの場合に、挿入、更新、削除といったソース DML の変更がターゲットにどのように適用されるかを示します。次のオプションがあります。 - **標準**。1 回の適用サイクルの間の変更を累積し、それをターゲットに適用する前に、より少ない SQL 文になるようにそれらをインテリジェントにマージします。例えば、ソース行で更新とそれに続く削除が発生した場合、ターゲットに行は適用されません。同じカラムまたはフィールドで複数の更新が発生した場合、最後の更新のみがターゲットに適用されます。異なるカラムまたはフィールドで複数の更新が発生した場合、更新はターゲットに適用される前に 1 つの更新レコードにマージされます。 - **監査**。ソーステーブルで行われたすべての DML 操作の監査証跡をターゲットに適用します。ソーステーブルの各 DML 変更の行が、**[詳細]** セクションで選択した監査カラムとともに、生成されたターゲットテーブルに書き込まれます。監査カラムには、DML 操作タイプ、時刻、所有者、トランザクション ID、生成された昇順シーケンス番号などの変更に関するメタデータが含まれています。監査履歴を使用して、データをターゲットデータベースに書き込む前にダウンストリームの計算または処理を実行する場合、またはキャプチャされた変更に関するメタデータを調べる場合は、監査適用モードの使用を検討してください。 デフォルト値は **[標準]** です。 **注**: タスクウィザードの **[ソース]** ページで手法として **[クエリベースの** CDC**]** を選択した場合、このフィールドは表示されません。

次の表は、**［適用モード］**を **［監査］**に設定している場合に **［詳細］**で設定できる詳細ターゲットプロパティについて説明します。

フィールド	説明
操作タイプの追加	ジョブがターゲットデータベースにプロパゲートする出力、またはターゲットテーブルに挿入する出力にソース SQL 操作タイプを記録するメタデータカラムを追加するには、このチェックボックスを選択します。 ジョブは、挿入を表す「I」、更新を表す「U」、または削除を表す「D」を書き込みます。 デフォルトでは、このチェックボックスは選択されています。
操作時間の追加	ジョブがターゲットテーブルにプロパゲートする出力にソース SQL 操作タイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスをオンにします。 デフォルトでは、このチェックボックスは選択されていません。
操作所有者の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作の所有者を記録するメタデータカラムを追加するには、このチェックボックスを選択します。 デフォルトでは、このチェックボックスは選択されていません。 **注**: このプロパティは、MongoDB ソースまたは PostgreSQL ソースを持つジョブでは使用できません。
操作トランザクション ID の追加	ジョブが SQL 操作のターゲットにプロパゲートする出力にソーストランザクション ID を含むメタデータカラムを追加するには、このチェックボックスを選択します。 デフォルトでは、このチェックボックスは選択されていません。
操作シーケンスの追加	ジョブがターゲットテーブルに挿入する変更操作ごとに、生成された昇順のシーケンス番号を記録するメタデータカラムを追加するには、このチェックボックスを選択します。シーケンス番号には、操作の変更ストリーム位置が反映されます。 デフォルトでは、このチェックボックスは選択されていません。
メタデータカラムのプレフィックス	追加されたメタデータカラムの名前にプレフィックスを追加し、それらを簡単に識別して、既存のカラムの名前との競合を防ぐことができるようにします。 デフォルト値は INFA_です。

Oracle Cloud Object Storage ターゲットのプロパティ

Oracle Cloud Object Storage ターゲットのあるデータベース統合タスクを定義する場合、タスクウィザードの［**ターゲット**］タブでターゲットのプロパティをいくつか入力する必要があります。

［**ターゲット**］では、次の Oracle Cloud Object Storage ターゲットのプロパティを入力できます。

プロパティ	説明
出力形式	出力ファイルの形式を選択します。次のオプションがあります。 - CSV - AVRO - PARQUET デフォルトの値は［CSV］です。 **注**: CSV 形式の出力ファイルでは、各フィールドの区切り文字として二重引用符（""）が使用されます。
CSV ファイルへのヘッダーの追加	［CSV］が出力形式として選択されている場合は、このチェックボックスをオンにして、ソース列名を含むヘッダーを出力 CSV ファイルに追加します。
Avro 形式	出力形式として［AVRO］を選択した場合、ソーステーブルごとに作成される Avro スキーマの形式を選択します。次のオプションがあります。 - Avro-Flat。すべての Avro フィールドを 1 つのレコードに一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Generic。ソーステーブルのすべてのカラムを Avro フィールドの単一の配列に一覧表示する場合に、この Avro スキーマ形式を使用します。 - Avro-Nested。各タイプの情報を個別のレコードに編成する場合に、この Avro スキーマ形式を使用します。 デフォルト値は［Avro-Flat］です。
Avro シリアル化形式	出力形式として［AVRO］が選択されている場合は、Avro 出力ファイルのシリアル化形式を選択します。次のオプションがあります。 - **なし** - Binary - JSON デフォルト値は［Binary］です。
Avro スキーマディレクトリ	出力形式として［AVRO］が選択されている場合は、一括取り込みデータベースが各ソーステーブルの Avro スキーマ定義を格納しているローカルディレクトリを指定します。スキーマ定義ファイルには、次の命名パターンがあります。 *schemaname_tablename*.txt **注**: このディレクトリが指定されていない場合、Avro スキーマ定義ファイルは作成されません。
ファイル圧縮タイプ	CSV または AVRO 出力形式の出力ファイルのファイル圧縮タイプを選択します。次のオプションがあります。 - **なし** - Deflate - Gzip - Snappy デフォルト値は［**なし**］、これは圧縮が使用されないことを意味します。

プロパティ	説明
Avro 圧縮タイプ	[AVRO] が出力形式としてが選択されている場合は、Avro 圧縮タイプを選択します。次のオプションがあります。 - **なし** - Bzip2 - Deflate - Snappy デフォルト値は **[なし]**、これは圧縮が使用されないことを意味します。
Parquet 圧縮タイプ	[PARQUET] 出力形式が選択されている場合、Parquet でサポートされている圧縮タイプを選択できます。次のオプションがあります。 - **なし** - Gzip - Snappy デフォルト値は **[なし]**、これは圧縮が使用されないことを意味します。
Deflate 圧縮レベル	[Deflate] が [Avro **圧縮タイプ**] フィールドで選択されている場合、圧縮レベルとして 0〜9 を指定します。デフォルトは 0 です。
タスクターゲットディレクトリ	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、出力データファイル、スキーマファイル、および CDC サイクルのコンテンツと完了ファイルを保持する他のディレクトリのルートディレクトリ。このフィールドを使用して、タスクのカスタムルートディレクトリを指定できます。**[親としての接続ディレクトリ]** オプションを有効にしている場合は、必要に応じて、接続プロパティで指定された親ディレクトリで使用するタスクターゲットディレクトリを指定できます。 このフィールドは、次のディレクトリフィールドのいずれかのパターンで {TaskTargetDirectory}プレースホルダが指定されている場合は必須です。
ディレクトリタグの追加	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、Hive パーティショニングの命名規則と互換性を持たせるために適用サイクルディレクトリの名前に「dt=」プレフィックスを追加するには、このチェックボックスをオンにします。このチェックボックスはデフォルトでオフになっています。
親としての接続ディレクトリ	ターゲット接続プロパティで指定されたディレクトリ値を、タスクターゲットプロパティで指定されたカスタムディレクトリパスの親ディレクトリとして使用するようにするには、このチェックボックスをオンにします。初期ロードタスクの場合、親ディレクトリは、**データディレクトリ**と**スキーマディレクトリ**で使用されます。増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、親ディレクトリは**データディレクトリ**、**スキーマディレクトリ**、**サイクル完了ディレクトリ**、および**サイクルコンテンツディレクトリ**で使用されます。 このチェックボックスはデフォルトで選択されています。オフにしたとき、初期ロードの場合は、**[データディレクトリ]** フィールドで出力ファイルへのフルパスを定義します。増分ロードの場合は、必要に応じて **[タスクターゲットディレクトリ]** でタスクのルートディレクトリを指定します。

プロパティ	説明

データディレクトリ	初期ロードタスクの場合、一括取り込みデータベースが出力データファイルとオプションでスキーマを保存するディレクトリのディレクトリ構造を定義します。ディレクトリパターンを定義するには、次のタイプのエントリを使用できます。
	- プレースホルダ{SchemaName}、{TableName}、{Timestamp}、{YY}、{YYYY}、{MM}、および{DD}（{YY}、{YYYY}、{MM}、および{DD}は日付要素です）。{Timestamp}値の形式は yyyymmdd_hhmissms です。ディレクトリパスに生成された日付と時刻は、初期ロードジョブがターゲットへのデータの転送を開始した日付と時刻を表します。
	- 特定のディレクトリ名。
	- toUpper()および toLower()関数。これは、関連付けられた(*placeholder*)の値を強制的に大文字または小文字に変換します。
	注: プレースホルダの値の大文字と小文字は区別されません。
	例:
	`myDir1/{SchemaName}/{TableName}`
	`myDir1/myDir2/{SchemaName}/{YYYY}/{MM}/{TableName}_{Timestamp}`
	`myDir1/{toLower(SchemaName)}/{TableName}_{Timestamp}`
	デフォルトのディレクトリパターンは{TableName}_{Timestamp}です。
	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合は、cdc-data データファイルを含むサブディレクトリへのカスタムパスを定義します。ディレクトリパターンを定義する場合は、次のタイプのエントリを使用します。
	- プレースホルダ{TaskTargetDirectory}、{SchemaName}、{TableName}、{Timestamp}、{YY}、{YYYY}、{MM}、および{DD}（{YY}、{YYYY}、{MM}、および{DD}は日付要素です）。{Timestamp}値の形式は yyyymmdd_hhmissms です。ディレクトリパスに生成された日付と時刻は、CDC サイクルが開始された日付と時刻を表します。
	toUpper または toLower 関数を含める場合は、前の例に示すように、プレースホルダ名を丸かっこで囲み、関数とプレースホルダの両方を中かっこで囲みます。
	- 特定のディレクトリ名。
	デフォルトのディレクトリパターンは{TaskTargetDirectory}/data/{TableName}/data です。
	注: Amazon S3、フラットファイル、Microsoft Azure Data Lake Storage Gen2、および Oracle Cloud Object Store ターゲットでは、**[親としての接続ディレクトリ]** が選択されている場合、一括取り込みデータベースは、ターゲット接続プロパティで指定されたディレクトリをデータディレクトリパスのルートとして使用します。Google Cloud Storage ターゲットの場合、一括取り込みデータベースは取り込みタスクのターゲットプロパティで指定した**バケット**名を使用します。

スキーマディレクトリ	デフォルトディレクトリ以外のディレクトリにスキーマファイルを保存する場合は、スキーマファイルを保存するカスタムディレクトリを指定できます。初期ロードの場合、便利になるように、以前に使用した値を使用できる場合はそれがドロップダウンリストに表示されます。このフィールドはオプションです。 初期ロードの場合、デフォルトでは、スキーマはデータディレクトリに保存されます。増分ロード、および初期ロードと増分ロードの組み合わせの場合、スキーマファイルのデフォルトディレクトリは{TaskTargetDirectory}/data/{TableName}/schema です。 **[データディレクトリ]** フィールドと同じプレースホルダを使用できます。プレースホルダは必ず中かっこ{}で囲んでください。 toUpper または toLower 関数を含める場合は、{toLower(SchemaName)}のように、プレースホルダ名を丸かっこで囲み、関数とプレースホルダの両方を中かっこで囲みます。 **注**: スキーマは、CSV 形式で出力データファイルにのみ書き込まれます。Parquet および Avro 形式のデータファイルには、独自の埋め込みスキーマが含まれています。
サイクル完了ディレクトリ	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、サイクル完了ファイルを含むディレクトリへのパス。デフォルトは{TaskTargetDirectory}/cycle/completed です。
サイクルコンテンツディレクトリ	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、サイクルコンテンツファイルを含むディレクトリへのパス。デフォルトは{TaskTargetDirectory}/cycle/contents です。

プロパティ	説明
データディレクトリにサイクルのパーティション化を使用する	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合、各データディレクトリの下に、CDC サイクルごとにタイムスタンプサブディレクトリが作成されます。 このオプションが選択されていない場合、別のディレクトリ構造を定義しない限り、個々のデータファイルがタイムスタンプなしで同じディレクトリに書き込まれます。
サマリディレクトリにサイクルのパーティション化を使用する	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合は、サマリコンテンツサブディレクトリおよび完了サブディレクトリの下に CDC サイクルごとにタイムスタンプサブディレクトリが作成されます。
コンテンツ内の個々のファイルを一覧表示する	増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクの場合は、コンテンツサブディレクトリの下に個々のデータファイルが一覧表示されます。 **[サマリディレクトリにサイクルのパーティション化を使用する]** がオフの場合は、このオプションがデフォルトでオンになります。タイムスタンプや日付などのプレースホルダを使用してカスタムサブディレクトリを設定できる場合を除き、コンテンツサブディレクトリ内の個々のファイルがすべて一覧表示されます。 **[データディレクトリにサイクルのパーティション化を使用する]** が選択されている場合でも、必要に応じてこのチェックボックスを選択して、個々のファイルを一覧表示し、CDC サイクルごとにグループ化することができます。

[詳細] で次のような詳細ターゲットプロパティを入力して、監査テーブルに記録された各削除操作または各 DML 変更にメタデータカラムを追加することができます。

フィールド	説明
操作タイプの追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイプを記録するメタデータカラムを追加するには、このチェックボックスを選択します。 増分ロードの場合、ジョブは、挿入を表す「I」、更新を表す「U」、または削除を表す「D」を書き込みます。初期ロードの場合、ジョブは常に、挿入を表す「I」を書き込みます。 デフォルトでは、このチェック ボックスは、増分ロードジョブ、初期および増分ロードジョブの場合はオンになっており、初期ロードジョブの場合はオフになっています。
操作時間の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作タイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスをオンにします。 初期ロードの場合、ジョブは常に現在の日付と時刻を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。
操作所有者の追加	ジョブがターゲットにプロパゲートする出力にソース SQL 操作の所有者を記録するメタデータカラムを追加するには、このチェックボックスを選択します。 初期ロードの場合、ジョブは常に所有者として「INFA」を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。 **注**: このプロパティは、MongoDB ソースまたは PostgreSQL ソースを持つジョブでは使用できません。

フィールド	説明
操作トランザクション ID の追加	ジョブが SQL 操作のターゲットにプロパゲートする出力にソーストランザクション ID を含むメタデータカラムを追加するには、このチェックボックスを選択します。 初期ロードの場合、ジョブは常に ID として「1」を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。
前のイメージを追加	ジョブがターゲットに書き込む出力に UNDO データを含めるには、このチェックボックスを選択します。 初期ロードの場合、ジョブは null を書き込みます。 デフォルトでは、このチェックボックスは選択されていません。

PostgreSQL ターゲットプロパティ

データベース統合タスクを定義する場合は、PostgreSQL ターゲットのいくつかのプロパティを、タスクウィザードの［**ターゲット**］ページで入力する必要があります。

次の表は、［**ターゲット**］に表示されるターゲットのプロパティについて説明しています。

プロパティ	説明
ターゲット作成	利用可能なただ 1 つのオプションは、［**ターゲットテーブルを作成する**］であり、これによりソーステーブルをベースにしてターゲットテーブルを生成します。 **注**: ターゲットテーブルが作成された後、一括取り込みデータベースは、後続のジョブ実行でターゲットテーブルをインテリジェントに処理します。一括取り込みデータベースは、特定の状況に応じて、ターゲットテーブルを切り詰めたり再作成したりする場合があります。
スキーマ	一括取り込みデータベースがターゲットテーブルを作成するターゲットスキーマを選択します。

Snowflake Data Cloud ターゲットのプロパティ

データベース統合タスクを定義する場合は、Snowflake Data Cloud ターゲットのいくつかのプロパティをタスクウィザードの［**ターゲット**］ページで入力する必要があります。プロパティは、ロードタイプによってわずかに異なります。

次の表は、［**ターゲット**］に表示される Snowflake ターゲットのプロパティについて説明しています。

プロパティ	説明
ターゲット作成	利用可能なただ 1 つのオプションは、［**ターゲットテーブルを作成する**］であり、これによりソーステーブルをベースにしてターゲットテーブルを生成します。 **注**: ターゲットテーブルが作成された後、一括取り込みデータベースは、後続のジョブ実行でターゲットテーブルをインテリジェントに処理します。一括取り込みデータベースは、特定の状況に応じて、ターゲットテーブルを切り詰めたり再作成したりする場合があります。
スキーマ	一括取り込みデータベースがターゲットテーブルを作成するターゲットスキーマを選択します。

プロパティ	説明
ステージ	データがターゲットテーブルに書き込まれる前にソースから読み取られたデータを保持する内部ステージング領域の名前。この名前にスペースを含めることはできません。指定されたステージング領域が存在しない場合、自動的に作成されます。
適用モード	増分ロードジョブ、および初期ロードと増分ロードの組み合わせジョブの場合に、挿入、更新、削除といったソース DML の変更がターゲットにどのように適用されるかを示します。次のオプションがあります。 - **標準**。1 回の適用サイクルの間の変更を累積し、それをターゲットに適用する前に、より少ない SQL 文になるようにそれらをインテリジェントにマージします。例えば、ソース行で更新とそれに続く削除が発生した場合、ターゲットに行は適用されません。同じカラムまたはフィールドで複数の更新が発生した場合、最後の更新のみがターゲットに適用されます。異なるカラムまたはフィールドで複数の更新が発生した場合、更新はターゲットに適用される前に 1 つの更新レコードにマージされます。 - **論理削除**。ソース削除操作を論理削除としてターゲットに適用します。論理削除では、削除された行をデータベースから実際には削除せずに、削除済みとしてマークします。例えば、ソースで削除を行うと、ターゲットの変更レコードの INFA_OPERATION_TYPE カラムに「D」が表示されます。 処理を完了するために論理的に削除されたデータが必要となる、長期にわたるビジネスプロセスがある場合、誤って削除したデータを復元する必要がある場合、または削除された値を監査目的で追跡する必要がある場合は、論理削除の使用を検討してください。 **注**: **［論理削除］**モードを使用する場合は、ソーステーブルのプライマリキーに対して更新を実行しないでください。そうしないと、ターゲットでデータ破損が発生する可能性があります。 - **監査**。ソーステーブルで実行されたすべての DML 操作の監査証跡をターゲットに適用します。ソーステーブルの各 DML 変更の行が、**［詳細］**セクションで選択した監査カラムとともに、生成されたターゲットテーブルに書き込まれます。監査カラムには、DML 操作タイプ、時刻、所有者、トランザクション ID、生成された昇順シーケンス番号、前のイメージなどの変更に関するメタデータが含まれています。監査履歴を使用して、データをターゲットデータベースに書き込む前にダウンストリームの計算または処理を実行する場合、またはキャプチャされた変更に関するメタデータを調べる場合は、監査適用モードの使用を検討してください。 デフォルト値は **［標準］**です。 **注**: タスクウィザードの **［ソース］**ページで手法として **［クエリベースの** CDC**］**を選択した場合、このフィールドは表示されません。

次の表は、**［詳細］**に表示されるターゲットの詳細プロパティについて説明しています。

プロパティ	説明
最終レプリケート時刻を追加	ターゲットテーブルでレコードが挿入または最後に更新された時点のタイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスをオンにします。初期ロードの場合、ロードされたすべてのレコードのタイムスタンプは同じですが、Superpipe オプションを使用する Snowflake ターゲットに限り、分と秒がわずかに異なる可能性があります。増分ロード、および初期ロードと増分ロードの組み合わせの場合、このカラムには、ターゲットに適用された最後の DML 操作のタイムスタンプが記録されます。 デフォルトでは、このチェックボックスは選択されていません。
操作タイプの追加	ジョブがターゲットデータベースにプロパゲートする出力、またはターゲットシステムの監査テーブルに挿入する出力にソース SQL 操作タイプを記録するメタデータカラムを追加するには、このチェックボックスを選択します。 このフィールドは、**［適用モード］**オプションが**［監査］**または**［論理削除］**に設定されている場合にのみ使用できます。 監査モードでは、ジョブはこのメタデータカラムに、挿入の場合は「I」、更新の場合は「U」、削除の場合は「D」を書き込みます。 論理削除モードでは、ジョブは削除の場合は「D」を書き込み、挿入と更新の場合は NULL を書き込みます。操作タイプが NULL の場合、他の［操作の追加...］メタデータカラムも NULL です。操作タイプが「D」の場合にのみ、他のメタデータカラムに NULL 以外の値が含まれます。 デフォルトでは、このチェックボックスは選択されています。論理削除を使用している場合は、選択を解除できません。
操作時間の追加	ジョブがターゲットデータベースにプロパゲートする出力、またはターゲットシステムの監査テーブルに挿入する出力にソース SQL 操作のタイムスタンプを記録するメタデータカラムを追加するには、このチェックボックスを選択します。 このフィールドは、**［適用モード］**が**［監査］**または**［論理削除］**に設定されている場合にのみ使用できます。 デフォルトでは、このチェックボックスは選択されていません。
操作所有者の追加	ジョブがターゲットデータベースにプロパゲートする出力、またはターゲットシステムの監査テーブルに挿入する出力にソース SQL 操作の所有者を記録するメタデータカラムを追加するには、このチェックボックスを選択します。 このフィールドは、**［適用モード］**が**［監査］**または**［論理削除］**に設定されている場合にのみ使用できます。 デフォルトでは、このチェックボックスは選択されていません。 **注:** このプロパティは、MongoDB ソースまたは PostgreSQL ソースを持つジョブでは使用できません。
操作トランザクション ID の追加	ジョブが SQL 操作のターゲットにプロパゲートする出力にソーストランザクション ID を含むメタデータカラムを追加するには、このチェックボックスを選択します。 このフィールドは、**［適用モード］**が**［監査］**または**［論理削除］**に設定されている場合にのみ使用できます。 デフォルトでは、このチェックボックスは選択されていません。
操作シーケンスの追加	ジョブがターゲットシステムの監査テーブルに挿入する変更操作ごとに、生成された昇順のシーケンス番号を記録するメタデータカラムを追加するには、このチェックボックスを選択します。シーケンス番号には、操作の変更ストリーム位置が反映されます。 このフィールドは、**［適用モード］**が**［監査］**に設定されている場合にのみ使用できます。 デフォルトでは、このチェックボックスは選択されていません。

プロパティ	説明
前のイメージを追加	ジョブがターゲットテーブルに挿入する出力に UNDO の「操作前のイメージ」データを含む_OLD カラムを追加するには、このチェックボックスを選択します。これにより、各データカラムの以前の値と現在の値を比較できるようになります。削除操作の場合、現在の値は NULL になります。 このフィールドは、**[適用モード]** が **[監査]** に設定されている場合にのみ使用できます。 デフォルトでは、このチェックボックスは選択されていません。
メタデータカラムのプレフィックス	追加されたメタデータカラムの名前にプレフィックスを追加し、それらを簡単に識別して、既存のカラムの名前との競合を防ぐことができるようにします。 デフォルト値は INFA_です。
Superpipe	最初にデータをステージファイルに書き込むのではなく、Snowpipe Streaming API を使用して、短い待ち時間で Snowflake Data Cloud ターゲットテーブルにデータ行を直接すばやくストリーミングするには、このチェックボックスを選択します。このオプションは、すべてのロードタイプで有効にすることができます。 ターゲット接続を設定するときに、KeyPair 認証を選択します。 デフォルトでは、このチェックボックスは選択されています。中間ステージファイルにデータを書き込む場合は、選択解除します。
マージ頻度	[Superpipe] が選択されている場合、必要に応じて、変更データ行がマージされて Snowflake ターゲットテーブルに適用される頻度を秒単位で設定できます。このフィールドは、増分ロードタスク、および初期ロードと増分ロードの組み合わせタスクに適用されます。有効な値は 300～604800 です。デフォルトは 3600 秒です。
大文字と小文字の変換を有効にする	デフォルトでは、ターゲットテーブル名およびカラム名は、対応するソース名と同じ大文字と小文字で生成されます。ただし、ターゲットのクラスタレベルまたはセッションレベルのプロパティがこの大文字と小文字を区別する動作をオーバーライドしている場合を除きます。ターゲット名の大文字と小文字を制御する場合は、このチェックボックスを選択します。次に、**[大文字と小文字の変換ストラテジ]** オプションを選択します。 **注**: このチェックボックスは、[Superpipe] オプションを選択した場合は使用できません。Snowflake の Superpipe オプションを使用している場合は、大文字と小文字の変換を有効にできません。
大文字と小文字の変換ストラテジ	**[大文字と小文字の変換を有効にする]** を選択した場合は、以下のいずれかのオプションを選択して、生成されたターゲットテーブル（またはオブジェクト）名およびカラム（またはフィールド）名の大文字と小文字の処理方法を指定します。 - **ソースと同じ**。ソーステーブル（またはオブジェクト）名およびカラム（またはフィールド）名と同じ大文字と小文字を使用します。 - UPPERCASE。すべて大文字を使用します。 - lowercase。すべて小文字を使用します。 デフォルト値は、**[ソースと同じ]** です。 **注**: 選択したストラテジは、大文字と小文字の制御に関するターゲットのクラスタレベルまたはセッションレベルのプロパティをオーバーライドします。

スケジュールとランタイムオプションの設定

データベース統合タスクウィザードの［**スケジュールおよびランタイムオプション**］ページでは、初期ロードジョブを定期的に実行するスケジュールを指定し、任意のロードタイプのジョブのランタイムオプションを設定できます。

1. ［**詳細**］の下で、必要に応じて［**出力ファイルの行数**］の値を編集して、データベース統合タスクが出力データファイルに書き込む最大行数を指定します。

   注: Apache Kafka ターゲットを持つジョブの場合、詳細オプションは表示されません。

   増分ロード操作と、初期ロード操作と増分ロード操作の組み合わせの場合、この行数に達したとき、またはフラッシュ待ち時間が経過して、トランザクションの処理の途中でジョブが実行されない場合に、変更データがターゲットにフラッシュされます。フラッシュ待ち時間は、ジョブがデータをターゲットにフラッシュする前に、さらに変更データを待機する時間です。待ち時間は内部で 10 秒に設定されており、変更できません。

   有効な値は 1 から 100000000 です。Amazon S3、Microsoft Azure Data Lake Storage Gen2、および Oracle Cloud Infrastructure（OCI）Object Storage ターゲットのデフォルト値は 1000 行です。その他のターゲットの場合は、デフォルト値は 100000 行です。

   注: Microsoft Azure Synapse Analytics ターゲットの場合、データは最初に Microsoft Azure Data Lake Storage ステージングファイルに送信されてから、ターゲットテーブルに書き込まれます。データがターゲットに書き込まれた後、ステージングファイルを含むテーブル固有のディレクトリのコンテンツ全体が消去されます。Snowflake ターゲットの場合、データは最初に内部ステージ領域に格納されてから、ターゲットテーブルに書き込まれます。

2. 初期ロードジョブの場合のみ、フラットファイル、Amazon S3、Google Cloud Storage、Microsoft Azure Data Lake Storage、Microsoft Fabric OneLake、または Oracle Cloud Object Storage ターゲットの出力データファイルに.dat 拡張子を付ける場合は、必要に応じて［**ファイルタイプに基づくファイル拡張子**］チェックボックスをクリアします。このチェックボックスはデフォルトで選択されており、出力ファイルのファイルタイプに基づいてファイル名拡張子が付けられます。

   注: これらのターゲットタイプの増分ロードジョブの場合、このオプションは使用できません。一括取り込みデータベースは、常にファイルタイプに基づいて出力ファイル名拡張子を使用します。

3. Amazon S3、Google Cloud Storage、Microsoft Azure Data Lake Storage Gen2、または Oracle Cloud Object Storage ターゲットを持つデータベース取り込み増分ロードタスクの場合、次の適用サイクルオプションを設定します。

オプション	説明
サイクル間隔の適用	データベース取り込みジョブが適用サイクルを終了するまでに経過する必要のある時間を指定します。日、時間、分、秒を指定するか、これらの時間フィールドのサブセットに値を指定して、他のフィールドを空白のままにすることができます。 デフォルト値は 15 分です。
サイクル変更制限の適用	ジョブが適用サイクルを終了する前に処理する必要がある、データベース取り込みジョブのすべてのテーブル内のレコードの合計数を指定します。このレコード制限に達すると、データベース取り込みジョブは適用サイクルを終了し、変更データをターゲットに書き込みます。 デフォルト値は 10000 レコードです。 **注**: 起動中に、古いデータのバックログの処理に追いつく必要がある場合、ジョブは適用サイクル間隔よりも頻繁にこの制限に達する可能性があります。
低アクティビティのフラッシュ間隔	データベース取り込みジョブが適用サイクルを終了する前に、ソースで変更アクティビティがない期間中に経過する必要がある時間を時間、分、またはその両方で指定します。この時間制限に達すると、データベース取り込みジョブは適用サイクルを終了し、変更データをターゲットに書き込みます。 このオプションの値を指定しない場合、データベース取り込みジョブは、**［サイクル変更制限の適用］**または**［サイクル間隔の適用］**のいずれかの制限に達した後にのみ適用サイクルを終了します。 デフォルト値は指定されていません。

注意事項:

- **［サイクル間隔の適用］**または**［サイクル変更制限の適用］**フィールドのいずれかが、ゼロ以外の値であるか、デフォルト値を使用する必要があります。
- 適用サイクルは、ジョブが 3 つの制限のいずれかで、最初に満たされた制限に達すると終了します。

4. **［スキーマドリフトオプション］**で、スキーマドリフトの検出がソースとターゲットの組み合わせでサポートされている場合は、サポートされている各タイプの DDL 操作に使用するスキーマドリフトオプションを指定します。

   スキーマドリフトオプションは、次のソースとターゲットの組み合わせおよびロードタイプでサポートされています。

ソース	ロードタイプ	ターゲット
Db2 for i	差分 初期と増分の組み合わせ	Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Kafka（増分ロードのみ）、Microsoft Azure Data Lake Storage、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、または Snowflake
DB2 for LUW	差分 初期ロードと増分ロードの組み合わせ	Snowflake

ソース	ロードタイプ	ターゲット
DB2 for z/OS（Db2 11 を除く）	差分 初期と増分の組み合わせ	Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Kafka（増分ロードのみ）、Microsoft Azure Data Lake Storage、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、または Snowflake
Microsoft SQL Server	差分 初期と増分の組み合わせ	Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Kafka（増分ロードのみ）、Microsoft Azure Data Lake Storage、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、または Snowflake
Oracle	差分 初期と増分の組み合わせ	Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Kafka（増分ロードのみ）、Microsoft Azure Data Lake Storage、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、または Snowflake
PostgreSQL	差分 初期と増分の組み合わせ	増分ロード: Amazon Redshift、Amazon S3、Databricks Delta、Google BigQuery、Google Cloud Storage、Kafka（増分ロードのみ）、Microsoft Azure Data Lake Storage、Microsoft Azure Synapse Analytics、Oracle、Oracle Cloud Object Storage、または Snowflake 初期と増分の組み合わせ: Oracle または Snowflake

サポートされている DDL 操作のタイプは次のとおりです。

- カラムの追加
- カラムの変更
- カラムの削除
- カラム名の変更

注: ［カラムの変更］および［カラムの名前変更］オプションはサポートされておらず、Google BigQuery ターゲットを持つデータベース取り込みジョブでは表示されません。

次の表に、DDL 操作タイプに設定できるスキーマドリフトオプションを示します。

オプション	説明
無視	ソースデータベースで発生する DDL の変更をターゲットにレプリケートしません。Amazon Redshift、Kafka、Microsoft Azure Synapse Analytics、または Snowflake ターゲットの場合、このオプションは、［カラムの削除］および［カラム名の変更］操作タイプのデフォルトオプションです。 CSV 出力形式を使用する Amazon S3、Google Cloud Storage、Microsoft Azure Data Lake Storage、および Oracle Cloud Object Storage ターゲットの場合、**［無視］**オプションは無効です。AVRO 出力形式の場合、このオプションは有効になっています。
レプリケート	DDL 操作をターゲットにレプリケートします。Amazon S3、Google Cloud Storage、Microsoft Azure Data Lake Storage、および Oracle Cloud Object Storage ターゲットの場合、このオプションはすべての操作タイプのデフォルトオプションです。他のターゲットの場合、このオプションは、［カラムの追加］および［カラムの変更］操作タイプのデフォルトオプションです。 **制限**: - ターゲットでサポートされていないタイプのスキーマ変更をレプリケートしようとすると、そのタスクに関連付けられたデータベース取り込みジョブはエラーが発生して終了します。例えば、Microsoft Azure Synapse Analytics ターゲットでのカラム名の変更操作に **［レプリケート］**を選択すると、ジョブが終了します。 - プライマリキーカラムを追加するカラムの追加操作はサポートされておらず、予期しない結果を引き起こす可能性があります。 - Databricks Delta ターゲットの場合、**［レプリケート］**オプションは **［カラムの削除］**では使用できません。**［カラム名の変更］**オプションに **［レプリケート］**を指定した場合、タスクがデプロイされた後、ジョブが実行される前に、次の Databricks Delta テーブルプロパティを指定する必要があります: 'delta.minReaderVersion' = '2', 'delta.minReaderVersion' = '2', 'delta.minWriterVersion' = '5', and 'delta.columnMapping.mode' = 'name'。 - カラムの NULL 制約または NOT NULL 制約を変更するカラム変更操作は、仕様により、ターゲットにレプリケートされません。これは、ターゲットカラムの NULL 許容可否を変更すると、後続の変更が適用されるときに問題が発生する可能性があるためです。 - DDL 操作は、一部のデータ変更がキャプチャされた後にのみターゲットにレプリケートされます。
ジョブの停止	データベース取り込みジョブ全体を停止します。
テーブルの停止	DDL 変更が発生したソーステーブルの処理を停止します。1 つ以上のテーブルが **［テーブルの停止］**スキーマドリフトオプションによってレプリケーションから除外された場合、ジョブの状態が **［実行中(警告あり)］**に変わります。 **重要**: データベース取り込みジョブは、ジョブがソーステーブルの処理を停止した後にソーステーブルで発生したデータ変更を取得できません。その結果、ターゲットでデータ損失が発生する可能性があります。データの損失を回避するには、ジョブが処理を停止したソースオブジェクトとターゲットオブジェクトを再同期する必要があります。**［オプションで再開］** > **［再同期］**オプションを使用します。詳細については、「データベース取り込みジョブの再開時のスキーマドリフトオプションのオーバーライド」 (ページ 165)を参照してください。

5. Apache Kafka ターゲットを持つ増分ロードジョブの場合、次のチェックポイントオプションを設定します。

オプション	説明
チェックポイントのすべての行	データベース取り込みジョブが、Kafka ターゲットに送信されるすべてのメッセージに対してチェックポイント処理を実行するかどうかを示します。 **注**: このチェックボックスが選択されている場合、**［チェックポイントすべてのコミット］**、**［チェックポイントの行数］**、および **［チェックポイントの頻度(秒)］** オプションは無視されます。
チェックポイントのすべてのコミット	データベース取り込みジョブが、ソースで発生するすべてのコミットに対してチェックポイント処理を実行するかどうかを示します。
チェックポイントの行数	チェックポイントを追加する前に、データベース取り込みジョブがターゲットに送信するメッセージの最大数を指定します。このオプションを 0 に設定すると、データベース取り込みジョブはメッセージの数に基づいてチェックポイント処理を実行しません。このオプションを 1 に設定すると、データベース取り込みジョブは各メッセージにチェックポイントを追加します。
チェックポイントの頻度(秒)	データベース取り込みジョブがチェックポイントを追加するまでに経過する必要がある最大秒数を指定します。このオプションを 0 に設定すると、データベース取り込みジョブは経過時間に基づいてチェックポイント処理を実行しません。

6. 監視インタフェースの 1 つからデプロイされた後にジョブを手動で開始するのではなく、既存のスケジュールに基づいて初期ロードタスクのジョブインスタンスを実行する場合は、**［スケジュール］** で **［このタスクは指定したスケジュールを使用する］** を選択して、事前定義されたスケジュールを選択します。デフォルトのオプションは **［このタスクはスケジュールを使用しない］** です。

   このフィールドは、増分ロードおよび初期ロードタスクと増分ロードタスクの組み合わせには使用できません。

   Administrator でスケジュールオプションを表示および編集できます。スケジュールを編集すると、変更はスケジュールを使用するすべてのジョブに適用されます。タスクのデプロイ後にスケジュールを編集する場合、タスクを再デプロイする必要はありません。

   ジョブを実行するためのスケジュール条件が満たされていて、前のジョブ実行がまだアクティブである場合、一括取り込みデータベースは新しいジョブの実行をスキップします。

7. **［カスタムプロパティ］** で、特別な要件を満たすために Informatica が提供するカスタムプロパティを指定できます。プロパティを追加するには、**［プロパティの作成］** フィールドに、プロパティの名前と値を入力します。次に、**［プロパティの追加］** をクリックします。

   これらのプロパティを指定する場合は、Informatica グローバルカスタマサポートにお問い合わせください。通常、これらのプロパティは、固有の環境または特別な処理のニーズに対応します。必要に応じて、複数のプロパティを指定できます。プロパティ名には、英数字と次の特殊文字のみを含めることができます: ピリオド（.）、ハイフン（-）、およびアンダースコア（_）。

   ヒント: プロパティを削除するには、リストのプロパティ行の右端にある［削除］アイコンボタンをクリックします。

8. **［保存］** をクリックします。

データベース取り込みタスクのデプロイ

データベース統合タスクを定義して保存した後、Secure Agent とデータベース一括取り込みエージェントサービスおよび DBMI パッケージを含むオンプレミスシステム上にタスクをデプロイして、実行可能なジョブインスタンスを作成します。ジョブを実行する前に、タスクをデプロイする必要があります。デプロイプロセスは、タスク定義も検証します。

Microsoft Azure Synapse Analytics または Snowflake ターゲットを持つタスクをデプロイする前に、ソースカラムの追加や削除、カラムの NULL 制約やデータ型の変更が行われたことが原因でソーステーブルの構造と一致していない既存のターゲットテーブルを削除します。タスクをデプロイすると、最新のソース構造に基づいてターゲットテーブルが生成されます。

▶ タスクをデプロイするには、データベース統合タスクウィザードで、完了したタスク定義を保存し、**［デプロイ］**をクリックします。

タスクを正常にデプロイすると、関連付けられたジョブインスタンスは［デプロイ済み］状態になります。一括取り込みの**［マイジョブ］**ページ、またはオペレーションインサイトの**［一括取り込み］**ページの**［すべてのジョブ］**タブのいずれかからタスクを実行できます。

デプロイに関する考慮事項:

- データベース取り込みタスク名にスペースを含めた場合、生成されたジョブインスタンスの対応するジョブ名からスペースが省略されます。
- デプロイプロセスが失敗した場合、ジョブのステータスは「失敗」に切り替わります。その後、ジョブをデプロイ解除できます。エラーを診断するには、一括取り込みの**［マイジョブ］**ページ、またはオペレーションインサイトの**［一括取り込み］**ページの**［すべてのジョブ］**タブからエラーログをダウンロードします。ジョブの［アクション］メニューで、**［エラーログ］**をクリックします。問題を解決したら、データベース統合タスクウィザードからタスクを再度デプロイします。
- ジョブをデプロイ解除してから、関連付けられた取り込みタスクのジョブを再度実行する場合は、タスクを再度デプロイして、新しいジョブインスタンスを作成する必要があります。新しいジョブインスタンス名は、形式 *taskname-job_instance_number* の増分番号で終わります。ジョブインスタンス番号は、すべての取り込みジョブの最大インスタンス数に 1 を加算することにより、取り込みタスクをデプロイするたびに増加します。
- タスクのデプロイ中に Secure Agent を再起動すると、ジョブのステータスが「失敗」に切り替わります。タスクのデプロイ中に Secure Agent を再起動しないでください。
- タスクが「デプロイ中」状態でハングしているように見える場合は、Secure Agent を再起動します。関連付けられたジョブインスタンスは、「失敗」のステータスになります。デプロイ解除してから、再度デプロイできます。

データベース取り込みジョブの実行

デプロイされたデータベース取り込みジョブは、監視インタフェースのいずれかから実行できます。または、データベース取り込みの初期ロードタスクを作成するときに、タスクに関連付けられたジョブインスタンスを実行するためのスケジュールを指定できます。

デプロイ済みデータベース統合ジョブの実行

以前にデプロイされ［デプロイされていない］以外の状態にあるデータベース統合ジョブは、一括取り込みサービスの**［マイジョブ］**ページまたは、オペレーションインサイトの**［一括取り込み］**ページの**［すべてのジョブ］**タブから実行することができます。

1. 実行するジョブの行に移動します。
2. 行の［アクション］メニューで、**［実行］**をクリックします。

   サブタスクは、ソーステーブルごとに開始されます。

   注意事項:

   - 初期ロードジョブと増分ロードジョブを組み合わせた初期ロード部分が、ソーステーブルからターゲットテーブルへのデータのロードに失敗した場合、データベース統合ジョブは、テーブルのサブタスクを最大 3 回再試行します。再試行の間隔は最低 60 秒です。すべての初期ロード再試行が失敗した場合、サブタスクは**［エラー］**の状態を取得し、テーブルはレプリケーションから除外されます。その後、ジョブは増分ロードを続行しようとします。この場合、ジョブのステータスは**［実行中(警告あり)］**に変わります。
   - 初期ロードジョブがソーステーブルとターゲットテーブルのカラム定義間の不整合を検出した場合、ジョブはターゲットテーブルを削除し、ソースデータをターゲットにロードする前に、ソーステーブルと一致するように再作成します。
   - 初期ロードタスクと複合タスクの場合、ソーステーブルに多くの行が含まれていると、初期ロードの実行に時間がかかることがあります。

スケジュールに基づいた初期ロードジョブの実行

データベース統合の初期ロードタスクを設定する場合、タスクに関連付けられたジョブインスタンスを実行するためのスケジュールを指定できます。Administrator で事前にスケジュールを定義しておく必要があります。

1. スケジュールが存在しない場合は管理者で作成します。

   スケジュールオプションを設定して、特定のタイムゾーンに基づいて特定の日時に 10 秒を加えてジョブインスタンスを開始し、ジョブインスタンスを繰り返し実行することができます。
2. データベース統合の初期ロードタスクを一括取り込みで設定する場合、**［スケジュールとランタイムオプション］**ページで、**［このタスクは指定したスケジュールを使用する］**を選択して、スケジュールを選択します。

詳細については、Administrator ヘルプの「スケジューリング」を参照してください。

データベース取り込みジョブの管理

データベースタスクを設定して実行した後、ジョブの停止、再開、デプロイ解除、または再デプロイなどのジョブ管理タスクを実行することが必要になる場合があります。

データベース統合ジョブの停止

すべてのロードタイプのデータベース統合ジョブのうち、**［稼働中］**、**［警告付きで実行］**、 また **［保留］**状態にあるものは停止できます。

増分ロードジョブの場合、チェックポイントが作成された後にジョブが停止します。チェックポイントは、リカバリの目的で、増分処理が中断された変更ストリーム内のポイントを記録します。

初期ロードジョブと増分ロードジョブの組み合わせの場合、実行中の初期ロードサブタスクは完了するまで実行することができ、実行されていない初期ロードサブタスクは現在の状態のままになります。ジョブの増分ロード部分では、ジョブが停止する前に、チェックポイントがチェックポイントファイルまたはターゲットリカバリテーブルに書き込まれます。デプロイ後の最初のジョブ実行中に、ジョブ内の少なくとも 1 つのテーブルに対して変更レコードが処理されていない限り、データベース取り込みジョブではチェックポイントを記録できません。チェックポイントを使用できない場合、ジョブは設定された再開ポイントから処理を再開します。この再開ポイントは、デフォルトでは変更ストリーム内で利用可能な最新の位置です。

初期ロードジョブの場合、*実行中*のすべてのサブタスクは完了するまで実行することができ、その後ジョブが停止します。実行されていないサブタスクは現在の状態のままになります。

- ジョブの［アクション］メニューから、**［停止］**を選択します。

  ジョブの状態が**［停止中］**に切り替わり、次に**［停止］**に変わります。

  ヒント: 停止操作に時間がかかりすぎる場合は、ジョブを強制終了できます。

データベース統合ジョブの強制終了

［稼働中］、**［警告付きで実行］**、**［保留］**、 または **［停止］**状態のデータベース統合ジョブを強制終了できます。

増分ロードジョブの場合、チェックポイントが作成された直後にジョブが停止します。チェックポイントは、リカバリの目的で、増分処理が中断された変更ストリーム内のポイントを記録します。

初期ロードジョブと増分ロードジョブを組み合わせた場合、*実行中*の初期ロードサブタスクは直ちに停止します。ジョブの増分ロード部分では、チェックポイントが作成されてから、ジョブが停止します。

初期ロードジョブの場合、*実行中*のサブタスクは直ちに停止し、その後ジョブが停止します。実行されていないサブタスクは現在の状態のままになります。

- ジョブの［アクション］メニューから、**［強制終了］**を選択します。

  ジョブの状態が**［強制終了中］**に切り替わり、次に**［強制終了済み］**に変わります。

  初期ロードジョブの場合、開始済みのサブタスクと実行中のサブタスクの状態が**［強制終了済み］**に切り替わります。増分ロードまたは初期ロードジョブと増分ロードジョブの組み合わせの場合、サブタスクの状態は**［停止］**に切り替わります。

データベース統合ジョブの再開

停止状態、強制終了状態、または失敗状態のデータベース統合ジョブを再開できます。

一括取り込みサービスの**［マイジョブ］**ページまたはオペレーションインサイトの**［一括取り込み］**ページの**［すべてのジョブ］**タブのいずれかからジョブを再開できます。

複数のサブタスクを持つ初期ロードジョブを再開すると、一括取り込みデータベースは失敗、停止、強制終了、またはキューに格納の状態のサブタスクのみを開始します。

増分ロードジョブ、または初期ロードジョブと増分ロードジョブの組み合わせを再開すると、一括取り込みデータベースはソースデータの変更のレプリケートを、チェックポイントファイルまたはターゲットリカバリテーブルに記録された最後の位置から再開します。デプロイ後の最初のジョブ実行中に、少なくとも 1 つのテーブルに対して変更レコードが処理されていない限り、チェックポイントは使用できません。チェックポイントを使用できない場合、ジョブは設定された再開ポイントから処理を再開します。この再開ポイントは、デフォルトでは変更ストリーム内で利用可能な最新の位置です。

注: 初期ロードジョブの場合、**［実行］**コマンドも利用できる場合があります。取り込みジョブですべてのターゲットテーブルを切り詰めてから、ソースデータをターゲットテーブルに再ロードする場合、**［実行］**をクリックします。

1. 再開するジョブの行に移動します。

2. 行の［アクション］メニューで、**［再開］**をクリックします。

   注: タスクのデプロイメントが失敗したためにジョブが**［失敗］**状態になっている場合、**［再開］**コマンドは使用できません。

   サブタスクは、ソーステーブルごとに開始されます。

   エラーが発生した場合は、ページ上部にエラーメッセージが表示されます。

データベース取り込みジョブの再開時のスキーマドリフトオプションのオーバーライド

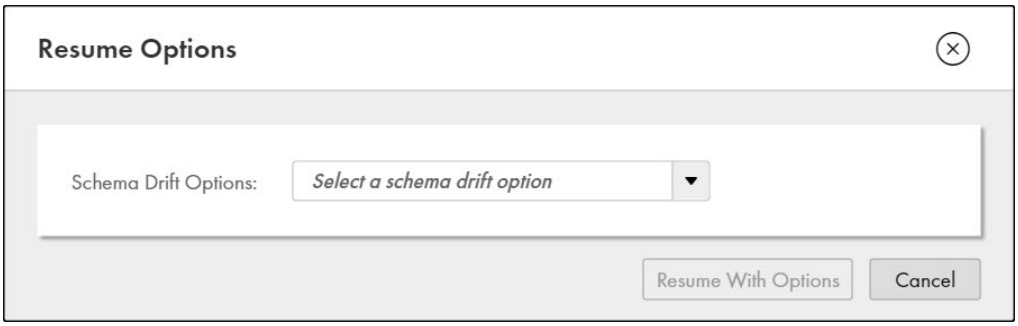
停止状態、強制終了状態、または失敗状態のデータベース統合ジョブを再開するときにスキーマドリフトオプションをオーバーライドできます。オーバーライドは、**［テーブルの停止］**または**［ジョブの停止］**スキーマドリフトオプションによって、現在エラー状態にあるテーブルにのみ影響します。オーバーライドを使用して、これらのエラーを修正または解決します。

スキーマドリフトオプションをオーバーライドして、一括取り込みサービスの**［マイジョブ］**ページまたはオペレーションインサイトの**［一括取り込み］**ページの**［すべてのジョブ］**タブから増分ロードジョブまたは初期ロードジョブと増分ロードジョブの組み合わせを再開できます。

1. オーバーライドを使用して再開するジョブの行に移動します。
2. 行の［アクション］メニューで、**［オプションで再開］**をクリックします。

   注: タスクのデプロイメントが失敗したためにジョブが**［失敗］**状態になっている場合、**［オプションで再開］**コマンドは使用できません。

   ［オプションの再開］ダイアログボックスが表示されます。



3. **［スキーマドリフトオプション］**リストで、データベース取り込みジョブを停止させたソースでの DDL 操作の処理に使用されるスキーマドリフトオプションを選択します。

   次の表に、スキーマドリフトのオプションを示します。

オプション	説明
無視	ソースデータベースで発生する DDL の変更をターゲットにレプリケートしません。
テーブルの停止	DDL 変更が発生したソーステーブルの処理を停止します。 **重要**: データベース取り込みジョブは、ジョブがソーステーブルの処理を停止した後にソーステーブルで発生したデータ変更を取得できません。その結果、ターゲットでデータ損失が発生する可能性があります。データの損失を回避するには、ジョブが処理を停止したソースオブジェクトとターゲットオブジェクトを再同期する必要があります。**［オプションで再開］** > **［再同期］**オプションを使用します。

オプション	説明
再同期	ターゲットテーブルをソーステーブルと再同期し、CDC 処理に使用したテーブル構造を保持します。このオプションは、**［スキーマドリフト］**オプションの**［テーブルの停止］**設定によって、ジョブが処理を停止したテーブルに使用します。 **重要**: このオプションは、初期ロードジョブと増分ロードジョブを組み合わせた場合にのみ使用できます。
再同期（更新）	Oracle または SQL Server ソースを持つデータベース取り込みの組み合わせロードジョブの場合、このオプションを使用して、スキーマドリフトで無視された DDL 変更を含め、ターゲットテーブルを最新のソーステーブル定義と再同期します。ターゲットテーブルが更新されると、ソーステーブルとターゲットテーブルの構造が一致します。このオプションは、**［再同期］**オプションの動作を模倣します。
再同期（保持）	Oracle または SQL Server ソースを持つデータベース取り込みの組み合わせロードジョブの場合は、このオプションを使用して、ソーステーブルとターゲットテーブルの現在の構造を保持したまま、CDC 用に処理されたのと同じカラムを再同期します。ソーステーブル定義またはターゲットテーブル定義の変更のチェックは行われません。ソース DDL の変更がソーステーブル構造に影響を与えた場合、それらの変更は処理されません。
レプリケート	データベース取り込みジョブが DDL 変更をターゲットにレプリケートできるようにします。 **重要**: Microsoft Azure Synapse Analytics ターゲットでのカラムの名前変更操作に**［レプリケート］**オプションを選択すると、エラーが発生してジョブが終了します。

4. **［オプションで再開］**をクリックします。

   再開されたジョブは、ステップ 3 で指定したスキーマドリフトオプションを使用して、ジョブを停止させる原因となったスキーマの変更を処理します。その後、タスクの作成時に指定したスキーマドリフトオプションが再び有効になります。

   重要: 一括取り込みデータベースがソーステーブルのスキーマ変更を処理するのは、テーブルで DML 操作が発生した後のみです。したがって、ジョブを再開した後、テーブルで最初の DML 操作が発生するまで、テーブルのサブタスクの状態は変更されません。

データベース取り込みジョブの再デプロイ

関連するデータベース統合タスクで利用可能なフィールドを編集した後にデータベース統合ジョブを再デプロイして、新しい設定を有効にできるようにします。

最初にジョブをデプロイ解除せずに、以前にデプロイされた取り込みタスク定義のすべてではなく一部のフィールドを編集できます。ソーステーブルを追加し、編集可能なランタイムオプションとターゲットオプションを変更できます。例えば、さまざまな設定の効果をテストするために、いくつかのターゲットオプションをリセットすることができます。

再デプロイ操作は、ソーステーブルの各ジョブサブタスクを停止し、更新された取り込みタスクをデプロイし、停止されたサブタスクと追加されたソーステーブルの新しいサブタスクを自動的に開始します。

1. **［マイジョブ］**ページで、再デプロイするジョブの行に移動します。
2. 行の［アクション］メニューで、**［再デプロイ］**を選択します。

   ジョブインスタンスは自動的に実行を開始します。
   ［再デプロイ］を選択したときにジョブが実行されていた場合、一括取り込みデータベースはジョブを停止してから、取り込みタスクを再デプロイし、ジョブを再開します。

注:

- 増分ロードジョブおよび初期ロードジョブと増分ロードジョブの組み合わせの場合、再デプロイ操作では、前回のデプロイメント中に作成された選択済みテーブルのリストは変更されません。テーブルのリストを更新するには、関連するタスクのテーブル選択ルールを編集してから、ジョブを再デプロイします。既存のテーブル選択ルールと一致するテーブルを追加した場合でも、テーブル選択ルールを更新する必要があります。
- データベース取り込みの初期ロードと増分ロードの組み合わせジョブでは、以前にカラムのサブセットを選択したソーステーブルに対して追加のカラムを選択し、ジョブを再デプロイすると、ジョブによって再同期操作がトリガされ、追加されたカラムのデータを取得してターゲットに書き込みます。テーブルのカラムを選択解除すると、再同期操作はトリガされません。代わりに、選択解除されたカラムは、スキーマドリフトの**［カラムの削除］**設定に基づいて処理されます。初期ロードジョブの場合、ジョブは次回のジョブ実行時に、追加されたカラムの内容をターゲットに書き込みます。
- 以前にカラムのサブセットを選択したテーブルの一部のカラムに DDL の変更を加えた後に、増分ロードジョブを再デプロイすると、ジョブは、タスクウィザードの**［スケジュールおよびランタイムオプション］**ページで設定したスキーマドリフトオプションに基づいて変更の処理を試みます。ただし、カラムの選択と DDL の変更が同時に行われると、正しい結果が得られない可能性があります。同じ状況で初期ロードと増分ロードの組み合わせタスクを再デプロイすると、再同期操作が自動的にトリガされ、ソースとターゲットの整合性が保たれます。
- Microsoft Azure Synapse Analytics または Snowflake Cloud Data Warehouse ターゲットを使用するジョブの場合、再デプロイ操作では、ターゲットテーブルが存在することも検証され、テーブル選択ルールが変更された場合は新しいテーブルが作成されます。

データベース統合ジョブのデプロイ解除

ジョブを実行する必要がなくなった場合、ジョブが失敗状態にある場合、または最初にジョブをデプロイ解除しないと編集できない関連タスクの接続またはプロパティを変更する必要がある場合は、データベース統合ジョブをデプロイ解除します。

ジョブをデプロイ解除する前に、ジョブが実行されていないことを確認してください。

ジョブがデプロイ解除された後は、ジョブを再度実行したり、再デプロイしたりすることはできません。関連付けられた取り込みタスクのジョブを再度実行する場合は、タスクウィザードからタスクを再度デプロイして、新しいジョブインスタンスを作成する必要があります。例えば、ターゲット接続を変更する場合は、ジョブをデプロイ解除し、取り込みタスクを編集して接続を変更し、タスクを再度デプロイしてから、新しいジョブインスタンスを実行します。

1. 一括取り込みの**［マイジョブ］**ページ、またはオペレーションインサイトの一括取り込みページの**［すべてのジョブ］**タブで、デプロイ解除するジョブの行に移動します。
2. 行の**［アクション］**メニューで、**［デプロイ解除］**をクリックします。

   デプロイ解除操作が失敗すると、ジョブのステータスは、［強制終了］状態であっても［失敗］に切り替わるか、［失敗］のままになります。

   注: ジョブをデプロイ解除した直後に、Secure Agent をシャットダウンしないようにしてください。一括取り込みデータベースが/root/infaagent/apps/Database_Ingestion/data/tasks ディレクトリ内のタスクのファイルをクリーンアップするまでにはしばらく時間がかかります。

ソースオブジェクトとターゲットオブジェクトの再同期

実行中のデータベース統合初期ロードジョブと増分ロードジョブの組み合わせの一部となっているサブタスクのソースオブジェクトとターゲットオブジェクトを再同期できます。サブタスクは、キューに格納または開始以外の状態である必要があります。

例えば、初期ロードまたは増分ロードの処理が失敗した場合、または特定の再起動ポイントからジョブを最初からやり直す場合は、ターゲットをソースと再同期することをお勧めします。

重要: **［テーブルの停止］**の**［スキーマドリフト］**設定によって停止して現在**［エラー］**状態のテーブルを再同期するには、［アクション］メニューの**［オプションで再開］** > **［再同期］**オプションを使用する必要があります。詳細については、「データベース取り込みジョブの再開時のスキーマドリフトオプションのオーバーライド」 (ページ 165)を参照してください。

1. 一括取り込みサービスの**［マイジョブ］**ページ、またはオペレーションインサイトの**［一括取り込み］**ページの**［すべてのジョブ］**タブで、取り込みジョブをドリルダウンすることにより、ジョブの詳細を表示します。

   ジョブは**［稼働中］**状態で、初期ロード操作と増分ロード操作を組み合わせたものである必要があります。
2. **［オブジェクトの詳細］**タブをクリックします。
3. 再同期するソースオブジェクトとターゲットオブジェクトのサブタスク行で、［アクション］メニューをクリックし、**［再同期］**を選択します。再同期操作では、CDC 用に処理されたソーステーブルとターゲットテーブルの構造が保持されます。

   注: ［アクション］メニューと**［再同期］**オプションを使用可能にするには、サブタスクがキューに格納または開始以外の状態である必要があります。

   Oracle または SQL Server ソースを持つデータベース取り込み組み合わせロードジョブのサブタスクを再同期する場合は、**［再同期］**オプションの代わりに次のいずれかの再同期オプションを使用します。

   - **再同期（更新）** - ターゲットテーブルを最新のソーステーブル定義（スキーマドリフトで無視された DDL 変更を含む）と再同期するには、このオプションを使用します。ターゲットテーブルが更新されると、ターゲットテーブルの構造は現行のソーステーブル構造と一致します。このオプションは、**［再同期］**オプションの動作を模倣します。
   - **再同期（保持）** - CDC 用に処理したのと同じカラムを再同期し、ソーステーブルとターゲットテーブルの現在の構造を保持するには、このオプションを使用します。ソーステーブル定義またはターゲットテーブル定義の変更のチェックは行われません。ソース DDL の変更がソーステーブル構造に影響を与えた場合、それらの変更は処理されません。

注:

- ソーステーブルに多くの行が含まれている場合、再同期の実行に長い時間がかかる可能性があります。
- ソーステーブルスキーマがターゲットテーブルスキーマと一致しない場合、取り込みサブタスクはターゲットテーブルを削除し、ソーススキーマと一致する新しいテーブルを作成します。ターゲットテーブルが再作成されるかどうかに関係なく、サブタスクはターゲットテーブルを切り詰めてから、ソースデータをテーブルに再ロードします。
- データベース取り込みサブタスクを Snowflake ターゲットと再同期し、監査適用モードを使用すると、監査情報を保持できます。一括取り込みでは、ターゲットテーブルが再作成され、監査情報を含む既存のテーブルの名前にタイムスタンプが付加されて<target_table_name>_<current_UTC_timestamp>形式の名前に変更されます。新しいターゲットテーブルに監査情報が必要な場合は、結合操作などでロードする必要があります。既存のテーブル名にタイムスタンプを追加することによって名前が最大文字数を超えると、サブタスクはエラーで失敗します。スキーマドリフトを有効にし、スキーマドリフトの変更（カラムの追加など）が発生した場合、新しいカラムは再作成されたターゲットテーブルには含まれますが、名前が変更されたテーブルには含まれません。この動作を有効にするには、タスクウィザードの**［ターゲット］**ページで backupTargetTableBeforeResync カスタムプロパティを **true** に設定します。

既存の監査情報がある組み合わせロードジョブを再同期する場合、以下の制限を考慮してください。

- ターゲットで既存のテーブルに監査情報を格納すると、データベースストレージが余分に消費されます。
- 監査情報の統合ビューを取得するには、ターゲットテーブルの複数のバージョンを結合する必要があります。

増分変更データ処理のための再開とリカバリ

一括取り込みデータベースは、変更データを失うことなく、エラーまたはユーザーによる停止要求によって停止した増分ロードと、初期ロードジョブと増分ロードジョブの組み合わせを再開できます。

最初のジョブの実行後、一括取り込みデータベースは変更がターゲットに適用されると、変更ストリーム内の処理位置の識別子を継続的に記録します。Amazon S3、Azure Data Lake Storage、Google Cloud Storage、Kafka、Oracle Cloud Object Storage などのファイルベースのターゲットの場合、識別子はチェックポイントファイルに保存されます。データベースターゲットの場合、識別子は、ターゲット上で INFORMATICA_CDC_RECOVERY と呼ばれる生成されたリカバリテーブルに格納されます。

注: 増分ロードジョブの最初の実行では、一括取り込みデータベースはデータベース取り込みタスクを定義するときに、**[増分ロードの当初の再開点]** フィールドで設定した開始点を使用します。

増分変更データ処理が異常終了した場合、またはユーザーによる停止要求または強制終了要求に応答して終了した場合にジョブを再開すると、そのジョブは、チェックポイントファイルまたはリカバリテーブルに保存された最後の位置から再開されます。デプロイ後の最初のジョブ実行中に、少なくとも 1 つのテーブルに対して変更レコードが処理されていない限り、チェックポイントは使用できません。チェックポイントを使用できない場合、ジョブは設定された再開ポイントから処理を再開します。この再開ポイントは、デフォルトでは変更ストリーム内で利用可能な最新の位置です。

デフォルトデータ型のマッピング

このリファレンスは、リレーショナルソースと Amazon Redshift、Databricks Delta、Google BigQuery、Microsoft Azure Synapse Analytics、Microsoft SQL Server、Oracle、PostgreSQL、および Snowflake ターゲットのデフォルトのデータ型マッピングを提供します。一括取り込みデータベースでは、ターゲットテーブルの生成時にこれらのマッピングを使用します。

ターゲットを設定する場合、必要に応じてデータ型マッピングルールを定義して、ソースデータ型からターゲットデータテーブルへのデフォルトのマッピングをカスタマイズできます。詳細については、「データ型マッピングのカスタマイズ」 (ページ 117)を参照してください。

ソースデータ型がリストされていない場合は、一括取り込みデータベースはこのデータ型のソースカラムからデータを抽出できないか、抽出したデータを適切なターゲットデータ型に適用できません。

DB2 for i ソースと Amazon Redshift ターゲット

次の表は、DB2 for i ソースと Amazon Redshift ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for i **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
bigint	bigint
binary(*size*), 1 <= size <= 32766	binary varying(*size*), 1 <= size <= 32766

DB2 for i **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
char(*size*) for bit data, 1 <= size <= 32766	binary varying(*size*), 1 <= size <= 32766
char(*size*), 1 <= size <= 1024	character(*size*), 4 <= size <= 4096
char(*size*), 1025 <= size <= 32766	character varying(*size*), 4100 <= size <= 65535
date	date
decfloat(*precision*), 16 <= p <= 34	character varying(255)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
decimal(p,s), 38 <= p <= 63, 0 <= s <= 62	character varying(*size*), 40 <= size <= 65
float	doubleprecision
integer	integer
long varbinary	binary varying(1024000)
long varchar	character varying(65535)
long varchar for bit data	binary varying(1024000)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
numeric(p,s), 38 <= p <= 63, 0 <= s <= 62	character varying(*size*), 40 <= size <= 65
real	real
rowid	binary varying(40)
smallint	smallint
time	time without time zone
timestamp(*precision*), 0 <= p <= 6	timestamp without time zone
timestamp(*precision*), 7 <= p <= 12	character varying(*size*), 27 <= size <= 32
varbinary(*size*), 1 <= size <= 32740	binary varying(*size*), 1 <= size <= 32740
varchar(*size*) for bit data, 1 <= size <= 32740	binary varying(*size*), 1 <= size <= 32740
varchar(*size*), 1 <= size <= 32740	character varying(*size*), 4 <= size <= 65535

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for i データ型はサポートされません。

- BLOB
- CLOB
- DATALINK

- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

DB2 for i ソースと Databricks Delta ターゲット

次の表は、DB2 for i ソースと Databricks Delta ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for i **ソースデータ型**	Databricks Delta **ターゲットデータ型**
bigint	long
binary(*size*), 1 <= size <= 32766	binary
char(*size*) for bit data, 1 <= size <= 32766	binary
char(*size*), 1 <= size <= 32766	string
date	string
decfloat(*precision*), 16 <= p <= 34	string
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
decimal(p,s), 39 <= p <= 63, 0 <= s <= 62	string
float	double
integer	integer
long varbinary	binary
long varchar	string
long varchar for bit data	binary
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 63, 0 <= s <= 62	string
real	float
rowid	binary
smallint	integer
time	string
timestamp(*precision*), 0 <= p <= 6	TIMESTAMP

DB2 for i **ソースデータ型**	Databricks Delta **ターゲットデータ型**
timestamp(*precision*), 7 <= p <= 12	string
varbinary(*size*), 1 <= s <= 32740	binary
varchar(*size*) for bit data, 1 <= size <= 32740	binary
varchar(*size*), 1 <= size <= 32740	string

サポートされていないソースデータ型
一括取り込みデータベースでは、次の DB2 for i データ型はサポートされません。

- Blob
- CLOB
- DATALINK
- 精度よりスケールが大きい DECIMAL または NUMERIC
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

DB2 for i ソースと Google BigQuery ターゲット

次の表は、DB2 for i ソースと Google BigQuery ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for i **ソースデータ型**	Google BigQuery **ターゲットデータ型**
bigint	int64
binary(*size*), 1 <= size <= 32766	bytes
char(*size*) for bit data, 1 <= size <= 32766	bytes
char(*size*), 1 <= size <= 32766	string
date	date
decfloat(*precision*), 16 <= p <= 34	string
decimal(1,1)	numeric
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric

DB2 for i **ソースデータ型**	Google BigQuery **ターゲットデータ型**
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(2,s), 1 <= s <= 2	numeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(3,s), 1 <= s <= 3	numeric
decimal(4,s), 1 <= s <= 4	numeric
decimal(5,s), 1 <= s <= 5	numeric
decimal(6,s), 1 <= s <= 6	numeric
decimal(7,s), 1 <= s <= 7	numeric
decimal(8,s), 1 <= s <= 8	numeric
decimal(p,0), 1 <= p <= 18	int64
decimal(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
decimal(p,s), 29 <= p <= 48, 0 <= s <= 29	bignumeric

DB2 for i **ソースデータ型**	Google BigQuery **ターゲットデータ型**
decimal(p,s), 30 <= p <= 49, 0 <= s <= 30	bignumeric
decimal(p,s), 31 <= p <= 50, 0 <= s <= 31	bignumeric
decimal(p,s), 32 <= p <= 51, 0 <= s <= 32	bignumeric
decimal(p,s), 33 <= p <= 52, 0 <= s <= 33	bignumeric
decimal(p,s), 34 <= p <= 53, 0 <= s <= 34	bignumeric
decimal(p,s), 35 <= p <= 54, 0 <= s <= 35	bignumeric
decimal(p,s), 36 <= p <= 55, 0 <= s <= 36	bignumeric
decimal(p,s), 37 <= p <= 56, 0 <= s <= 37	bignumeric
decimal(p,s), 38 <= p <= 56, 0 <= s <= 38	bignumeric
decimal(p,s), 39 <= p <= 40, 0 <= s <= 39	string
decimal(p,s), 39 <= p <= 40, 21 <= s <= 38	bignumeric
decimal(p,s), 40 <= p <= 41, 0 <= s <= 40	string
decimal(p,s), 41 <= p <= 42, 0 <= s <= 41	string
decimal(p,s), 42 <= p <= 43, 0 <= s <= 42	string
decimal(p,s), 43 <= p <= 44, 0 <= s <= 43	string
decimal(p,s), 44 <= p <= 63, 0 <= s <= 44	string
decimal(p,s), 45 <= p <= 46, 0 <= s <= 45	string
decimal(p,s), 46 <= p <= 47, 0 <= s <= 46	string
decimal(p,s), 47 <= p <= 48, 0 <= s <= 47	string
decimal(p,s), 48 <= p <= 49, 0 <= s <= 48	string
decimal(p,s), 49 <= p <= 50, 0 <= s <= 49	string
decimal(p,s), 50 <= p <= 51, 0 <= s <= 50	string
decimal(p,s), 51 <= p <= 52, 0 <= s <= 51	string
decimal(p,s), 52 <= p <= 53, 0 <= s <= 52	string
decimal(p,s), 53 <= p <= 54, 0 <= s <= 53	string
decimal(p,s), 54 <= p <= 55, 0 <= s <= 54	string
decimal(p,s), 55 <= p <= 56, 0 <= s <= 55	string

DB2 for i **ソースデータ型**	Google BigQuery **ターゲットデータ型**
decimal(p,s), 56 <= p <= 57, 0 <= s <= 56	string
decimal(p,s), 56 <= p <= 63, 39 <= s <= 62	string
decimal(p,s), 57 <= p <= 58, 0 <= s <= 57	string
decimal(p,s), 58 <= p <= 59, 0 <= s <= 58	string
decimal(p,s), 59 <= p <= 60, 0 <= s <= 59	string
decimal(p,s), 60 <= p <= 61, 0 <= s <= 60	string
decimal(p,s), 61 <= p <= 62, 0 <= s <= 61	string
decimal(p,s), 62 <= p <= 63, 0 <= s <= 62	string
decimal(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
float	float64
integer	int64
long varbinary	bytes
long varchar	string
long varchar for bit data	bytes
numeric(1,1)	numeric
numeric(10,10)	bignumeric
numeric(11,s), 10 <= s <= 11	bignumeric
numeric(12,s), 10 <= s <= 12	bignumeric
numeric(13,s), 10 <= s <= 13	bignumeric
numeric(14,s), 10 <= s <= 14	bignumeric
numeric(15,s), 10 <= s <= 15	bignumeric
numeric(16,s), 10 <= s <= 16	bignumeric
numeric(17,s), 10 <= s <= 17	bignumeric
numeric(18,s), 10 <= s <= 18	bignumeric
numeric(19,s), 10 <= s <= 19	bignumeric
numeric(2,s), 1 <= s <= 2	numeric
numeric(20,s), 10 <= s <= 20	bignumeric

DB2 for i **ソースデータ型**	Google BigQuery **ターゲットデータ型**
numeric(21,s), 10 <= s <= 21	bignumeric
numeric(22,s), 10 <= s <= 22	bignumeric
numeric(23,s), 10 <= s <= 23	bignumeric
numeric(24,s), 10 <= s <= 24	bignumeric
numeric(25,s), 10 <= s <= 25	bignumeric
numeric(26,s), 10 <= s <= 26	bignumeric
numeric(27,s), 10 <= s <= 27	bignumeric
numeric(28,s), 10 <= s <= 28	bignumeric
numeric(3,s), 1 <= s <= 3	numeric
numeric(4,s), 1 <= s <= 4	numeric
numeric(5,s), 1 <= s <= 5	numeric
numeric(6,s), 1 <= s <= 6	numeric
numeric(7,s), 1 <= s <= 7	numeric
numeric(8,s), 1 <= s <= 8	numeric
numeric(p,0), 1 <= p <= 18	int64
numeric(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
numeric(p,s), 29 <= p <= 48, 0 <= s <= 29	bignumeric
numeric(p,s), 30 <= p <= 49, 0 <= s <= 30	bignumeric
numeric(p,s), 31 <= p <= 50, 0 <= s <= 31	bignumeric
numeric(p,s), 32 <= p <= 51, 0 <= s <= 32	bignumeric
numeric(p,s), 33 <= p <= 52, 0 <= s <= 33	bignumeric
numeric(p,s), 34 <= p <= 53, 0 <= s <= 34	bignumeric
numeric(p,s), 35 <= p <= 54, 0 <= s <= 35	bignumeric
numeric(p,s), 36 <= p <= 55, 0 <= s <= 36	bignumeric
numeric(p,s), 37 <= p <= 56, 0 <= s <= 37	bignumeric
numeric(p,s), 38 <= p <= 56, 0 <= s <= 38	bignumeric
numeric(p,s), 39 <= p <= 40, 0 <= s <= 39	string

DB2 for i ソースデータ型	Google BigQuery ターゲットデータ型
numeric(p,s), 39 <= p <= 40, 21 <= s <= 38	bignumeric
numeric(p,s), 40 <= p <= 41, 0 <= s <= 40	string
numeric(p,s), 41 <= p <= 42, 0 <= s <= 41	string
numeric(p,s), 42 <= p <= 43, 0 <= s <= 42	string
numeric(p,s), 43 <= p <= 44, 0 <= s <= 43	string
numeric(p,s), 44 <= p <= 63, 0 <= s <= 44	string
numeric(p,s), 45 <= p <= 46, 0 <= s <= 45	string
numeric(p,s), 46 <= p <= 47, 0 <= s <= 46	string
numeric(p,s), 47 <= p <= 48, 0 <= s <= 47	string
numeric(p,s), 48 <= p <= 49, 0 <= s <= 48	string
numeric(p,s), 49 <= p <= 50, 0 <= s <= 49	string
numeric(p,s), 50 <= p <= 51, 0 <= s <= 50	string
numeric(p,s), 51 <= p <= 52, 0 <= s <= 51	string
numeric(p,s), 52 <= p <= 53, 0 <= s <= 52	string
numeric(p,s), 53 <= p <= 54, 0 <= s <= 53	string
numeric(p,s), 54 <= p <= 55, 0 <= s <= 54	string
numeric(p,s), 55 <= p <= 56, 0 <= s <= 55	string
numeric(p,s), 56 <= p <= 57, 0 <= s <= 56	string
numeric(p,s), 56 <= p <= 63, 39 <= s <= 62	string
numeric(p,s), 57 <= p <= 58, 0 <= s <= 57	string
numeric(p,s), 58 <= p <= 59, 0 <= s <= 58	string
numeric(p,s), 59 <= p <= 60, 0 <= s <= 59	string
numeric(p,s), 60 <= p <= 61, 0 <= s <= 60	string
numeric(p,s), 61 <= p <= 62, 0 <= s <= 61	string
numeric(p,s), 62 <= p <= 63, 0 <= s <= 62	string
numeric(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
real	float64

DB2 for i **ソースデータ型**	Google BigQuery **ターゲットデータ型**
rowid	bytes
smallint	int64
time	time
timestamp(*precision*), 0 <= p <= 6	datetime
timestamp(*precision*), 7 <= p <= 12	string
varbinary(*size*), 1 <= size <= 32740	bytes
varchar(*size*) for bit data, 1 <= size <= 32740	bytes
varchar(*size*), 1 <= size <= 32740	string

サポートされていないソースデータ型
一括取り込みデータベースでは、次の DB2 for i データ型はサポートされません。

- Blob
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

DB2 for i ソースと Microsoft Azure Synapse Analytics ターゲット

次の表は、DB2 for ソースと Microsoft Azure Synapse Analytics ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for i **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
bigint	bigint
binary(*size*), 1 <= size <= 8000	binary(*size*), 1 <= size <= 8000
binary(*size*), 8001 <= size <= 32766	varbinary(max)
char(*size*) for bit data, 1 <= size <= 8000	binary(*size*), 1 <= size <= 8000
char(*size*) for bit data, 8001 <= size <= 32766	varbinary(max)
char(*size*), 1 <= size <= 8000	char(*size*), 1 <= size <= 8000

DB2 for i ソースデータ型	Microsoft Azure Synapse Analytics ターゲットデータ型
char(*size*), 8001 <= size <= 32766	varchar(max)
date	date
decfloat(*precision*), 16 <= p <= 34	varchar（255）
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
decimal(p,s), 39 <= p <= 63, 0 <= s <= 62	char(*size*), 40 <= size <= 65
float	float
integer	int
longvarbinary	varbinary(max)
long varchar	varchar(max)
long varchar for bit data	varbinary(max)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 63, 0 <= s <= 62	char(*size*), 40 <= size <= 65
real	real
rowid	varbinary(40)
smallint	smallint
time	time(0)
timestamp(*precision*), 0 <= p <= 7	datetime2(*precision*), 0 <= p <= 7
timestamp(*precision*), 8 <= p <= 12	char(*size*), 28 <= size <= 32
varbinary(*size*), 1 <= size <= 32740	varbinary(*size*), 1 <= size <= max
varchar(*size*) for bit data, 1 <= size <= 32740	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 32740	varchar(*size*), 1 <= size <= max

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for i データ型はサポートされません。

- BLOB
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC

- VARGRAPHIC
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

Db2 for i ソースと Microsoft SQL Server ターゲット

次の表は、Db2 for i ソースと Microsoft SQL Server ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for i **ソースデータ型**	Microsoft SQL Server **ターゲットデータ型**
bigint	bigint
binary(*size*), 1 <= size <= 8000	varbinary(*n*), 1 <= n <= 8000
binary(*size*), 8001 <= size <= 32766	varbinary(max)
char(*size*) for bit data, 1 <= size <= 8000	varbinary(*n*), 1 <=n <= 8000
char(*size*) for bit data, 8001 <= size <= 32766	varbinary(max)
char(*size*), 1 <= size <= 8000	varchar(*n*), 1 <= n <= 8000
char(*size*), 8001 <= size <= 32766	varchar(max)
date	date
decfloat(*precision*), 16 <= p <= 34	varchar（255）
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
decimal(p,s), 39 <= p <= 63, 0 <= s <= 62	char(*n*), 40 <= n <= 65
float	float
integer	**整数型**
long varbinary	varbinary(max)
long varchar	varchar(max)
long varchar for bit data	varbinary(max)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 63, 0 <= s <= 62	char(*n*), 40 <= n <= 65
real	real
rowid	varbinary(40)
smallint	smallint
time	time(0)

DB2 for i **ソースデータ型**	Microsoft SQL Server **ターゲットデータ型**
timestamp(*precision*), 0 <= p <= 7	datetime2(*precision*), 0 <= p <= 7
timestamp(*precision*), 8 <= p <= 12	char(*n*), 28 <= n <= 32
varbinary(*size*), 1 <= size <= 32740	varbinary(*n*), 1 <= n <= max
varchar(*size*) for bit data, 1 <= size <= 32740	varbinary(*n*), 1 <= n <= max
varchar(*size*), 1 <= size <= 32740	varchar(*n*), 1 <= n <= max

サポートされていないソースデータ型
一括取り込みデータベースでは、次の DB2 for i データ型はサポートされません。

- Blob
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

DB2 for i ソースと Oracle ターゲット

次の表は、DB2 for i ソースと Oracle ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for i **ソースデータ型**	Oracle **ターゲットデータ型**
bigint	number(19)
binary(*size*), 1 <= s <= 2000	raw(*size*), 1 <= size <= 2000
binary(*size*), 2001 <= s <= 32766	blob
char(*size*) for bit data, 1 <= s <= 32766	blob
char(*size*), 1 <= s <= 2000	char(s byte), 1 <= s <= 2000
char(*size*), 2001 <= s <= 4000	varchar2(s byte), 2001 <= s <= 4000
char(*size*), 4001 <= s <= 32766	clob
date	date
decfloat(*precision*), 16 <= p <= 34	char(255 char)

DB2 for i ソースデータ型	Oracle ターゲットデータ型
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	number(p,s), 1 <= p <= 38, 0 <= s <= 38
decimal(p,s), 39 <= p <= 63, 0 <= s <= 62	char(s char), 40 <= s <= 65
float	binary_double
integer	number(10)
long varbinary	blob
long varchar	clob
long varchar for bit data	blob
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	number(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 63, 0 <= s <= 62	char(s char), 40 <= s <= 65
real	binary_double
rowid	blob
smallint	number(5)
time	char(8 char)
timestamp(0)	date
timestamp(*precision*), 1 <= p <= 9	timestamp(*precision*), 1 <= p <= 9
timestamp(*precision*), 10 <= p <= 12	char(s char), 30 <= s <= 32
varbinary(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
varbinary(*size*), 2001 <= size <= 32740	blob
varchar(*size*) for bit data, 1 <= size <= 32740	blob
varchar(*size*), 1 <= size <= 4000	varchar2(s byte), 1 <= s <= 4000
varchar(*size*), 4001 <= size <= 32740	clob

サポートされていないソースデータ型
一括取り込みデータベースでは、次の DB2 for i データ型はサポートされません。

- BLOB
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC

- VARGRAPHIC
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

DB2 for i ソースと PostgreSQL ターゲット

次の表は、DB2 for i ソースと PostgreSQL ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for i **ソースデータ型**	PostgreSQL **ターゲットデータ型**
bigint	bigint
binary(*size*), 1 <= size <= 32766	bytea
char(*size*) for bit data, 1 <= size <= 32766	bytea
char(*size*), 1 <= size <= 32766	character varying(*size*), 1 <= size <= 32766
date	date
decfloat(*precision*), 16 <= p <= 34	character varying(255)
decimal(p,s), 1 <= p <= 63, 0 <= s <= 62	numeric(p,s), 0 <= p <= 63, 1 <= s <= 62
float	doubleprecision
integer	integer
long varbinary	bytea
long varchar	text
long varchar for bit data	bytea
numeric(p,s), 0 <= p <= 63, 1 <= s <= 62	numeric(p,s), 0 <= p <= 63, 1 <= s <= 62
real	real
rowid	bytea
smallint	smallint
time	time(0) without time zone
timestamp(*precision*), 0 <= p <= 6	timestamp(*precision*) without time zone, 0 <= p <= 6
timestamp(*precision*), 7 <= p <= 12	character varying(*size*), 27 <= size <= 32
varbinary(*size*), 1 <= size <= 32740	bytea
varchar(*size*) for bit data, 1 <= size <= 32740	bytea
varchar(*size*), 1 <= size <= 32740	character varying(*size*), 1 <= size <= 32740

サポートされていないソースデータ型
一括取り込みデータベースでは、次の DB2 for i データ型はサポートされません。

- Blob
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

DB2 for i ソースと Snowflake ターゲット

次の表は、DB2 for i ソースと Snowflake ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for i **ソースデータ型**	Snowflake **ターゲットデータ型**
bigint	integer
binary(*size*), 1 <= size <= 32766	binary(*size*), 1 <= size <= 32766
char(*size*) for bit data, 1 <= size <= 32766	binary(*size*), 1 <= size <= 32766
char(*size*), 1 <= size <= 32766	char(*size*), 4 <= size <= 131064
date	date
decfloat(*precision*), 16 <= p <= 34	char(255)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
decimal(p,s), 38 <= p <= 63, 0 <= s <= 62	char(*size*), 40 <= size <= 65
float	float
integer	integer
long varbinary	binary
long varchar	varchar
long varchar for bit data	binary
numeric(p,s), 1 <= p <= 38, 0 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
numeric(p,s), 38 <= p <= 63, 0 <= s <= 62	char(*size*), 40 <= size <= 65
real	float

DB2 for i **ソースデータ型**	Snowflake **ターゲットデータ型**
rowid	binary(40)
smallint	integer
time	time(0)
timestamp(*precision*), 0 <= p <= 9	timestamp_ntz(*precision*), 0 <= p <= 9
timestamp(*precision*), 10 <= p <= 12	char(*size*), 30 <= size <= 32
varbinary(*size*), 1 <= size <= 32740	binary(*size*), 1 <= size <= 32740
varchar(*size*) for bit data, 1 <= size <= 32740	binary(*size*), 1 <= size <= 32740
varchar(*size*), 1 <= size <= 32740	varchar(*size*), 4 <= size <= 130960

サポートされていないソースデータ型
一括取り込みデータベースでは、次の DB2 for i データ型はサポートされません。

- BLOB
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

DB2 for LUW ソースと Amazon Redshift ターゲット

次の表は、DB2 for Linux、UNIX、および Windows（LUW）ソースと Amazon Redshift ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for LUW **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
bigint	bigint
binary(*size*), 1 <= size <= 255	binary varying(*size*), 1 <= size <= 255
boolean	boolean
char for bit data	binary varying(1)
char(*size*) for bit data, 2 <= size <= 255	binary varying(*size*), 2 <= size <= 255
character(*size*), 1 <= s <= 255	character(*size*), 4 <= size <= 1020

DB2 for LUW **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
date	date
decfloat(*precision*), 16 <= p <= 34	character varying(255)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	numeric(p,s), 1 <= p <= 31, 0 <= s <= 31
double	double precision
integer	integer
long varchar	character varying(65535)
long varchar for bit data	binary varying(32700)
real	real
smallint	smallint
time	time without time zone
timestamp(*precision*), 0 <= p <= 6	timestamp without time zone
timestamp(*precision*), 7 <= p <= 12	character varying(*size*), 27 <= size <= 32
varbinary(*size*), 1 <= size <= 32672	binary varying(*size*), 1 <= size <= 32672
varchar(*size*) for bit data, 1 <= size <= 32672	binary varying(*size*), 1 <= size <= 32672
varchar(*size*), 1 <= size <= 32672	character varying(*size*), 4 <= size <= 65535

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for LUW データ型はサポートされません。

- LONG VARCHAR
- LONG VARGRAPHIC
- LONG VARCHAR FOR BIT DATA
- BLOB
- CLOB
- DBCLOB
- NCLOB
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

DB2 for LUW ソースと Databricks Delta ターゲット

次の表は、DB2 for Linux、UNIX、および Windows（LUW）ソースと Databricks Delta ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for LUW **ソースデータ型**	Databricks Delta **ターゲットデータ型**
bigint	long
binary(*size*), 1 <= size <= 255	binary
boolean	boolean
char for bit data	binary
char(*size*) for bit data, 2 <= size <= 255	binary
character(*size*), 1 <= size <= 255	string
date	string
decfloat(*precision*), 16 <= p <= 34	string
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	decimal(p,s), 1 <= p <= 31, 0 <= s <= 31
double	double
integer	integer
long varchar	string
long varchar for bit data	binary
real	float
smallint	integer
time	string
timestamp(*precision*), 0 <= p <= 6	timestamp
timestamp(*precision*), 7 <= p <= 12	string
varbinary(*size*), 1 <= size <= 32672	binary
varchar(*size*) for bit data, 1 <= size <= 32672	binary
varchar(*size*), 1 <= size <= 32672	string

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for LUW データ型はサポートされません。

- LONG VARCHAR
- LONG VARGRAPHIC
- LONG VARCHAR FOR BIT DATA

- BLOB
- CLOB
- DBCLOB
- 精度よりスケールが大きい DECIMAL
- nclob
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

DB2 for LUW ソースと Google BigQuery ターゲット

次の表は、DB2 for Linux、UNIX、および Windows（LUW）ソースと Google BigQuery ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for LUW **ソースデータ型**	Google BigQuery **ターゲットデータ型**
bigint	int64
binary(*size*), 1 <= size <= 255	bytes
boolean	bool
char for bit data	bytes
char(*size*) for bit data, 2 <= size <= 255	bytes
character(*size*), 1 <= size <= 255	string
date	date
decfloat(*precision*), 16 <= p <= 34	string
decimal(1,1)	numeric
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric

DB2 for LUW **ソースデータ型**	Google BigQuery **ターゲットデータ型**
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(2,s), 1 <= s <= 2	numeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(3,s), 1 <= s <= 3	numeric
decimal(31,s), 10 <= s <= 31	bignumeric
decimal(4,s), 1 <= s <= 4	numeric
decimal(5,s), 1 <= s <= 5	numeric
decimal(6,s), 1 <= s <= 6	numeric
decimal(7,s), 1 <= s <= 7	numeric
decimal(8,s), 1 <= s <= 8	numeric
decimal(p,0), 1 <= p <= 18	int64
decimal(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
decimal(p,s), 29 <= p <= 30, 0 <= s <= 29	bignumeric
decimal(p,s), 30 <= p <= 31, 0 <= s <= 30	bignumeric
decimal(p,s), 9 <= p <= 31, 1 <= s <= 9	numeric
double	float64
integer	int64
long varchar	string

DB2 for LUW **ソースデータ型**	Google BigQuery **ターゲットデータ型**
long varchar for bit data	bytes
real	float64
smallint	int64
time	time
timestamp(*precision*), 0 <= p <= 6	datetime
timestamp(*precision*), 7 <= p <= 12	string
varbinary(*size*), 1 <= s <= 32672	bytes
varchar(*size*) for bit data, 1 <= size <= 32672	bytes
varchar(*size*), 1 <= size <= 32672	string

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for LUW データ型はサポートされません。

- LONG VARCHAR
- LONG VARGRAPHIC
- LONG VARCHAR FOR BIT DATA
- BLOB
- CLOB
- DBCLOB
- nclob
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

DB2 for LUW ソースと Microsoft Azure Synapse Analytics ターゲット

次の表は、DB2 for Linux、UNIX、および Windows（LUW）ソースと Microsoft Azure Synapse Analytics ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for LUW **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
bigint	bigint
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255
boolean	bit

DB2 for LUW **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
char for bit data	binary
char(*size*) for bit data, 2 <= size <= 255	binary(*size*), 2 <= size <= 255
char(*size*), 1 <= size <= 255	char(*size*), 1 <= size <= 255
date	date
decfloat(*precision*), 16 <= p <= 34	varchar（255）
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	decimal(p,s), 1 <= p <= 31, 0 <= s <= 31
double	float
integer	int
longvarchar	varchar(max)
long varchar for bit data	varbinary(max)
real	real
smallint	smallint
time	time(0)
timestamp(*precision*), 0 <= p <= 7	datetime2(*precision*), 0 <= p <= 7
timestamp(*precision*), 8 <= p <= 12	char(*size*), 28 <= size <= 32
varbinary(*size*), 1 <= size <= 32672	varbinary(*size*), 1 <= size <= max
varchar(*size*) for bit data, 1 <= size <= 32672	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 32672	varchar(*size*), 1 <= size <= max

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for LUW データ型はサポートされません。

- LONG VARCHAR
- LONG VARGRAPHIC
- LONG VARCHAR FOR BIT DATA
- BLOB
- CLOB
- DBCLOB
- NCLOB
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

Db2 for LUW ソースと Microsoft SQL Server ターゲット

次の表に、DB2 for Linux、UNIX、および Windows（LUW）ソースと Microsoft SQL Server ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示します。

DB2 for LUW **ソースデータ型**	Microsoft SQL Server **ターゲットデータ型**
bigint	bigint
binary(*size*), 1 <= size <= 255	varbinary(*size*), 1 <= size <= 255
boolean	bit
char for bit data	varbinary
char(*size*) for bit data, 2 <= size <= 255	varbinary(*size*), 2 <= size <= 255
character(*size*), 1 <= size <= 255	varchar(*size*), 1 <= size <= 255
date	date
decfloat(*precision*), 16 <= p <= 34	varchar（255）
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	decimal(p,s), 1 <= p <= 31, 0 <= s <= 31
double	float
integer	float
long varchar	varchar(max)
long varchar for bit data	varbinary(max)
real	real
smallint	smallint
time	time(0)
timestamp(*precision*), 0 <= p <= 7	datetime2(*precision*), 0 <= p <= 7
timestamp(*precision*), 8 <= p <= 12	varchar(*size*), 28 <= size <= 32
varbinary(*size*), 1 <= size <= 32672	varbinary(*size*), 1 <= size <= max
varchar(*size*) for bit data, 1 <= size <= 32672	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 32672	varchar(*size*), 1 <= size <= max

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for LUW データ型はサポートされません。

- LONG VARCHAR
- LONG VARGRAPHIC
- LONG VARCHAR FOR BIT DATA
- BLOB
- CLOB
- DBCLOB
- nclob
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

DB2 for LUW ソースと Oracle ターゲット

次の表に、DB2 for Linux、UNIX、および Windows（LUW）ソースと Oracle ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示します。

DB2 for LUW **ソースデータ型**	Oracle **ターゲットデータ型**
bigint	number(19)
binary(*size*), 1 <= size <= 255	raw(*size*), 1 <= size <= 255
boolean	char(1 char)
char for bit data	raw(1)
char(*size*) for bit data, 2 <= size <= 255	raw(*size*), 2 <= size <= 255
character(*size*), 1 <= size <= 255	varchar2(s byte), 1 <= s <= 255
date	date
decfloat(*precision*), 16 <= p <= 34	varchar2(255 char)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	number(p,s), 1 <= p <= 31, 0 <= s <= 31
double	binary_double
integer	number(10)
long varchar	clob
long varchar for bit data	blob
real	binary_float
smallint	number(5)

DB2 for LUW **ソースデータ型**	Oracle **ターゲットデータ型**
time	timestamp(0)
timestamp(0)	date
timestamp(*precision*), 1 <= p <= 9	timestamp(*precision*), 1 <= p <= 9
timestamp(*precision*), 10 <= p <= 12	char(s char), 30 <= s <= 32
varbinary(*size*), 1 <= size <= 1501	raw(*size*), 1 <= size <= 1501
varbinary(*size*), 2001 <= size <= 32672	blob
varchar(*size*) for bit data, 1 <= size <= 1501	raw(*size*), 1 <= size <= 1501
varchar(*size*) for bit data, 2001 <= size <= 32672	blob
varchar(*size*), 1 <= size <= 3501	varchar2(s byte), 1 <= s <= 3501
varchar(*size*), 4001 <= size <= 32672	clob

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for LUW データ型はサポートされません。

- LONG VARCHAR
- LONG VARGRAPHIC
- LONG VARCHAR FOR BIT DATA
- BLOB
- CLOB
- DBCLOB
- nclob
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

DB2 for LUW ソースと Snowflake ターゲット

次の表は、DB2 for Linux、UNIX、および Windows（LUW）ソースと Snowflake ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for LUW **ソースデータ型**	Snowflake **ターゲットデータ型**
bigint	integer
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255

DB2 for LUW **ソースデータ型**	Snowflake **ターゲットデータ型**
boolean	boolean
char for bit data	binary(1)
char(*size*) for bit data, 2 <= size <= 255	binary(*size*), 2 <= size <= 255
character(*size*), 1 <= size <= 255	char(*size*), 4 <= size <= 1020
date	date
decfloat(*precision*), 16 <= p <= 34	char(255)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	number(p,s), 1 <= p <= 31, 0 <= s <= 31
double	float
integer	integer
long varchar	varchar(130816)
long varchar for bit data	binary(32700)
real	float
smallint	integer
time	time(0)
timestamp(*precision*), 0 <= p <= 9	timestamp_ntz(*precision*), 0 <= p <= 9
timestamp(*precision*), 10 <= p <= 12	char(*size*), 30 <= size <= 32
varbinary(*size*), 1 <= size <= 32672	binary(*size*), 1 <= size <= 32672
varchar(*size*) for bit data, 1 <= size <= 32672	binary(*size*), 1 <= size <= 32672
varchar(*size*), 1 <= size <= 32672	varchar(*size*), 4 <= size <= 130688

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for LUW データ型はサポートされません。

- LONG VARCHAR
- LONG VARGRAPHIC
- LONG VARCHAR FOR BIT DATA
- BLOB
- CLOB
- DBCLOB
- NCLOB
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

DB2 for z/OS ソースと Amazon Redshift ターゲット

次の表は、DB2 for zOS ソースと Amazon Redshift ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for z/OS **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
bigint	bigint
binary(*size*), 1 <= size <= 255	binary varying(*size*), 1 <= size <= 255
char for bit data	binary varying(1)
char(*size*) for bit data, 2 <= size <= 255	binary varying(*size*), 2 <= size <= 255
char(*size*), 1 <= size <= 255	character(*size*), 4 <= size <= 1020
date	date
decfloat(*precision*), 16 <= p <= 34	character varying(255)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	numeric(p,s), 1 <= p <= 31, 0 <= s <= 31
float	doubleprecision
integer	integer
long varchar	character varying(65535)
long varchar for bit data	binary varying(32704)
real	real
rowid	binary varying(40)
smallint	smallint
time	time without time zone
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp with time zone
timestamp(*precision*) with time zone, 7 <= p <= 12	character varying(*size*), 67 <= size <= 72
timestamp(*precision*), 0 <= p <= 6	timestamp without time zone
timestamp(*precision*), 7 <= p <= 12	character varying(*size*), 27 <= size <= 32
varbinary(*size*), 1 <= size <= 32704	binary varying(*size*), 1 <= size <= 32704

DB2 for z/OS **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
varchar(*size*) for bit data, 1 <= size <= 32704	binary varying(*size*), 1 <= size <= 32704
varchar(*size*), 1 <= size <= 32704	character varying(*size*), 4 <= size <= 65535

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for z/OS データ型はサポートされません。

- BLOB
- CLOB
- DBCLOB
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

DB2 for z/OS ソースと Databricks Delta ターゲット

次の表は、DB2 for z/OS ソースと Databricks Delta ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for z/OS **ソースデータ型**	Databricks Delta **ターゲットデータ型**
bigint	long
binary(*size*), 1 <= size <= 255	binary
char for bit data	binary
char(*size*) for bit data, 2 <= size <= 255	binary
char(*size*), 1 <= size <= 255	string
date	string
decfloat(*precision*), 16 <= p <= 34	string
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	decimal(p,s), 1 <= p <= 31, 0 <= s <= 31
float	double
integer	integer
long varchar	string
long varchar for bit data	binary
real	float
rowid	binary
smallint	integer

DB2 for z/OS **ソースデータ型**	Databricks Delta **ターゲットデータ型**
time	string
timestamp(*precision*) with time zone, 0 <= p <= 12	string
timestamp(*precision*), 0 <= p <= 6	timestamp
timestamp(*precision*), 7 <= p <= 12	string
varbinary(*size*), 1 <= size <= 32704	binary
varchar(*size*) for bit data, 1 <= size <= 32704	binary
varchar(*size*), 1 <= size <= 32704	string

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for z/OS データ型はサポートされません。

- Blob
- CLOB
- DBCLOB
- 精度よりスケールが大きい DECIMAL
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

DB2 for z/OS ソースと Google BigQuery ターゲット

次の表は、DB2 for z/OS ソースと Google BigQuery ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for z/OS **ソースデータ型**	Google BigQuery **ターゲットデータ型**
bigint	int64
binary(*size*), 1 <= size <= 255	bytes
char for bit data	bytes
char(*size*) for bit data, 2 <= size <= 255	bytes
char(*size*), 1 <= size <= 255	string
date	date
decfloat(*precision*), 16 <= p <= 34	string
decimal(1,1)	numeric
decimal(10,10)	bignumeric

DB2 for z/OS **ソースデータ型**	Google BigQuery **ターゲットデータ型**
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(2,s), 1 <= s <= 2	numeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(3,s), 1 <= s <= 3	numeric
decimal(31,s), 10 <= s <= 31	bignumeric
decimal(4,s), 1 <= s <= 4	numeric
decimal(5,s), 1 <= s <= 5	numeric
decimal(6,s), 1 <= s <= 6	numeric
decimal(7,s), 1 <= s <= 7	numeric
decimal(8,s), 1 <= s <= 8	numeric
decimal(p,0), 1 <= p <= 18	int64

DB2 for z/OS **ソースデータ型**	Google BigQuery **ターゲットデータ型**
decimal(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
decimal(p,s), 29 <= p <= 30, 0 <= s <= 29	bignumeric
decimal(p,s), 30 <= p <= 31, 0 <= s <= 30	bignumeric
decimal(p,s), 9 <= p <= 31, 1 <= s <= 9	numeric
float	float64
integer	int64
long varchar	string
long varchar for bit data	bytes
real	float64
rowid	bytes
smallint	int64
time	time
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*) with time zone, 7 <= p <= 12	string
timestamp(*precision*), 0 <= p <= 6	datetime
timestamp(*precision*), 7 <= p <= 12	string
varbinary(*size*), 1 <= size <= 32704	bytes
varchar(*size*) for bit data, 1 <= size <= 32704	bytes
varchar(*size*), 1 <= size <= 32704	string

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for z/OS データ型はサポートされません。

- Blob
- CLOB
- DBCLOB
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

DB2 for z/OS ソースと Microsoft Azure Synapse Analytics ターゲット

次の表は、DB2 for z/OS ソースと Microsoft Azure Synapse Analytics ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for z/OS **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
bigint	bigint
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255
char for bit data	binary
char(*size*) for bit data, 2 <= size <= 255	binary(*size*), 2 <= size <= 255
char(*size*), 1 <= size <= 255	char(*size*), 1 <= size <= 255
date	date
decfloat(*precision*), 16 <= p <= 34	varchar（255）
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	decimal(p,s), 1 <= p <= 31, 0 <= s <= 31
float	float
integer	int
longvarchar	varchar(max)
long varchar for bit data	varbinary(max)
real	real
rowid	varbinary(40)
smallint	smallint
time	time(0)
timestamp(*precision*) with time zone, 0 <= p <= 7	datetimeoffset(*precision*), 0 <= p <= 7
timestamp(*precision*) with time zone, 8 <= p <= 12	char(*size*), 68 <= size <= 72
timestamp(*precision*), 0 <= p <= 7	datetime2(*precision*), 0 <= p <= 7
timestamp(*precision*), 8 <= p <= 12	char(*size*), 28 <= size <= 32
varbinary(*size*), 1 <= size <= 32704	varbinary(*size*), 1 <= size <= max
varchar(*size*) for bit data, 1 <= size <= 32704	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 32704	varchar(*size*), 1 <= size <= max

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for z/OS データ型はサポートされません。

- BLOB
- CLOB
- DBCLOB
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

DB2 for z/OS ソースと Oracle ターゲット

次の表は、DB2 for z/OS ソースと Oracle ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for z/OS **ソースデータ型**	Oracle **ターゲットデータ型**
bigint	number(19)
binary(*size*), 1 <= size <= 255	raw(*size*), 1 <= size <= 255
char for bit data	blob
char(*size*) for bit data, 2 <= size <= 255	blob
char(*size*), 1 <= size <= 255	char(s byte), 1 <= s <= 255
date	date
decfloat(*precision*), 16 <= p <= 34	char(255 char)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	number(p,s), 1 <= p <= 31, 0 <= s <= 31
float	binary_double
integer	number(10)
long varchar	clob
long varchar for bit data	blob
real	binary_double
rowid	blob
smallint	number(5)
time	char(8 char)
timestamp(0)	date
timestamp(*precision*) with time zone, 0 <= p <= 9	timestamp(*precision*) with time zone, 0 <= p <= 9
timestamp(*precision*) with time zone, 10 <= p <= 12	char(s char), 70 <= s <= 72
timestamp(*precision*), 1 <= p <= 9	timestamp(*precision*), 1 <= p <= 9

DB2 for z/OS **ソースデータ型**	Oracle **ターゲットデータ型**
timestamp(*precision*), 10 <= p <= 12	char(s char), 30 <= s <= 32
varbinary(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
varbinary(*size*), 2001 <= size <= 32704	blob
varchar(*size*) for bit data, 1 <= size <= 32704	blob
varchar(*size*), 1 <= size <= 3501	varchar2(s byte), 1 <= s <= 3501
varchar(*size*), 4001 <= size <= 32704	clob

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for z/OS データ型はサポートされません。

- BLOB
- CLOB
- DBCLOB
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

DB2 for z/OS ソースと Snowflake ターゲット

次の表は、DB2 for z/OS ソースと Snowflake ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

DB2 for z/OS **ソースデータ型**	Snowflake **ターゲットデータ型**
bigint	integer
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255
char for bit data	binary(1)
char(*size*) for bit data, 2 <= size <= 255	binary(*size*), 2 <= size <= 255
char(*size*), 1 <= size <= 255	char(*size*), 4 <= size <= 1020
date	date
decfloat(*precision*), 16 <= p <= 34	char(255)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	number(p,s), 1 <= p <= 31, 0 <= s <= 31
float	float
integer	integer

DB2 for z/OS **ソースデータ型**	Snowflake **ターゲットデータ型**
long varchar	varchar(130816)
long varchar for bit data	binary(32704)
real	float
rowid	binary(40)
smallint	integer
time	time(0)
timestamp(*precision*) with time zone, 0 <= p <= 9	timestamp_tz(*precision*), 0 <= p <= 9
timestamp(*precision*) with time zone, 10 <= p <= 12	char(*size*), 70 <= s <= 72
timestamp(*precision*), 0 <= p <= 9	timestamp_ntz(*precision*), 0 <= p <= 9
timestamp(*precision*), 10 <= p <= 12	char(*size*), 30 <= size <= 32
varbinary(*size*), 1 <= size <= 32704	binary(*size*), 1 <= size <= 32704
varchar(*size*) for bit data, 1 <= size <= 32704	binary(*size*), 1 <= size <= 32704
varchar(*size*), 1 <= size <= 32704	varchar(*size*), 4 <= size <= 130816

サポートされていないソースデータ型

一括取り込みデータベースでは、次の DB2 for z/OS データ型はサポートされません。

- BLOB
- CLOB
- DBCLOB
- XML

データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

Microsoft SQL Server または Azure SQL Database ソースと Amazon Redshift ターゲット

次の表は、Microsoft SQL Server ソースまたは Azure SQL Database ソースと Amazon Redshift ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Microsoft SQL Server **または** Azure SQL Database **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
bigint	bigint
binary(*size*), 1 <= size <= 8000	binary varying(*size*), 1 <= size <= 8000
bit	boolean
char(*size*), 1 <= size <= 8000	character varying(*size*), 1 <= size <= 8000
date	date
datetime	timestamp without time zone
datetime2(7)	character varying(27)
datetime2(*precision*), 0 <= p <= 6	timestamp without time zone
datetimeoffset(7)	character varying(67)
datetimeoffset(*precision*), 0 <= p <= 6	timestamp with time zone
decimal(38,38)	character varying(41)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
float	double precision
geography	binary varying(1024000)
geometry	binary varying(1024000)
hierarchyid	binary varying(892)
image	binary varying(1024000)
int	integer
money	numeric(20,4)
nchar(*size*), 1 <= size <= 4000	character varying(*size*), 1 <= size <= 8000
ntext	character varying(65535)
numeric(38,38)	character varying(41)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
nvarchar(*size*), 1 <= size <= 4000	character varying(*size*), 1 <= size <= 8000

Microsoft SQL Server **または** Azure SQL Database **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
real	real
smalldatetime	timestamp without time zone
smallint	smallint
smallmoney	numeric(10,4)
sql_variant	binary varying(8016)
text	character varying(65535)
time(*precision*), 0 <= p <= 7	varchar(16)
timestamp(8)	binary varying(8)
tinyint	smallint
uniqueidentifier	character(36)
varbinary(*size*), 1 <= size <= 8000	binary varying(*size*), 1 <= size <= 8000
varchar(*size*), 1 <= size <= 8000	character varying(*size*), 1 <= size <= 8000
xml	character varying(65535)

LOB の制限事項

データベース取り込み初期ロードジョブで、SQL Server の GEOGRAPHY、GEOMETRY、IMAGE、NTEXT、NVARCHAR(MAX)、TEXT、VARBINARY(MAX)、VARCHAR(MAX)、および XML カラムから Amazon Redshift ターゲットにデータをレプリケートできます。LOB データは、ターゲットに書き込まれる前に切り詰められる場合があります。切り詰めポイントは、データ型とターゲットタイプによって異なります。Amazon Redshift ターゲットを使用した初期ロードジョブの場合、GEOGRAPHY、GEOMETRY、IMAGE、および VARBINARY(MAX) データは 1024000 バイトに切り詰められ、NTEXT、NVARCHAR(MAX)、TEXT、VARCHAR(MAX)、および XML データは 65535 バイトに切り詰められます。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

なし。

Microsoft SQL Server ソースと Databricks Delta ターゲット

次の表は、Microsoft SQL Server ソースと Databricks Delta ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Microsoft SQL Server **ソースデータ型**	Databricks Delta **ターゲットデータ型**
bigint	bigint
binary(*size*), 1 <= size <= 8000	binary

Microsoft SQL Server **ソースデータ型**	Databricks Delta **ターゲットデータ型**
bit	boolean
char(*size*), 1 <= size <= 8000	string
date	string
datetime	timestamp
datetime2(7)	string
datetime2(*precision*), 0 <= p <= 6	timestamp
datetimeoffset(7)	string
datetimeoffset(*precision*), 0 <= p <= 6	timestamp
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
float	double
geography	binary
geometry	binary
hierarchyid	binary
image	binary
int	integer
money	decimal(19,4)
nchar(*size*), 1 <= size <= 4000	string
ntext	string
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
nvarchar(*size*), 1 <= size <= 4000	string
real	float
smalldatetime	timestamp
smallint	integer
smallmoney	decimal(10,4)
sql_variant	binary
text	string
time(*precision*), 0 <= p <= 7	string

Microsoft SQL Server **ソースデータ型**	Databricks Delta **ターゲットデータ型**
timestamp(8)	binary
tinyint	integer
uniqueidentifier	string
varbinary(*size*), 1 <= size <= 8000	binary
varchar(*size*), 1 <= size <= 8000	string
xml	string

LOB の制限事項

いずれかのタイプのロードタイプを使用するデータベース取り込みジョブで、SQL Server の GEOGRAPHY、GEOMETRY、IMAGE、NTEXT、NVARCHAR（MAX）、TEXT、VARBINARY（MAX）、VARCHAR（MAX）、および XML カラムから Databricks Delta ターゲットにデータをレプリケートできます。LOB データは、ターゲットに書き込まれる前に切り詰められる場合があります。切り詰めポイントは、データ型とロードタイプによって異なります。Databricks Delta ターゲットを使用した初期ロードジョブの場合、すべての SQL Server LOB データ型の切り詰めポイントは 16777216 バイトです。増分ロードおよび複合ロードの場合、LOB カラムに 8 KB を超えるデータが含まれていると、データは、インラインで格納されている場合は 4000 バイト、オフラインで格納されている場合は約 8000 バイトに切り詰められます。詳細については、「ソースの設定」 (ページ 102) の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

精度よりスケールが大きい DECIMAL または NUMERIC 型はサポートされていません。

Microsoft SQL Server ソースと Google BigQuery ターゲット

次の表は、Microsoft SQL Server ソースと Google BigQuery ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Microsoft SQL Server **ソースデータ型**	Google BigQuery **ターゲットデータ型**
bigint	int64
binary(*size*), 1 <= size <= 8000	bytes
bit	bool
char(*size*), 1 <= size <= 8000	string
date	date
datetime	datetime
datetime2(7)	string
datetime2(*precision*), 0 <= p <= 6	datetime
datetimeoffset(7)	string

Microsoft SQL Server **ソースデータ型**	Google BigQuery **ターゲットデータ型**
datetimeoffset(*precision*), 0 <= p <= 6	timestamp
decimal(1,1)	numeric
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(2,s), 1 <= s <= 2	numeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(3,s), 1 <= s <= 3	numeric
decimal(38,s), 10 <= s <= 38	bignumeric
decimal(4,s), 1 <= s <= 4	numeric
decimal(5,s), 1 <= s <= 5	numeric
decimal(6,s), 1 <= s <= 6	numeric

Microsoft SQL Server **ソースデータ型**	Google BigQuery **ターゲットデータ型**
decimal(7,s), 1 <= s <= 7	numeric
decimal(8,s), 1 <= s <= 8	numeric
decimal(p,0), 1 <= p <= 18	int64
decimal(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
decimal(p,s), 29 <= p <= 30, 0 <= s <= 29	bignumeric
decimal(p,s), 30 <= p <= 31, 0 <= s <= 30	bignumeric
decimal(p,s), 31 <= p <= 32, 0 <= s <= 31	bignumeric
decimal(p,s), 32 <= p <= 33, 0 <= s <= 32	bignumeric
decimal(p,s), 33 <= p <= 34, 0 <= s <= 33	bignumeric
decimal(p,s), 34 <= p <= 35, 0 <= s <= 34	bignumeric
decimal(p,s), 35 <= p <= 36, 0 <= s <= 35	bignumeric
decimal(p,s), 36 <= p <= 37, 0 <= s <= 36	bignumeric
decimal(p,s), 37 <= p <= 38, 0 <= s <= 37	bignumeric
decimal(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
float	float64
geography	bytes
geometry	bytes
hierarchyid	bytes
image	bytes
int	int64
money	numeric
nchar(*size*), 1 <= size <= 4000	string
ntext	string
numeric(1,1)	numeric
numeric(10,10)	bignumeric
numeric(11,s), 10 <= s <= 11	bignumeric
numeric(12,s), 10 <= s <= 12	bignumeric

Microsoft SQL Server **ソースデータ型**	Google BigQuery **ターゲットデータ型**
numeric(13,s), 10 <= s <= 13	bignumeric
numeric(14,s), 10 <= s <= 14	bignumeric
numeric(15,s), 10 <= s <= 15	bignumeric
numeric(16,s), 10 <= s <= 16	bignumeric
numeric(17,s), 10 <= s <= 17	bignumeric
numeric(18,s), 10 <= s <= 18	bignumeric
numeric(19,s), 10 <= s <= 19	bignumeric
numeric(2,s), 1 <= s <= 2	numeric
numeric(20,s), 10 <= s <= 20	bignumeric
numeric(21,s), 10 <= s <= 21	bignumeric
numeric(22,s), 10 <= s <= 22	bignumeric
numeric(23,s), 10 <= s <= 23	bignumeric
numeric(24,s), 10 <= s <= 24	bignumeric
numeric(25,s), 10 <= s <= 25	bignumeric
numeric(26,s), 10 <= s <= 26	bignumeric
numeric(27,s), 10 <= s <= 27	bignumeric
numeric(28,s), 10 <= s <= 28	bignumeric
numeric(3,s), 1 <= s <= 3	numeric
numeric(38,s), 10 <= s <= 38	bignumeric
numeric(4,s), 1 <= s <= 4	numeric
numeric(5,s), 1 <= s <= 5	numeric
numeric(6,s), 1 <= s <= 6	numeric
numeric(7,s), 1 <= s <= 7	numeric
numeric(8,s), 1 <= s <= 8	numeric
numeric(p,0), 1 <= p <= 18	int64
numeric(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
numeric(p,s), 29 <= p <= 30, 0 <= s <= 29	bignumeric

Microsoft SQL Server **ソースデータ型**	Google BigQuery **ターゲットデータ型**
numeric(p,s), 30 <= p <= 31, 0 <= s <= 30	bignumeric
numeric(p,s), 31 <= p <= 32, 0 <= s <= 31	bignumeric
numeric(p,s), 32 <= p <= 33, 0 <= s <= 32	bignumeric
numeric(p,s), 33 <= p <= 34, 0 <= s <= 33	bignumeric
numeric(p,s), 34 <= p <= 35, 0 <= s <= 34	bignumeric
numeric(p,s), 35 <= p <= 36, 0 <= s <= 35	bignumeric
numeric(p,s), 36 <= p <= 37, 0 <= s <= 36	bignumeric
numeric(p,s), 37 <= p <= 38, 0 <= s <= 37	bignumeric
numeric(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
nvarchar(*size*), 1 <= size <= 4000	string
real	float64
smalldatetime	datetime
smallint	int64
smallmoney	numeric
sql_variant	bytes
text	string
time(7)	string
time(*precision*), 0 <= p <= 6	time
timestamp(8)	bytes
tinyint	int64
uniqueidentifier	string
varbinary(*size*), 1 <= size <= 8000	bytes
varchar(*size*), 1 <= size <= 8000	string
xml	string

LOB の制限事項

データベース取り込み初期ロードジョブで、SQL Server の GEOGRAPHY、GEOMETRY、IMAGE、NTEXT、NVARCHAR(MAX)、TEXT、VARBINARY(MAX)、VARCHAR(MAX)、および XML カラムから Google BigQuery ターゲットにデータをレプリケートできます。LOB データは、ターゲットに書き込まれる前に切り詰められる場

合があります。すべての LOB データ型で、切り詰めポイントは 8388608 バイトです。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

なし。

Microsoft SQL Server ソースと Parquet 出力形式を使用するターゲット

次の表は、Microsoft SQL Server ソースと、Parquet 出力形式を使用する Amazon S3、Google Cloud Storage、Microsoft Azure Data Lake Storage Gen2、または Microsoft Fabric OneLake ターゲットを使用した一括取り込みデータベース構成の推奨データ型マッピングを示しています。

Microsoft SQL Server **ソースデータ型**	Parquet **データ型**
bigint	init64
binary	byte array
bit	int32
char	string
date	string
datetime	string
datetime2	string
datetimeoffset	string
decimal	string
float	string
geography	byte array
geometry	byte array
hierarchyid	byte array
image	byte array
int	int32
money	string
nchar	string
ntext	string
numeric	string
nvarchar	string

Microsoft SQL Server **ソースデータ型**	Parquet **データ型**
real	string
smalldatetime	string
smallint	int32
smallmoney	string
sql_variant	byte array（初期ロード） string（増分ロード）
text	string
time	string
timestamp	binary
tinyint	int32
uniqueidentifier	string
varbinary	byte array
varchar	string
xml	string

Microsoft SQL Server または Azure SQL Database ソースと Microsoft Azure Synapse Analytics ターゲット

次の表は、Microsoft SQL Server または Azure SQL Database ソースと Microsoft Azure Synapse Analytics ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Microsoft SQL Server **または** Azure SQL Database **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
bigint	bigint
binary(*size*), 1 <= size <= 8000	binary(*size*), 1 <= size <= 8000
bit	bit
char(*size*), 1 <= size <= 8000	char(*size*), 1 <= size <= 8000
date	date
datetime	datetime2(3)
datetime2(*precision*), 0 <= p <= 7	datetime2(*precision*), 0 <= p <= 7
datetimeoffset(*precision*), 0 <= p <= 7	datetimeoffset(*precision*), 0 <= p <= 7

Microsoft SQL Server または Azure SQL Database ソースデータ型	Microsoft Azure Synapse Analytics ターゲットデータ型
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
float	float
geography	varbinary(max)
geometry	varbinary(max)
hierarchyid	varbinary(892)
image	varbinary(max)
int	int
money	money
nchar(*size*), 1 <= size <= 4000	nchar(*size*), 1 <= size <= 4000
ntext	nvarchar(max)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	numeric(p,s), 1 <= p <= 38, 0 <= s <= 38
nvarchar(*size*), 1 <= size <= 4000	nvarchar(*size*), 1 <= size <= 4000
real	real
smalldatetime	datetime2(0)
smallint	smallint
smallmoney	smallmoney
sql_variant	varbinary(max)
text	varchar(max)
time(*precision*), 0 <= p <= 7	time(*precision*), 0 <= p <= 7
timestamp(8)	varbinary(8)
tinyint	tinyint
uniqueidentifier	uniqueidentifier
varbinary(*size*), 1 <= size <= 8000	varbinary(*size*), 1 <= size <= 8000
varchar(*size*), 1 <= size <= 8000	varchar(*size*), 1 <= size <= 8000
xml	varchar(max)

LOB の制限事項

データベース取り込み初期ロードジョブで、SQL Server の GEOGRAPHY、GEOMETRY、IMAGE、NTEXT、NVARCHAR(MAX)、TEXT、VARBINARY(MAX)、VARCHAR(MAX)、および XML カラムから Azure Synapse Analytics ターゲットにデータをレプリケートできます。LOB データは、ターゲットに書き込まれる前に切り詰められる場合があります。切り詰めポイントは、データ型とターゲットタイプによって異なります。Azure Synapse Analytics ターゲットを使用した初期ロードジョブの場合、GEOGRAPHY、GEOMETRY、IMAGE、および VARBINARY(MAX)データは 1000000 バイトに切り詰められ、NTEXT、NVARCHAR(MAX)、TEXT、VARCHAR(MAX)、および XML データは 500000 バイトに切り詰められます。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

なし。

Microsoft SQL Server ソースと Microsoft SQL Server ターゲット

次の表は、Microsoft SQL Server ソースと Microsoft SQL Server ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

SQL Server **ソースデータ型**	SQL Server **ターゲットデータ型**
bigint	bigint
binary	varbinary
bit	bit
char	varchar
date	date
datetime	datetime
datetime2	datetime2
datetimeoffset	datetimeoffset
decimal	decimal
float	float
geography	geography
geometry	geometry
hierarchyid	hierarchyid
image	image
int	int
money	money
nchar	nvarchar
ntext	ntext

SQL Server **ソースデータ型**	SQL Server **ターゲットデータ型**
numeric	numeric
nvarchar	nvarchar
real	real
smalldatetime	smalldatetime
smallint	smallint
smallmoney	smallmoney
sql_variant	sql_variant
text	text
time	time
timestamp	varbinary
tinyint	tinyint
uniqueidentifier	uniqueidentifier
varbinary	varbinary
varchar	varchar
xml	xml

Sql_variant ターゲットデータ型

SQL Server ソースと SQL Server ターゲットを持ち、sql_variant ソースカラムを含むデータベース取り込みの初期ロードジョブは、ターゲット上で sql_variant データを 16 進形式に変換します。データを 16 進形式から varbinary 形式に変換するには、次のクエリを実行します。

```
SELECT <column_name>, CONVERT(varbinary,<column_name>) from <table_name>;
```

<*column_name*>と<*table_name*>を実際のターゲットカラムとテーブル名に置き換えます。

LOB の制限事項

データベース取り込み初期ロードジョブと増分ロードジョブで、SQL Server の GEOGRAPHY、GEOMETRY、IMAGE、NTEXT、NVARCHAR（MAX）、TEXT、VARBINARY（MAX）、VARCHAR（MAX）、および XML カラムから SQL Server ターゲットにデータをレプリケートできます。LOB データは、ターゲットに書き込まれる前に切り詰められる場合があります。切り詰めポイントは、データ型とターゲットタイプによって異なります。SQL Server ターゲットを使用した初期ロードジョブの場合、GEOGRAPHY、GEOMETRY、IMAGE、TEXT、VARBINARY（MAX）、および VARCHAR（MAX）データは 16777216 バイトに、NTEXT と NVARCHAR（MAX）データは 33554432 バイトに、XML データは 33554442 バイトに切り詰められます。

サポートされていないソースデータ型

なし。

Microsoft SQL Server ソースと Oracle ターゲット

次の表は、Microsoft SQL Server ソースと Oracle ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Microsoft SQL Server **ソースデータ型**	Oracle **ターゲットデータ型**
bigint	number(19)
binary(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
binary(*size*), 2001 <= size <= 8000	blob
bit	char(1 char)
char(*size*), 1 <= size <= 2000	char(s char), 1 <= size <= 2000
char(*size*), 2001 <= size <= 4000	varchar2(s char), 2001 <= size <= 4000
char(*size*), 4001 <= size <= 8000	clob
date	date
datetime	timestamp(3)
datetime2(0)	date
datetime2(*precision*), 1 <= p <= 7	timestamp(*precision*), 1 <= p <= 7
datetimeoffset(*precision*), 0 <= p <= 7	timestamp(*precision*) with time zone, 0 <= p <= 7
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	number(p,s), 1 <= p <= 38, 0 <= s <= 38
float	binary_double
geography	blob
geometry	blob
hierarchyid	blob
image	blob
int	number(10)
money	number(19,4)
nchar(*size*), 1 <= size <= 1000	nchar(s char), 1 <= size <= 1000
nchar(*size*), 1001 <= size <= 2000	nvarchar2(s char), 1001 <=size <= 2000
nchar(*size*), 2001 <= size <= 4000	nclob
ntext	nclob
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	number(p,s), 1 <= p <= 38, 0 <= s <= 38

Microsoft SQL Server **ソースデータ型**	Oracle **ターゲットデータ型**
nvarchar(*size*), 1 <= size <= 2000	nvarchar2(s char), 1 <= size <= 2000
nvarchar(*size*), 2001 <= size <= 4000	nclob
real	binary_float
smalldatetime	date
smallint	number(5)
smallmoney	number(10,4)
sql_variant	blob
text	clob
time(*precision*), 0 <= p <= 7	timestamp(*precision*), 0 <= p <= 7
timestamp(8)	raw(8)
tinyint	number(3)
uniqueidentifier	char(36 char)
varbinary(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
varbinary(*size*), 2001 <= size <= 8000	blob
varchar(*size*), 1 <= size <= 4000	varchar2(s char), 1 <= size <= 4000
varchar(*size*), 4001 <= size <= 8000	clob
xml	clob

LOB の制限事項

データベース取り込み初期ロードジョブで、SQL Server の GEOGRAPHY、GEOMETRY、IMAGE、NTEXT、NVARCHAR(MAX)、TEXT、VARBINARY(MAX)、VARCHAR(MAX)、および XML カラムから Oracle ターゲットにデータをレプリケートできます。LOB データは、ターゲットに書き込まれる前に切り詰められる場合があります。すべての LOB データ型で、切り詰めポイントは 16777216 バイトです。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

なし。

Microsoft SQL Server または Azure SQL Database ソースと Snowflake ターゲット

次の表は、Microsoft SQL Server または Azure SQL Database ソースと Snowflake ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Microsoft SQL Server **または** Azure SQL Database **ソースデータ型**	Snowflake **ターゲットデータ型**
bigint	number(38,0)
binary(*size*), 1 <= size <= 8000	binary(*size*), 1 <= size <= 8000
bit	boolean
char(*size*), 1 <= size <= 8000	varchar(*size*), 1 <= size <= 8000
date	date
datetime	timestamp_ntz(3)
datetime2(*precision*), 0 <= p <= 7	timestamp_ntz(7)
datetimeoffset(*precision*), 0 <= p <= 7	timestamp_tz(*precision*), 0 <= p <= 7
decimal(38,38)	char(41)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
float	float
geography	binary
geometry	binary
hierarchyid	binary(892)
image	binary
int	number(38,0)
money	number(19,4)
nchar(*size*), 1 <= size <= 4000	varchar(*size*), 4 <= size <= 16000
ntext	varchar
numeric(38,38)	char(41)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
nvarchar(*size*), 1 <= size <= 4000	varchar(*size*), 4 <= size <= 16000
real	float
smalldatetime	timestamp_ntz(0)

Microsoft SQL Server **または** Azure SQL Database **ソースデータ型**	Snowflake **ターゲットデータ型**
smallint	number(38,0)
smallmoney	number(10,4)
sql_variant	binary(8016)
text	varchar
time(*precision*), 0 <= p <= 7	time(*precision*), 0 <= p <= 7
timestamp(8)	binary(8)
tinyint	number(38,0)
uniqueidentifier	char(36)
varbinary(*size*), 1 <= size <= 8000	binary(*size*), 1 <= size <= 8000
varchar(*size*), 1 <= size <= 8000	varchar(*size*), 1 <= size <= 8000
xml	varchar

LOB の制限事項

いずれかのタイプのロードタイプを使用するデータベース取り込みジョブで、SQL Server の GEOGRAPHY、GEOMETRY、IMAGE、NTEXT、NVARCHAR（MAX）、TEXT、VARBINARY（MAX）、VARCHAR（MAX）、および XML カラムから Snowflake ターゲットにデータをレプリケートできます。LOB データは、ターゲットに書き込まれる前に切り詰められる場合があります。切り詰めポイントは、データ型とロードタイプによって異なります。Snowflake ターゲットを使用した初期ロードジョブの場合、GEOGRAPHY、GEOMETRY、IMAGE、および VARBINARY(MAX)データは 8388608 バイトに切り詰められ、NTEXT、NVARCHAR(MAX)、TEXT、VARCHAR(MAX)、および XML データは 16777216 バイトに切り詰められます。増分ロードおよび複合ロードの場合、LOB カラムに 8 KB を超えるデータが含まれていると、データは、インラインで格納されている場合は 4000 バイト、オフラインで格納されている場合は約 8000 バイトに切り詰められます。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

Snowflake ターゲットを持つ増分ロードジョブ、および初期ロードジョブと増分ロードジョブの組み合わせの場合、一括取り込みデータベースは、SQL Server Hierarchyid データ型をサポートしていません。データベース統合ジョブは、これらのデータ型を持つカラムには null をプロパゲートします。

MongoDB ソースと Amazon Redshift ターゲット

次の表は、MongoDB ソースと Amazon Redshift ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

MongoDB **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
document(2147483647)	super
id（32）	character varying(128)

注: MongoDB ソースの場合、一括取り込みデータベースは、ターゲットに 2 つのカラムを作成します。つまり、プライマリキーと同様の識別子を提供する ID カラムと、MongoDB レコード全体を含んだ JSON オブジェクトが格納された Document カラムです。したがって、マッピングは、これらの各カラムに関連付けられたターゲットデータ型を示します。

MongoDB ソースと Databricks Delta ターゲット

次の表は、MongoDB ソースと Databricks Delta ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

MongoDB **ソースデータ型**	Databricks Delta **ターゲットデータ型**
document(2147483647)	string
id（32）	string

注: MongoDB ソースの場合、一括取り込みデータベースは、ターゲットに 2 つのカラムを作成します。つまり、プライマリキーと同様の識別子を提供する ID カラムと、MongoDB レコード全体を含んだ JSON オブジェクトが格納された Document カラムです。したがって、マッピングは、これらの各カラムに関連付けられたターゲットデータ型を示します。

MongoDB ソースと Google BigQuery ターゲット

次の表は、MongoDB ソースと Google BigQuery ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

MongoDB **ソースデータ型**	Google BigQuery **ターゲットデータ型**
document(2147483647)	string
id（32）	string

注: MongoDB ソースの場合、一括取り込みデータベースは、ターゲットに 2 つのカラムを作成します。つまり、プライマリキーと同様の識別子を提供する ID カラムと、MongoDB レコード全体を含んだ JSON オブジェクトが格納された Document カラムです。したがって、マッピングは、これらの各カラムに関連付けられたターゲットデータ型を示します。

MongoDB ソースと Microsoft Azure Synapse Analytics ターゲット

次の表は、MongoDB ソースと Microsoft Azure Synapse Analytics ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

MongoDB **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
document(2147483647)	nvarchar(max)
id（32）	nchar（32）

注: MongoDB ソースの場合、一括取り込みデータベースは、ターゲットに 2 つのカラムを作成します。つまり、プライマリキーと同様の識別子を提供する ID カラムと、MongoDB レコード全体を含んだ JSON オブジェ

クトが格納された Document カラムです。したがって、マッピングは、これらの各カラムに関連付けられたターゲットデータ型を示します。

MongoDB ソースと Microsoft SQL Server ターゲット

次の表に、MongoDB ソースと Microsoft SQL Server ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示します。

MongoDB **ソースデータ型**	Microsoft SQL Server **ターゲットデータ型**
document(2147483647)	varchar(max)
id（32）	char(128)

注: MongoDB ソースの場合、一括取り込みデータベースは、ターゲットに 2 つのカラムを作成します。つまり、プライマリキーと同様の識別子を提供する ID カラムと、MongoDB レコード全体を含んだ JSON オブジェクトが格納された Document カラムです。したがって、マッピングは、これらの各カラムに関連付けられたターゲットデータ型を示します。

MongoDB ソースと Oracle ターゲット

次の表は、MongoDB ソースと Oracle ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

MongoDB **ソースカラムタイプ**	Oracle **ターゲットデータ型**
document(2147483647)	clob
id（32）	char(128)

注: MongoDB ソースの場合、一括取り込みデータベースは、ターゲットに 2 つのカラムを作成します。つまり、プライマリキーと同様の識別子を提供する ID カラムと、MongoDB レコード全体を含んだ JSON オブジェクトが格納された Document カラムです。したがって、マッピングは、これらの各カラムに関連付けられたターゲットデータ型を示します。

MongoDB ソースと Snowflake ターゲット

次の表は、MongoDB ソースと Snowflake ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

MongoDB **ソースデータ型**	Snowflake **ターゲットデータ型**
document(2147483647)	variant
id（32）	varchar(128)

注: MongoDB ソースの場合、一括取り込みデータベースは、ターゲットに 2 つのカラムを作成します。つまり、プライマリキーと同様の識別子を提供する ID カラムと、MongoDB レコード全体を含んだ JSON オブジェクトが格納された Document カラムです。したがって、マッピングは、これらの各カラムに関連付けられたターゲットデータ型を示します。

MySQL ソースと Amazon Redshift ターゲット

次の表は、MySQL ソースと Amazon Redshift ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

MySQL **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
bigint	bigint
bigint unsigned	numeric(20,0)
binary(*size*), 1 <= size <= 255	binary varying(*size*), 1 <= size <= 255
bit	binary varying(2)
bit(*precision*), 1 <= p <= 64	binary varying(*size*), 1 <= size <= 8
blob	binary varying(65535)
char(*size*), 1 <= size <= 255	character varying(*size*), 3 <= size <= 765
date	date
datetime	timestamp without time zone
decimal(p,s), 1 <= p <= 38, 0 <= s <= 29	numeric(p,s), 1 <= p <= 38, 0 <= s <= 29
decimal(p,s), 39 <= p <= 65, 0 <= s <= 29	character varying(*size*), 40 <= size <= 67
double	double precision
float	real
geomcollection	binary varying(1024000)
geometry	binary varying(1024000)
geometrycollection	binary varying(1024000)
int	integer
int unsigned	bigint
json	character varying(65535)
linestring	binary varying(1024000)
longblob	binary varying(1024000)
longtext	character varying(65535)
mediumblob	binary varying(1024000)
mediumint	integer
mediumint unsigned	integer

MySQL ソースデータ型	Amazon Redshift ターゲットデータ型
mediumtext	character varying(65535)
multilinestring	binary varying(1024000)
multipoint	binary varying(1024000)
multipolygon	binary varying(1024000)
numeric	numeric(10,0)
point	binary varying(1024000)
polygon	binary varying(1024000)
smallint	smallint
smallint unsigned	integer
text	character varying(65535)
time(*precision*), 0 <= p <= 6	character varying(*size*), 10 <= size <= 17
timestamp(*precision*), 0 <= p <= 6	timestamp without time zone
tinyblob	binary varying(255)
tinyint	smallint
tinyint unsigned	smallint
tinytext	character varying(255)
varbinary(*size*), 1 <= size <= 65535	binary varying(*size*), 1 <= size <= 65535
varchar(*size*), 1 <= size <= 21844	character varying(*size*), 3 <= size <= 65532
year	smallint

サポートされていないソースデータ型

一括取り込みデータベースでは、次の MySQL データ型はサポートされません。

- BLOB
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TEXT
- TINYBLOB

- TINYTEXT

データベース取り込みタスクで、JSON データ型のカラムを含むソーススキーマを指定する場合、タスクをデプロイすると、JSON カラムが無視され、ターゲットに対応するカラムは作成されません。サポートされていない他のデータ型の場合、データ型がデフォルトのマッピングに表示されていても、データベース取り込みジョブは null をプロパゲートします。

MySQL ソースと Databricks Delta ターゲット

次の表は、MySQL ソースと Databricks Delta ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

MySQL **ソースデータ型**	Databricks Delta **ターゲットデータ型**
bigint	long
bigint unsigned	decimal(20)
binary(*size*), 1 <= size <= 255	binary
bit(*precision*), 1 <= p <= 64	binary
blob	binary
char(*size*), 1 <= size <= 255	string
date	string
datetime	timestamp
decimal(p,s), 1 <= p <= 38, 0 <= s <= 29	decimal(p,s), 1 <= p <= 38, 0 <= s <= 29
decimal(p,s), 39 <= p <= 65, 0 <= s <= 29	string
double	double
float	float
geomcollection	binary
geometry	binary
geometrycollection	binary
int	integer
int unsigned	long
json	string
linestring	binary
longblob	binary
longtext	string

MySQL **ソースデータ型**	Databricks Delta **ターゲットデータ型**
mediumblob	binary
mediumint	integer
mediumint unsigned	integer
mediumtext	string
multilinestring	binary
multipoint	binary
multipolygon	binary
numeric	decimal
point	binary
polygon	binary
smallint	integer
smallint unsigned	integer
text	string
time(*precision*), 0 <= p <= 6	string
timestamp(*precision*), 0 <= p <= 6	TIMESTAMP
tinyblob	binary
tinyint	integer
tinyint unsigned	integer
tinytext	string
varbinary(*size*), 1 <= size <= 65535	binary
varchar(*size*), 1 <= size <= 21844	string
year	integer

サポートされていないソースデータ型

一括取り込みデータベースでは、次の MySQL データ型はサポートされません。

- BLOB
- 精度よりスケールが大きい DECIMAL
- JSON
- LONGBLOB

- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TEXT
- TINYBLOB
- TINYTEXT

データベース取り込みタスクで、JSON データ型のカラムを含むソーススキーマを指定する場合、タスクをデプロイすると、JSON カラムが無視され、ターゲットに対応するカラムは作成されません。サポートされていない他のデータ型の場合、データ型がデフォルトのマッピングに表示されていても、データベース取り込みジョブは null をプロパゲートします。

MySQL ソースと Google BigQuery ターゲット

次の表は、MySQL ソースと Google BigQuery ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

MySQL **ソースデータ型**	Google BigQuery **ターゲットデータ型**
bigint	int64
bigint unsigned	bignumeric
binary(*size*), 1 <= size <= 255	bytes
bit(*precision*), 1 <= p <= 64	bytes
blob	bytes
char(*size*), 1 <= size <= 255	string
date	date
datetime	datetime
decimal(1,1)	numeric
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric

MySQL ソースデータ型	Google BigQuery ターゲットデータ型
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(2,s), 1 <= s <= 2	numeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(3,s), 1 <= s <= 3	numeric
decimal(4,s), 1 <= s <= 4	numeric
decimal(40,0)	string
decimal(41,s), 0 <= s <= 1	string
decimal(42,s), 0 <= s <= 2	string
decimal(43,s), 0 <= s <= 3	string
decimal(44,s), 0 <= s <= 4	string
decimal(45,s), 0 <= s <= 5	string
decimal(46,s), 0 <= s <= 6	string
decimal(47,s), 0 <= s <= 7	string
decimal(48,s), 0 <= s <= 8	string
decimal(49,s), 0 <= s <= 9	string
decimal(5,s), 1 <= s <= 5	numeric
decimal(50,s), 0 <= s <= 10	string
decimal(51,s), 0 <= s <= 11	string

MySQL ソースデータ型	Google BigQuery ターゲットデータ型
decimal(52,s), 0 <= s <= 12	string
decimal(53,s), 0 <= s <= 13	string
decimal(54,s), 0 <= s <= 14	string
decimal(55,s), 0 <= s <= 15	string
decimal(56,s), 0 <= s <= 16	string
decimal(57,s), 0 <= s <= 17	string
decimal(58,s), 0 <= s <= 18	string
decimal(59,s), 0 <= s <= 19	string
decimal(6,s), 1 <= s <= 6	numeric
decimal(60,s), 0 <= s <= 20	string
decimal(61,s), 0 <= s <= 21	string
decimal(62,s), 0 <= s <= 22	string
decimal(63,s), 0 <= s <= 23	string
decimal(64,s), 0 <= s <= 24	string
decimal(65,s), 0 <= s <= 25	string
decimal(7,s), 1 <= s <= 7	numeric
decimal(8,s), 1 <= s <= 8	numeric
decimal(p,0), 1 <= p <= 18	int64
decimal(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
decimal(p,s), 29 <= p <= 64, 0 <= s <= 29	bignumeric
decimal(p,s), 30 <= p <= 31, 0 <= s <= 29	bignumeric
decimal(p,s), 39 <= p <= 65, 21 <= s <= 29	bignumeric
decimal(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
double	float64
float	float64
geomcollection	bytes
geometry	bytes

MySQL ソースデータ型	Google BigQuery ターゲットデータ型
geometrycollection	bytes
int	int64
int unsigned	int64
json	string
linestring	bytes
longblob	bytes
longtext	string
mediumblob	bytes
mediumint	int64
mediumint unsigned	int64
mediumtext	string
multilinestring	bytes
multipoint	bytes
multipolygon	bytes
numeric	int64
point	bytes
polygon	bytes
smallint	int64
smallint unsigned	int64
text	string
time(*precision*), 0 <= p <= 6	string
timestamp(*precision*), 0 <= p <= 6	datetime
tinyblob	bytes
tinyint	int64
tinyint unsigned	int64
tinytext	string
varbinary(*size*), 1 <= size <= 65535	bytes

MySQL **ソースデータ型**	Google BigQuery **ターゲットデータ型**
varchar(*size*), 1 <= size <= 21844	string
year	int64

サポートされていないソースデータ型

一括取り込みデータベースでは、次の MySQL データ型はサポートされません。

- BLOB
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TEXT
- TINYBLOB
- TINYTEXT

データベース取り込みタスクで、JSON データ型のカラムを含むソーススキーマを指定する場合、タスクをデプロイすると、JSON カラムが無視され、ターゲットに対応するカラムは作成されません。サポートされていない他のデータ型の場合、データ型がデフォルトのマッピングに表示されていても、データベース取り込みジョブは null をプロパゲートします。

MySQL ソースと Microsoft Azure Synapse Analytics ターゲット

次の表は、MySQL ソースと Microsoft Azure Synapse Analytics ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

MySQL **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
bigint	bigint
bigint unsigned	decimal(20)
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255
bit	bit
bit(*precision*), 1 <= p <= 64	binary(*size*), 1 <= size <= 8
blob	varbinary(max)
char(*size*), 1 <= size <= 255	varchar(*size*), 4 <= size <= 1020
date	date
datetime	datetime2(0)

MySQL ソースデータ型	Microsoft Azure Synapse Analytics ターゲットデータ型
decimal(p,s), 1 <= p <= 38, 0 <= s <= 29	decimal(p,s), 1 <= p <= 38, 0 <= s <= 29
decimal(p,s), 39 <= p <= 65, 0 <= s <= 29	char(*size*), 40 <= size <= 67
double	float
float	real
geomcollection	varbinary(max)
geometry	varbinary(max)
geometrycollection	varbinary(max)
int	int
int unsigned	bigint
json	varchar(max)
linestring	varbinary(max)
longblob	varbinary(max)
longtext	varchar(max)
mediumblob	varbinary(max)
mediumint	int
mediumint unsigned	int
mediumtext	varchar(max)
multilinestring	varbinary(max)
multipoint	varbinary(max)
multipolygon	varbinary(max)
numeric	decimal(10)
point	varbinary(max)
polygon	varbinary(max)
smallint	smallint
smallint unsigned	int
text	varchar(max)
time(*precision*), 0 <= p <= 6	varchar(*size*), 10 <= size <= 17

MySQL **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
timestamp(*precision*), 0 <= p <= 6	datetime2(*precision*), 0 <= p <= 6
tinyblob	binary(255)
tinyint	smallint
tinyint unsigned	smallint
tinytext	char(256)
varbinary(*size*), 1 <= size <= 65535	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 21844	varchar(*size*), 4 <= size <= max
year	smallint

サポートされていないソースデータ型

一括取り込みデータベースでは、次の MySQL データ型はサポートされません。

- BLOB
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TEXT
- TINYBLOB
- TINYTEXT

データベース取り込みタスクで、JSON データ型のカラムを含むソーススキーマを指定する場合、タスクをデプロイすると、JSON カラムが無視され、ターゲットに対応するカラムは作成されません。サポートされていない他のデータ型の場合、データ型がデフォルトのマッピングに表示されていても、データベース取り込みジョブは null をプロパゲートします。

MySQL ソースと Microsoft SQL Server ターゲット

次の表に、MySQL ソースと Microsoft SQL Server ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示します。

MySQL **ソースデータ型**	Microsoft SQL Server **ターゲットデータ型**
bigint	bigint
bigint unsigned	decimal(20,0)
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255

MySQL **ソースデータ型**	Microsoft SQL Server **ターゲットデータ型**
bit	bit
bit(*precision*), 1 <= p <= 64	binary(*size*), 1 <= size <= 8
blob	varbinary(max)
char(*size*), 1 <= size <= 255	varchar(*size*), 3 <= size <= 765
date	date
datetime	datetime2
decimal(p,s), 1 <= p <= 38, 0 <= s <= 29	decimal(p,s), 1 <= p <= 38, 0 <= s <= 29
decimal(p,s), 39 <= p <= 65, 0 <= s <= 29	char(*size*), 40 <= size <= 67
double	float
float	real
geomcollection	binary(8000)
geometry	varbinary(max)
geometrycollection	varbinary(max)
int	int
int unsigned	bigint
json	varchar(max)
linestring	varbinary(max)
longblob	varbinary(max)
longtext	varchar(max)
mediumblob	varbinary(max)
mediumint	int
mediumint unsigned	int
mediumtext	varchar(max)
multilinestring	varbinary(max)
multipoint	varbinary(max)
multipolygon	varbinary(max)
numeric(10,0)	decimal(10,0)

MySQL **ソースデータ型**	Microsoft SQL Server **ターゲットデータ型**
point	varbinary(max)
polygon	varbinary(max)
smallint	smallint
smallint unsigned	int
text	varchar(max)
time(*precision*), 0 <= p <= 6	varchar(*size*), 10 <= size <= 17
timestamp(*precision*), 0 <= p <= 3	datetime2
timestamp(*precision*), 4 <= p <= 6	datetime2(*precision*), 4 <= p <= 6
tinyblob	binary(255)
tinyint	smallint
tinyint unsigned	tinyint
tinytext	varchar（255）
varbinary(*size*), 1 <= size <= 65535	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 21844	varchar(*size*), 3 <= size <= max
year	smallint

サポートされていないソースデータ型

一括取り込みデータベースでは、次の MySQL データ型はサポートされません。

- BLOB
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TEXT
- TINYBLOB
- TINYTEXT

データベース取り込みタスクで、JSON データ型のカラムを含むソーススキーマを指定する場合、タスクをデプロイすると、JSON カラムが無視され、ターゲットに対応するカラムは作成されません。サポートされていない他のデータ型の場合、データ型がデフォルトのマッピングに表示されていても、データベース取り込みジョブは null をプロパゲートします。

MySQL ソースと Oracle ターゲット

次の表に、MySQL ソースと Oracle ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示します。

MySQL ソースデータ型	Oracle ターゲットデータ型
bigint	number(19)
bigint unsigned	number(20)
binary(*size*), 1 <= size <= 255	raw(*size*), 1 <= size <= 255
bit(*precision*), 1 <= p <= 64	raw(*size*), 1 <= size <= 8
blob	blob
char(*size*), 1 <= size <= 255	varchar2(s char), 3 <= s <= 765
date	date
datetime	date
decimal(p,s), 1 <= p <= 38, 0 <= s <= 29	number(p,s), 1 <= p <= 38, 0 <= s <= 29
decimal(p,s), 39 <= p <= 65, 0 <= s <= 29	char(s char), 40 <= s <= 67
double	binary_double
float	binary_float
geomcollection	blob
geometry	blob
geometrycollection	blob
int	number(10)
int unsigned	number(10)
json	clob
linestring	blob
longblob	blob
longtext	clob
mediumblob	blob
mediumint	number(7)
mediumint unsigned	number(8)
mediumtext	clob

MySQL ソースデータ型	Oracle ターゲットデータ型
multilinestring	blob
multipoint	blob
multipolygon	blob
numeric(10,0)	number(10)
point	blob
polygon	blob
smallint	number(5)
smallint unsigned	number(5)
text	clob
time(*precision*), 0 <= p <= 6	char(s char), 10 <= s <= 17
timestamp(*precision*), 0 <= p <= 6	timestamp(*precision*), 0 <= p <= 6
tinyblob	raw(255)
tinyint	number(3)
tinyint unsigned	number(3)
tinytext	varchar2(255 char)
varbinary(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
varbinary(*size*), 2001 <= size <= 65535	long raw
varchar(*size*), 1 <= size <= 4000	varchar2(s char), 3 <= s <= 4000
varchar(*size*), 4001 <= size <= 21844	clob
year	number(4)

サポートされていないソースデータ型

一括取り込みデータベースでは、次の MySQL データ型はサポートされません。

- BLOB
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TEXT

- TINYBLOB
- TINYTEXT

データベース取り込みタスクで、JSON データ型のカラムを含むソーススキーマを指定する場合、タスクをデプロイすると、JSON カラムが無視され、ターゲットに対応するカラムは作成されません。サポートされていない他のデータ型の場合、データ型がデフォルトのマッピングに表示されていても、データベース取り込みジョブは null をプロパゲートします。

MySQL ソースと Snowflake ターゲット

次の表は、MySQL ソースと Snowflake ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

MySQL **ソースデータ型**	Snowflake **ターゲットデータ型**
bigint	number(19)
bigint unsigned	number(20)
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255
bit(*precision*), 1 <= p <= 64	binary(*size*), 1 <= size <= 8
blob	binary(65535)
char(*size*), 1 <= size <= 255	varchar(*size*), 4 <= size <= 1020
date	date
datetime	timestamp_ntz(0)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 29	number(p,s), 1 <= p <= 38, 0 <= s <= 29
decimal(p,s), 39 <= p <= 65, 0 <= s <= 29	char(*size*), 40 <= size <= 67
double	float
float	float
geomcollection	binary
geometry	binary
geometrycollection	binary
int	number(10)
int unsigned	number(10)
linestring	binary
longblob	binary
longtext	varchar

MySQL ソースデータ型	Snowflake ターゲットデータ型
mediumblob	binary
mediumint	number(7)
mediumint unsigned	number(8)
mediumtext	varchar
multilinestring	binary
multipoint	binary
multipolygon	binary
numeric(10,0)	integer
point	binary
polygon	binary
smallint	number(5)
smallint unsigned	number(5)
text	varchar(87380)
time(*precision*), 0 <= p <= 6	varchar(*size*), 10 <= size <= 17
timestamp(*precision*), 0 <= p <= 6	timestamp_ntz(*precision*), 0 <= p <= 6
tinyblob	binary(255)
tinyint	number(3)
tinyint unsigned	number(3)
tinytext	varchar(340)
varbinary(*size*), 1 <= size <= 65535	binary(*size*), 1 <= size <= 65535
varchar(*size*), 1 <= size <= 21844	varchar(*size*), 4 <= size <= 87376
year	number(4)

サポートされていないソースデータ型

一括取り込みデータベースでは、次の MySQL データ型はサポートされません。

- BLOB
- JSON
- LONGBLOB
- LONGTEXT

- MEDIUMBLOB
- MEDIUMTEXT
- TEXT
- TINYBLOB
- TINYTEXT

データベース取り込みタスクで、JSON データ型のカラムを含むソーススキーマを指定する場合、タスクをデプロイすると、JSON カラムが無視され、ターゲットに対応するカラムは作成されません。サポートされていない他のデータ型の場合、データ型がデフォルトのマッピングに表示されていても、データベース取り込みジョブは null をプロパゲートします。

Netezza ソースと Amazon Redshift ターゲット

次の表は、Netezza ソースと Amazon Redshift ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Netezza **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
bigint	bigint
boolean	boolean
byteint	smallint
char(*size*), 1 <= size <= 4096	character(*size*), 1 <= size <= 4096
char(*size*), 4097 <= size <= 64000	character varying(*size*), 4097 <= size <= 64000
date	date
double	double precision
integer	integer
interval	character varying(55)
nchar(*size*), 1 <= size <= 16000	character varying(*size*), 4 <= size <= 64000
numeric(38,38)	character varying(41)
numeric(p), 1 <= p <= 38	numeric(p,0), 1 <= p <= 38
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	numeric(p,s), 1 <= p <= 38, 1 <= s <= 37
nvarchar(*size*), 1 <= size <= 16000	character varying(*size*), 4 <= size <= 64000
real	real
smallint	smallint
st_geometry(p), 1 <= p <= 63001	binary varying(*size*), 1 <= size <= 63001
time	time without time zone

Netezza **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
timestamp	timestamp without time zone
timetz	timetz
varbinary(*size*), 1 <= size <= 64000	binary varying(*size*), 1 <= size <= 64000
varchar(*size*), 1 <= size <= 64000	character varying(*size*), 1 <= size <= 64000

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Netezza データ型はサポートされません。

- ST_GEOMETRY

注: このデータ型はデフォルトのマッピングに表示されますが、データベース取り込みジョブは、このデータ型を持つカラムにはデプロイできないか、null をプロパゲートします。

Netezza ソースと Databricks Delta ターゲット

次の表は、Netezza ソースと Databricks Delta ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Netezza **ソースデータ型**	Databricks Delta **ターゲットデータ型**
bigint	long
boolean	boolean
byteint	integer
char(*size*), 1 <= size <= 64000	string
date	string
double	double
integer	integer
interval	string
nchar(*size*), 1 <= size <= 16000	string
numeric(p,s), 1 <= p <= 38, 1 <= s <= 38	decimal(p,s), 1 <= p <= 38, 1 <= s <= 38
nvarchar(*size*), 1 <= size <= 16000	string
real	float
smallint	integer
st_geometry(*precision*), 1 <= p <= 63001	binary

Netezza **ソースデータ型**	Databricks Delta **ターゲットデータ型**
time	string
TIMESTAMP	TIMESTAMP
timetz	string
varbinary(*size*), 1 <= size <= 64000	binary
varchar(*size*), 1 <= size <= 64000	string

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Netezza データ型はサポートされません。

- 精度よりスケールが大きい NUMERIC
- ST_GEOMETRY

注: このデータ型はデフォルトのマッピングに表示されますが、データベース取り込みジョブは、このデータ型を持つカラムにはデプロイできないか、null をプロパゲートします。

Netezza ソースと Google BigQuery ターゲット

次の表は、Netezza ソースと Google BigQuery ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Netezza **ソースデータ型**	Google BigQuery **ターゲットデータ型**
bigint	int64
boolean	bool
byteint	int64
char(*size*), 1 <= size <= 64000	string
date	date
double	float64
integer	int64
interval	string
nchar(*size*), 1 <= size <= 16000	string
numeric(10,10)	bignumeric
numeric(11,s), 10 <= s <= 11	bignumeric
numeric(12,s), 10 <= s <= 12	bignumeric
numeric(13,s), 10 <= s <= 13	bignumeric

Netezza **ソースデータ型**	Google BigQuery **ターゲットデータ型**
numeric(14,s), 10 <= s <= 14	bignumeric
numeric(15,s), 10 <= s <= 15	bignumeric
numeric(16,s), 10 <= s <= 16	bignumeric
numeric(17,s), 10 <= s <= 17	bignumeric
numeric(18,s), 10 <= s <= 18	bignumeric
numeric(19,s), 10 <= s <= 19	bignumeric
numeric(20,s), 10 <= s <= 20	bignumeric
numeric(21,s), 10 <= s <= 21	bignumeric
numeric(22,s), 10 <= s <= 22	bignumeric
numeric(23,s), 10 <= s <= 23	bignumeric
numeric(24,s), 10 <= s <= 24	bignumeric
numeric(25,s), 10 <= s <= 25	bignumeric
numeric(26,s), 10 <= s <= 26	bignumeric
numeric(27,s), 10 <= s <= 27	bignumeric
numeric(28,s), 10 <= s <= 28	bignumeric
numeric(29,s), 10 <= s <= 29	bignumeric
numeric(38,s), 10 <= s <= 38	bignumeric
numeric(*precision*), 1 <= p <= 17	int64
numeric(*precision*), 19 <= p <= 29	numeric
numeric(*precision*), 30 <= p <= 38	bignumeric
numeric(p,s), 1 <= p <= 38, 1 <= s <= 9	numeric
numeric(p,s), 30 <= p <= 31, 1 <= s <= 30	bignumeric
numeric(p,s), 31 <= p <= 32, 1 <= s <= 31	bignumeric
numeric(p,s), 32 <= p <= 33, 1 <= s <= 32	bignumeric
numeric(p,s), 33 <= p <= 34, 1 <= s <= 33	bignumeric
numeric(p,s), 34 <= p <= 35, 1 <= s <= 34	bignumeric
numeric(p,s), 35 <= p <= 36, 1 <= s <= 35	bignumeric

Netezza **ソースデータ型**	Google BigQuery **ターゲットデータ型**
numeric(p,s), 36 <= p <= 37, 1 <= s <= 36	bignumeric
numeric(p,s), 37 <= p <= 38, 1 <= s <= 37	bignumeric
nvarchar(*size*), 1 <= size <= 16000	string
real	float64
smallint	int64
st_geometry(*precision*), 1 <= p <= 63001	bytes
time	time
timestamp	datetime
timetz	timestamp
varbinary(*size*), 1 <= size <= 64000	bytes
varchar(*size*), 1 <= size <= 64000	string

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Netezza データ型はサポートされません。

- ST_GEOMETRY

注: このデータ型はデフォルトのマッピングに表示されますが、データベース取り込みジョブは、このデータ型を持つカラムにはデプロイできないか、null をプロパゲートします。

Netezza ソースと Microsoft Azure Synapse Analytics ターゲット

次の表は、Netezza ソースと Microsoft Azure Synapse Analytics ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Netezza **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
bigint	bigint
boolean	bit
byteint	smallint
char(*size*), 1 <= size <= 8000	char(*size*), 1 <= size <= 8000
char(*size*), 9000 <= size <= 64000	nvarchar(max)
date	date
double	float

Netezza ソースデータ型	Microsoft Azure Synapse Analytics ターゲットデータ型
integer	int
interval	varchar(55)
nchar(*size*), 1 <= size <= 3100	nchar(*size*), 1 <= size <= 3100
nchar(*size*), 4100 <= size <= 16000	nvarchar(max)
numeric(p,s), 1 <= p <= 38, 1 <= s <= 38	decimal(p,s), 1 <= p <= 38, 1 <= s <= 38
nvarchar(*size*), 1 <= size <= 16000	nvarchar(*size*), 1 <= size <= max
real	real
smallint	smallint
st_geometry(*precision*), 1 <= p <= 63001	varbinary(*size*), 1 <= size <= max
time	time(6)
TIMESTAMP	datetime2(6)
timetz	datetimeoffset(6)
varbinary(*size*), 1 <= size <= 64000	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 64000	varchar(*size*), 1 <= size <= max

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Netezza データ型はサポートされません。

- ST_GEOMETRY

注: このデータ型はデフォルトのマッピングに表示されますが、データベース取り込みジョブは、このデータ型を持つカラムにはデプロイできないか、null をプロパゲートします。

Netezza ソースと Microsoft SQL Server ターゲット

次の表に、Netezza ソースと Microsoft SQL Server ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示します。

Netezza ソースデータ型	Microsoft SQL Server ターゲットデータ型
bigint	bigint
boolean	bit
byteint	smallint
char(*size*), 1 <= size <= 64000	varchar(*size*), 1 <= size <= max

Netezza ソースデータ型	Microsoft SQL Server ターゲットデータ型
date	date
double	float
integer	int
interval	varchar(55)
nchar(*size*), 1 <= size <= 16000	nvarchar(*size*), 4 <= size <= max
numeric(p,s), 1 <= p <= 38, 1 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
nvarchar(*size*), 1 <= size <= 16000	nvarchar(*size*), 4 <= size <= max
real	real
smallint	smallint
st_geometry(*precision*), 1 <= p <= 63001	varbinary(*size*), 1 <= size <= max
time	time(6)
timestamp	datetime2(6)
timetz	datetimeoffset(6)
varbinary(*size*), 1 <= size <= 64000	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 64000	varchar(*size*), 1 <= size <= max

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Netezza データ型はサポートされません。

- ST_GEOMETRY

注: このデータ型はデフォルトのマッピングに表示されますが、データベース取り込みジョブは、このデータ型を持つカラムにはデプロイできないか、null をプロパゲートします。

Netezza ソースと Oracle ターゲット

次の表に、Netezza ソースと Oracle ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示します。

Netezza ソースデータ型	Oracle ターゲットデータ型
bigint	number(19)
boolean	char(1 char)
byteint	number(3)

Netezza ソースデータ型	Oracle ターゲットデータ型
char(*size*), 1 <= size <= 3001	varchar2(s char), 1 <= s <= 3001
char(*size*), 4096 <= size <= 64000	clob
date	date
double	binary_double
integer	number(10)
interval	varchar2(55 char)
nchar(*size*), 1 <= size <= 901	nvarchar2(s char), 1 <= s <= 901
nchar(*size*), 1024 <= size <= 1100	char(2000 char)
nchar(*size*), 2100 <= size <= 16000	clob
numeric(p,s), 1 <= p <= 38, 1 <= s <= 38	number(p,s), 1 <= p <= 38, 1 <= s <= 38
nvarchar(1)	nvarchar2(1 char)
nvarchar(1001)	char(2000 char)
nvarchar(*size*), 2001 <= s <= 16000	clob
real	binary_float
smallint	number(5)
st_geometry(*precision*), 1 <= p <= 1001	raw(*size*), 1 <= size <= 1001
st_geometry(*precision*), 2001 <= p <= 63001	blob
time	timestamp(6)
timestamp	timestamp(6)
timetz	timestamp(*precision*) with time zone, 6 <= p <= null
varbinary(*size*), 1 <= size <= 1001	raw(*size*), 1 <= size <= 1001
varbinary(*size*), 2001 <= size <= 64000	blob
varchar(*size*), 1 <= size <= 3001	varchar2(s char), 1 <= s <= 3001
varchar(*size*), 4001 <= size <= 64000	clob

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Netezza データ型はサポートされません。

- ST_GEOMETRY

注: このデータ型はデフォルトのマッピングに表示されますが、データベース取り込みジョブは、このデータ型を持つカラムにはデプロイできないか、null をプロパゲートします。

Netezza ソースと Snowflake ターゲット

次の表は、Netezza ソースと Snowflake ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Netezza **ソースデータ型**	Snowflake **ターゲットデータ型**
bigint	integer
boolean	boolean
byteint	integer
char(*size*), 1 <= size <= 64000	char(*size*), 1 <= size <= 64000
date	date
double	float
integer	integer
interval	varchar(55)
nchar(*size*), 1 <= size <= 16000	char(*size*), 4 <= size <= 64000
numeric(38,38)	char(41)
numeric(*precision*), 1 <= p <= 38	integer
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
nvarchar(*size*), 1 <= size <= 16000	varchar(*size*), 4 <= size <= 64000
real	float
smallint	integer
st_geometry(*precision*), 1 <= p <= 63001	binary(*size*), 1 <= size <= 63001
time	time(6)
TIMESTAMP	timestamp_ntz(6)
timetz	timestamp_tz(6)
varbinary(*size*), 1 <= size <= 64000	binary(*size*), 1 <= size <= 64000
varchar(*size*), 1 <= size <= 64000	varchar(*size*), 1 <= size <= 64000

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Netezza データ型はサポートされません。

- ST_GEOMETRY

注: このデータ型はデフォルトのマッピングに表示されますが、データベース取り込みジョブは、このデータ型を持つカラムにはデプロイできないか、null をプロパゲートします。

Oracle ソースと Amazon Redshift ターゲット

次の表は、Oracle ソースと Amazon Redshift ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Oracle **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
binary_double	float8
binary_float	float4
blob	binary varying(1024000)
char(s byte), 1 <= size <= 2000	character varying(*size*), 4 <= size <= 2000
char(s char), 1 <= size <= 2000	character varying(*size*), 4 <= size <= 8000
clob	character varying(65535)
date	timestamp without time zone
float(*precision*), 1 <= p <= 126	character varying(255)
integer	character varying(255)
long raw	varbyte
long(2147483648 byte)	character varying(65535)
nchar(s char), 1 <= size <= 2000	character varying(*size*), 4 <= size <= 8000
nclob	character varying(65535)
number	number
number(*,s), -84 <= s <= 127	character varying(255)
number(38,s), 0 <= s <= 37	numeric(38,s), 0 <= s <= 37
number(p,s), 1 <= p <= 38, -37 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
number(p,s), 1 <= p <= 38, -84 <= s <= 127	character varying(*size*), 40 <= size <= 130
number(p,s), 10 <= p <= 38, -28 <= s <= 37	numeric(p,s), 10 <= p <= 38, 0 <= s <= 37
number(p,s), 11 <= p <= 38, -27 <= s <= 37	numeric(p,s), 11 <= p <= 38, 0 <= s <= 37
number(p,s), 12 <= p <= 38, -26 <= s <= 37	numeric(p,s), 12 <= p <= 38, 0 <= s <= 37

Oracle **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
number(p,s), 13 <= p <= 38, -25 <= s <= 37	numeric(p,s), 13 <= p <= 38, 0 <= s <= 37
number(p,s), 14 <= p <= 38, -24 <= s <= 37	numeric(p,s), 14 <= p <= 38, 0 <= s <= 37
number(p,s), 15 <= p <= 38, -23 <= s <= 37	numeric(p,s), 15 <= p <= 38, 0 <= s <= 37
number(p,s), 16 <= p <= 38, -22 <= s <= 37	numeric(p,s), 16 <= p <= 38, 0 <= s <= 37
number(p,s), 17 <= p <= 38, -21 <= s <= 37	numeric(p,s), 17 <= p <= 38, 0 <= s <= 37
number(p,s), 18 <= p <= 38, -20 <= s <= 37	numeric(p,s), 18 <= p <= 38, 0 <= s <= 37
number(p,s), 19 <= p <= 38, -19 <= s <= 37	numeric(p,s), 20 <= p <= 38, 0 <= s <= 37
number(p,s), 2 <= p <= 38, -36 <= s <= 37	numeric(p,s), 2 <= p <= 38, 0 <= s <= 37
number(p,s), 21 <= p <= 38, -17 <= s <= 37	numeric(p,s), 21 <= p <= 38, 0 <= s <= 37
number(p,s), 22 <= p <= 38, -16 <= s <= 37	numeric(p,s), 22 <= p <= 38, 0 <= s <= 37
number(p,s), 23 <= p <= 38, -15 <= s <= 37	numeric(p,s), 23 <= p <= 38, 0 <= s <= 37
number(p,s), 24 <= p <= 38, -14 <= s <= 37	numeric(p,s), 24 <= p <= 38, 0 <= s <= 37
number(p,s), 25 <= p <= 38, -13 <= s <= 37	numeric(p,s), 25 <= p <= 38, 0 <= s <= 37
number(p,s), 26 <= p <= 38, -12 <= s <= 37	numeric(p,s), 26 <= p <= 38, 0 <= s <= 37
number(p,s), 27 <= p <= 38, -11 <= s <= 37	numeric(p,s), 27 <= p <= 38, 0 <= s <= 37
number(p,s), 28 <= p <= 38, -10 <= s <= 37	numeric(p,s), 28 <= p <= 38, 0 <= s <= 37
number(p,s), 29 <= p <= 38, -9 <= s <= 37	numeric(p,s), 29 <= p <= 38, 0 <= s <= 37
number(p,s), 3 <= p <= 38, -35 <= s <= 37	numeric(p,s), 3 <= p <= 38, 0 <= s <= 37
number(p,s), 30 <= p <= 38, -8 <= s <= 37	numeric(p,s), 30 <= p <= 38, 0 <= s <= 37
number(p,s), 31 <= p <= 38, -7 <= s <= 37	numeric(p,s), 31 <= p <= 38, 0 <= s <= 37
number(p,s), 32 <= p <= 38, -6 <= s <= 37	numeric(p,s), 32 <= p <= 38, 0 <= s <= 37
number(p,s), 33 <= p <= 38, -5 <= s <= 37	numeric(p,s), 33 <= p <= 38, 0 <= s <= 37
number(p,s), 34 <= p <= 38, -4 <= s <= 37	numeric(p,s), 34 <= p <= 38, 0 <= s <= 37
number(p,s), 35 <= p <= 38, -3 <= s <= 37	numeric(p,s), 35 <= p <= 38, 0 <= s <= 37
number(p,s), 36 <= p <= 38, -2 <= s <= 37	numeric(p,s), 36 <= p <= 38, 0 <= s <= 37
number(p,s), 37 <= p <= 38, -1 <= s <= 37	numeric(p,s), 37 <= p <= 38, 0 <= s <= 37
number(p,s), 4 <= p <= 38, -34 <= s <= 37	numeric(p,s), 4 <= p <= 38, 0 <= s <= 37

Oracle **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
number(p,s), 5 <= p <= 38, -33 <= s <= 37	numeric(p,s), 5 <= p <= 38, 0 <= s <= 37
number(p,s), 6 <= p <= 38, -32 <= s <= 37	numeric(p,s), 6 <= p <= 38, 0 <= s <= 37
number(p,s), 7 <= p <= 38, -31 <= s <= 37	numeric(p,s), 7 <= p <= 38, 0 <= s <= 37
number(p,s), 8 <= p <= 38, -30 <= s <= 37	numeric(p,s), 8 <= p <= 38, 0 <= s <= 37
number(p,s), 9 <= p <= 38, -29 <= s <= 37	numeric(p,s), 9 <= p <= 38, 0 <= s <= 37
nvarchar2(s char), 1 <= size <= 4000	character varying(*size*), 4 <= size <= 16000
raw(*size*), 1 <= size <= 2000	varbyte
rowid	character varying(18)
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp with time zone
timestamp(*precision*) with time zone, 7 <= p <= 9	character varying(29)
timestamp(*precision*), 1 <= p <= 6	timestamp without time zone
timestamp(*precision*), 7 <= p <= 9	character varying(*size*), 27 <= s <= 29
varchar2(s byte), 1 <= size <= 4000	character varying(*size*), 4 <= size <= 4000
varchar2(s char), 1 <= size <= 4000	character varying(*size*), 4 <= size <= 16000

LOB の制限事項

データベース取り込み初期ロードジョブで、Oracle BLOB、CLOB、および NCLOB カラムから Amazon Redshift ターゲットにデータをレプリケートできます。LOB カラムデータは、ターゲットに書き込まれる前に切り詰められる場合があります。切り詰めポイントは、データ型とターゲットタイプによって異なります。Amazon Redshift ターゲットを使用した初期ロードジョブの場合、BLOB データは 1024000 バイトに切り詰められ、CLOB および NCLOB データは 65535 バイトに切り詰められます。詳細については、「ソースの設定」(ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Oracle ソースデータ型はサポートされません。

- ANYTYPE、ANYDATA、ANYDATASET などの「ANY タイプ」
- 拡張タイプ
- INTERVAL
- JSON
- LOB（初期ロードジョブの BLOB、CLOB、および NCLOB を除く）
- TIMESTAMP WITH LOCAL TIME ZONE
- UROWID
- XMLTYPE、URI タイプ、URIFactory パッケージのサブタイプなどの XML 定義タイプ
- SDO_GEOMETRY などの空間タイプ

- OBJECT、REF、VARRAY、ネストされたテーブルタイプなどのユーザー定義タイプ

サポートされていないデータ型を持つソースカラムは、ターゲット定義から除外されます。

Oracle ソースと Databricks Delta ターゲット

次の表は、Oracle ソースと Databricks Delta ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Oracle **ソースデータ型**	Databricks Delta **ターゲットデータ型**
binary_double	double
binary_float	float
blob	binary
char(s byte), 1 <= size <= 2000	string
char(s char), 1 <= size <= 2000	string
clob	string
date	timestamp
float(*precision*), 1 <= p <= 126	string
integer	string
long raw	binary
long(2147483648 byte)	string
nchar(s char), 1 <= size <= 2000	string
nclob	string
number	string
number(*,s), -84 <= s <= 127	string
number(p,s), 1 <= p <= 38, -37 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
number(p,s), 1 <= p <= 38, -84 <= s <= 127	string
nvarchar2(s char), 1 <= size <= 4000	string
raw(*size*), 1 <= size <= 2000	binary
rowid	string
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*) with time zone, 7 <= p <= 9	string
timestamp(*precision*), 1 <= p <= 6	timestamp

Oracle **ソースデータ型**	Databricks Delta **ターゲットデータ型**
timestamp(*precision*), 7 <= p <= 9	string
varchar2(s byte), 1 <= size <= 4000	string
varchar2(s char), 1 <= size <= 4000	string

LOB の制限事項

データベース取り込み初期ロードジョブで、Oracle BLOB、CLOB、および NCLOB カラムから Databricks Delta ターゲットにデータをレプリケートできます。LOB カラムデータは、ターゲットに書き込まれる前に切り詰められる場合があります。すべての Oracle LOB データ型とこのターゲットで、切り詰めポイントは 16777216 バイトです。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Oracle ソースデータ型はサポートされません。

- ANYTYPE、ANYDATA、ANYDATASET などの「ANY タイプ」
- 拡張タイプ
- INTERVAL
- JSON
- LOB（初期ロードジョブの BLOB、CLOB、および NCLOB を除く）
- 精度よりスケールが大きい NUMBER
- TIMESTAMP WITH LOCAL TIME ZONE
- UROWID
- XMLTYPE、URI タイプ、URIFactory パッケージのサブタイプなどの XML 定義タイプ
- SDO_GEOMETRY などの空間タイプ
- OBJECT、REF、VARRAY、ネストされたテーブルタイプなどのユーザー定義タイプ

サポートされていないデータ型を持つソースカラムは、ターゲット定義から除外されます。

Oracle ソースと Google BigQuery ターゲット

次の表は、Oracle ソースと Google BigQuery ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Oracle **ソースデータ型**	Google BigQuery **ターゲットデータ型**
binary_double	float64
binary_float	float64
blob	bytes
char(s byte), 1 <= size <= 2000	string
char(s char), 1 <= size <= 2000	string

Oracle **ソースデータ型**	Google BigQuery **ターゲットデータ型**
clob	string
date	datetime
float(*precision*), 1 <= p <= 126	string
integer	string
long raw	bytes
long(2147483648 byte)	string
nchar(s char), 1 <= size <= 2000	string
nclob	string
number	string
number(*,s), -84 <= s <= 127	string
number(1,-38)	bignumeric
number(1,s), -37 <= s <= -29	bignumeric
number(1,s), -84 <= s <= -39	string
number(10,-29)	bignumeric
number(11,-28)	bignumeric
number(12,-27)	bignumeric
number(13,-26)	bignumeric
number(14,-25)	bignumeric
number(15,-24)	bignumeric
number(16,-23)	bignumeric
number(17,-22)	bignumeric
number(18,-21)	bignumeric
number(19,-20)	bignumeric
number(2,-37)	bignumeric
number(20,-19)	bignumeric
number(21,-18)	bignumeric
number(22,-17)	bignumeric

Oracle ソースデータ型	Google BigQuery ターゲットデータ型
number(23,-16)	bignumeric
number(24,-15)	bignumeric
number(25,-14)	bignumeric
number(26,-13)	bignumeric
number(27,-12)	bignumeric
number(28,-11)	bignumeric
number(29,-10)	bignumeric
number(3,-36)	bignumeric
number(30,-9)	bignumeric
number(31,-8)	bignumeric
number(32,-7)	bignumeric
number(33,-6)	bignumeric
number(34,-5)	bignumeric
number(35,-4)	bignumeric
number(36,-3)	bignumeric
number(37,-2)	bignumeric
number(38,-1)	bignumeric
number(38,s), 10 <= s <= 38	bignumeric
number(4,-35)	bignumeric
number(5,-34)	bignumeric
number(6,-33)	bignumeric
number(7,-32)	bignumeric
number(8,-31)	bignumeric
number(9,-30)	bignumeric
number(p,s), 1 <= p <= 38, -28 <= s <= 9	numeric
number(p,s), 1 <= p <= 38, -36 <= s <= 38	bignumeric
number(p,s), 1 <= p <= 38, -84 <= s <= 127	string

Oracle **ソースデータ型**	Google BigQuery **ターゲットデータ型**
number(p,s), 2 <= p <= 3, -35 <= s <= 38	bignumeric
number(p,s), 2 <= p <= 3, -84 <= s <= 127	string
nvarchar2(s char), 1 <= s <= 4000	string
raw(*size*), 1 <= size <= 2000	bytes
rowid	string
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*) with time zone, 7 <= p <= 9	string
timestamp(*precision*), 1 <= p <= 6	datetime
timestamp(*precision*), 7 <= p <= 9	string
varchar2(s byte), 1 <= size <= 4000	string
varchar2(s char), 1 <= size <= 4000	string

LOB の制限事項

データベース取り込み初期ロードジョブで、Oracle BLOB、CLOB、および NCLOB カラムから Google BigQuery ターゲットにデータをレプリケートできます。LOB カラムデータは、ターゲットに書き込まれる前に切り詰められる場合があります。すべての Oracle LOB データ型で、切り詰めポイントは 8388608 バイトです。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Oracle ソースデータ型はサポートされません。

- ANYTYPE、ANYDATA、ANYDATASET などの「ANY タイプ」
- 拡張タイプ
- INTERVAL
- JSON
- LOB（初期ロードジョブの BLOB、CLOB、および NCLOB を除く）
- TIMESTAMP WITH LOCAL TIME ZONE
- UROWID
- XMLTYPE、URI タイプ、URIFactory パッケージのサブタイプなどの XML 定義タイプ
- SDO_GEOMETRY などの空間タイプ
- OBJECT、REF、VARRAY、ネストされたテーブルタイプなどのユーザー定義タイプ

サポートされていないデータ型を持つソースカラムは、ターゲット定義から除外されます。

Oracle ソースと Microsoft Azure Synapse Analytics ターゲット

次の表は、Oracle ソースと Microsoft Azure Synapse Analytics ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Oracle **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
binary_double	float
binary_float	real
blob	varbinary(max)
char(s byte), 1 <= size <= 2000	varchar(*size*), 4 <= size <= 2000
char(s char), 1 <= size <= 2000	varchar(*size*), 4 <= size <= 8000
clob	varchar(max)
date	datetime2(0)
float(*precision*), 1 <= p <= 126	varchar（255）
integer	varchar（255）
long raw	varbinary(max)
long(2147483648 byte)	varchar(max)
nchar(s char), 1 <= size <= 2000	nchar(*size*), 1 <= size <= 2000
nclob	nvarchar(max)
number	varchar（255）
number(*,s), -84 <= s <= 127	varchar（255）
number(p,s), 1 <= p <= 38, -37 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
number(p,s), 1 <= p <= 38, -84 <= s <= 127	varchar(*size*), 40 <= size <= 130
nvarchar2(s char), 1 <= s <= 4000	nvarchar(*size*), 1 <= size <= 4000
raw(*size*), 1 <= s <= 2000	varbinary(*size*), 1 <= size <= 2000
rowid	varchar(18)
timestamp(*precision*) with time zone, 0 <= p <= 7	datetimeoffset(*precision*), 0 <= p <= 7
timestamp(*precision*) with time zone, 8 <= p <= 9	char(*size*), 68 <= size <= 69
timestamp(*precision*), 1 <= p <= 7	datetime2(*precision*), 1 <= p <= 7
timestamp(*precision*), 8 <= p <= 9	char(*size*), 28 <= size <= 29

Oracle **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
varchar2(s byte), 1 <= size <= 4000	varchar(*size*), 4 <= size <= 4000
varchar2(s char), 1 <= size <= 4000	varchar(*size*), 4 <= size <= max

LOB の制限事項

データベース取り込み初期ロードジョブで、Oracle BLOB、CLOB、および NCLOB カラムから Microsoft Azure Synapse Analytics ターゲットにデータをレプリケートできます。LOB カラムデータは、ターゲットに書き込まれる前に切り詰められる場合があります。Azure Synapse Analytics ターゲットを使用した初期ロードジョブの場合、BLOB データは 1000000 バイトに切り詰められ、CLOB および NCLOB データは 500000 バイトに切り詰められます。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

一括取り込みデータベースでは、どのロードタイプでも次の Oracle ソースデータ型はサポートされません。

- ANYTYPE、ANYDATA、ANYDATASET などの「ANY タイプ」
- 拡張タイプ
- INTERVAL
- JSON
- LOB（初期ロードジョブの BLOB、CLOB、および NCLOB を除く）
- TIMESTAMP WITH LOCAL TIME ZONE
- UROWID
- XMLTYPE、URI タイプ、URIFactory パッケージのサブタイプなどの XML 定義タイプ
- SDO_GEOMETRY などの空間タイプ
- OBJECT、REF、VARRAY、ネストされたテーブルタイプなどのユーザー定義タイプ

サポートされていないデータ型を持つソースカラムは、ターゲット定義から除外されます。

Oracle ソースと Microsoft SQL Server ターゲット

次の表は、Oracle ソースと SQL Server ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Oracle **ソースデータ型**	SQL Server **ターゲットデータ型**
binary_double	float
binary_float	real
blob	varbinary
char	varchar
clob	varchar

Oracle ソースデータ型	SQL Server ターゲットデータ型
date	datetime2
float	varchar
long raw	varbinary
long	varchar
nchar	nvarchar
nclob	nvarchar
number	以下のいずれかのタイプです。 - decimal, 1 <= p <= 38 and s >= 0 and s <= p - Money, s > p or s < 0、桁数とスケールを格納可能 - bigint, s <= 0、ストレージサイズを格納可能 - char
nvarchar2	nvarchar
raw	varbinary
rowid	varchar
timestamp	以下のいずれかのタイプです。 - datetime2、小数の秒数 <=7 - char
timestamp with time zone	以下のいずれかのタイプです。 - datetimeoffset、小数の秒数 <=7 - char
varchar2	varchar

LOB の制限事項

データベース取り込み初期ロードジョブで、Oracle BLOB、CLOB、および NCLOB カラムから SQL Server ターゲットにデータをレプリケートできます。LOB カラムデータは、ターゲットに書き込まれる前に切り詰められる場合があります。切り詰めポイントは、データ型とターゲットタイプによって異なります。SQL Server ターゲットを使用した初期ロードジョブの場合、BLOB データと CLOB データは 16777216 バイトに切り詰められ、NCLOB データは 33554432 バイトに切り詰められます。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

一括取り込みデータベースでは、どのロードタイプでも次の Oracle ソースデータ型はサポートされません。

- ANYTYPE、ANYDATA、ANYDATASET などの「ANY タイプ」
- 拡張タイプ
- INTERVAL
- JSON
- LOB（初期ロードジョブの BLOB、CLOB、および NCLOB を除く）

- TIMESTAMP WITH LOCAL TIME ZONE
- UROWID
- XMLTYPE、URI タイプ、URIFactory パッケージのサブタイプなどの XML 定義タイプ
- SDO_GEOMETRY などの空間タイプ
- OBJECT、REF、VARRAY、ネストされたテーブルタイプなどのユーザー定義タイプ

サポートされていないデータ型を持つソースカラムは、ターゲット定義から除外されます。

Oracle ソースと Oracle ターゲット

次の表は、Oracle ソースと Oracle ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Oracle **ソースデータ型**	Oracle **ターゲットデータ型**
binary_double	binary_double
binary_float	binary_float
blob	blob
char(s byte), 1 <= size <= 2000	char(s byte), 1 <= size <= 2000
char(s char), 1 <= size <= 2000	char(s char), 1 <= size <= 2000
clob	clob
date	date
float(*precision*), 1 <= p <= 126	float(*precision*), 1 <= p <= 126
integer	integer
long raw	long raw
long(2147483648 byte)	long(2147483648 byte)
nchar(s char), 1 <= size <= 2000	nchar(s char), 1 <= size <= 2000
nclob	nclob
number	number
number(*,s), -84 <= s <= 127	number(*,s), -84 <= s <= 127
number(p,s), 1 <= p <= 38, -84 <= s <= 127	number(p,s), 1 <= p <= 38, -84 <= s <= 127
nvarchar2(s char), 1 <= size <= 4000	nvarchar2(s char), 1 <= size <= 4000
raw(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
rowid	rowid
timestamp(*precision*) with time zone, 0 <= p <= 9	timestamp(*precision*) with time zone, 0 <= p <= 9

Oracle **ソースデータ型**	Oracle **ターゲットデータ型**
timestamp(*precision*), 1 <= p <= 9	timestamp(*precision*), 1 <= p <= 9
varchar2(s byte), 1 <= size <= 4000	varchar2(s byte), 1 <= size <= 4000
varchar2(s char), 1 <= size <= 4000	varchar2(s char), 1 <= size <= 4000

LOB の制限事項

データベース取り込み初期ロードジョブで、Oracle BLOB、CLOB、および NCLOB カラムから Oracle ターゲットにデータをレプリケートできます。LOB カラムデータは、ターゲットに書き込まれる前に切り詰められる場合があります。すべての Oracle LOB データ型とこのターゲットで、切り詰めポイントは 16777216 バイトです。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

一括取り込みデータベースでは、どのロードタイプでも次の Oracle ソースデータ型はサポートされません。

- ANYTYPE、ANYDATA、ANYDATASET などの「ANY タイプ」
- 拡張タイプ
- INTERVAL
- JSON
- LOB（初期ロードジョブの BLOB、CLOB、および NCLOB を除く）
- TIMESTAMP WITH LOCAL TIME ZONE
- UROWID
- XMLTYPE、URI タイプ、URIFactory パッケージのサブタイプなどの XML 定義タイプ
- SDO_GEOMETRY などの空間タイプ
- OBJECT、REF、VARRAY、ネストされたテーブルタイプなどのユーザー定義タイプ

サポートされていないデータ型を持つソースカラムは、ターゲット定義から除外されます。

Oracle ソースと PostgreSQL ターゲット

次の表は、Oracle ソースと PostgreSQL ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Oracle **ソースデータ型**	PostgreSQL **ターゲットデータ型**
binary_double	double precision
binary_float	real
blob	bytea
char(s byte), 1 <= s <= 2000	character varying(*size*), 1 <= size <= 500
char(s char), 1 <= s <= 2000	character varying(*size*), 1 <= size <= 2000
clob	text

Oracle **ソースデータ型**	PostgreSQL **ターゲットデータ型**
date	timestamp(*precision*) without time zone, 0 <= p <= null
float(*precision*), 1 <= p <= 126	character varying(255)
integer	character varying(255)
long raw	bytea
long(2147483648 byte)	text
nchar(s char), 1 <= s <= 2000	character varying(*size*), 1 <= s <= 2000
nclob	text
number	character varying(255)
number(*,s), -84 <= s <= 127	character varying(255)
number(p,s), 1 <= p <= 38, -84 <= s <= 127	numeric(p,s), 0 <= p <= 127, 1 <= s <= 127
nvarchar2(s char), 1 <= s <= 4000	character varying(*size*), 1 <= s <= 4000
raw(*size*), 1 <= s <= 2000	bytea
rowid	character(18)
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp(*precision*) with time zone, 0 <= p <= 6
timestamp(*precision*) with time zone, 7 <= p <= 9	character varying(*size*), 67 <= size <= 69
timestamp(*precision*), 1 <= p <= 6	timestamp(*precision*) without time zone, 1 <= p <= 6
timestamp(*precision*), 7 <= p <= 9	character varying(*size*), 27 <= size <= 29
varchar2(s byte), 1 <= s <= 4000	character varying(*size*), 1 <= size <= 1000
varchar2(s char), 1 <= s <= 4000	character varying(*size*), 1 <= size <= 4000

サポートされていないソースデータ型

一括取り込みデータベースでは、どのロードタイプでも次の Oracle ソースデータ型はサポートされません。

- ANYTYPE、ANYDATA、ANYDATASET などの「ANY タイプ」
- 拡張タイプ
- INTERVAL
- JSON
- LOB
- TIMESTAMP WITH LOCAL TIME ZONE
- UROWID
- XMLTYPE、URI タイプ、URIFactory パッケージのサブタイプなどの XML 定義タイプ

- SDO_GEOMETRY などの空間タイプ
- OBJECT、REF、VARRAY、ネストされたテーブルタイプなどのユーザー定義タイプ

サポートされていないデータ型を持つソースカラムは、ターゲット定義から除外されます。

Oracle ソースと Snowflake ターゲット

次の表は、Oracle ソースと Snowflake ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Oracle **ソースデータ型**	Snowflake **ターゲットデータ型**
binary_double	double
binary_float	double
blob	binary
char(s byte), 1 <= size <= 2000	varchar(4)
char(s char), 1 <= size <= 2000	varchar(4)
clob	varchar
date	timestamp_ntz(0)
float(*precision*), 1 <= p <= 126	varchar（255）
integer	varchar（255）
long raw	binary
long（2147483648 バイト）	varchar（65535）
nchar(s char), 1 <= size <= 2000	varchar(4)
nclob	varchar
number	number
number(*,s), -84 <= s <= 127	char(255)
number(p,s), 1 <= p <= 38, -37 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
number(p,s), 1 <= p <= 38, -84 <= s <= 127	char(*size*), 40 <= size <= 130
nvarchar2(s char), 1 <= size <= 4000	varchar(4)
raw(*size*), 1 <= size <= 2000	binary(*size*), 1 <= size <= 2000
rowid	varchar(18)
timestamp(*precision*) with time zone, 0 <= p <= 9	timestamp_tz(*precision*), 0 <= p <= 9
timestamp(*precision*), 1 <= p <= 9	timestamp_ntz(*precision*), 1 <= p <= 9

Oracle **ソースデータ型**	Snowflake **ターゲットデータ型**
varchar2(s byte), 1 <= size <= 4000	varchar(*size*), 4 <= size <= 4000
varchar2(s char), 1 <= size <= 4000	varchar(*size*), 4 <= size <= 16000

LOB の制限事項

データベース取り込み初期ロードジョブで、Oracle BLOB、CLOB、および NCLOB カラムから Snowflake ターゲットにデータをレプリケートできます。LOB カラムデータは、ターゲットに書き込まれる前に切り詰められる場合があります。Snowflake ターゲットを使用した初期ロードジョブの場合、BLOB データは 8388608 バイトに切り詰められ、CLOB および NCLOB データは 16777216 バイトに切り詰められます。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

一括取り込みデータベースでは、どのロードタイプでも次の Oracle ソースデータ型はサポートされません。

- ANYTYPE、ANYDATA、ANYDATASET などの「ANY タイプ」
- 拡張タイプ
- INTERVAL
- JSON
- LOB（初期ロードジョブの BLOB、CLOB、および NCLOB を除く）
- TIMESTAMP WITH LOCAL TIME ZONE
- UROWID
- XMLTYPE、URI タイプ、URIFactory パッケージのサブタイプなどの XML 定義タイプ
- SDO_GEOMETRY などの空間タイプ
- OBJECT、REF、VARRAY、ネストされたテーブルタイプなどのユーザー定義タイプ

サポートされていないデータ型を持つソースカラムは、ターゲット定義から除外されます。

PostgreSQL ソースと Amazon Redshift ターゲット

次の表は、PostgreSQL ソースと Amazon Redshift ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

PostgreSQL **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
bigint	int8
bit varying(1)	boolean
bit varying(*precision*), 2 <= p <= 83886080	binary varying(*size*), 1 <= *size* <= 1024000
bit(1)	boolean
bit(*precision*), 2 <= p <= 83886080	binary varying(*size*), 1 <= *size* <= 1024000
boolean	boolean

PostgreSQL **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
box	binary varying(115)
character varying(*size*), 1 <= size <= 10485760	character varying(*size*), 4 <= *size* <= 65535
character(*size*), 1 <= size <= 10485760	character varying(*size*), 4 <= *size* <= 65535
cidr	character varying(45)
circle	binary varying(87)
date	date
daterange	character varying(29)
double precision	double precision
inet	character varying(45)
int4range	character varying(25)
int8range	character varying(43)
integer	integer
json	super
jsonb	super
line	binary varying(85)
lseg	binary varying(117)
macaddr	character varying(17)
macaddr8	character varying(23)
money	numeric(20,2)
numeric	character varying(65535)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
numeric(p,s), 38 <= p <= 1000, 38 <= s <= 1000	character varying(*size*), 41 <= *size* <= 1003
numrange	character varying(65535)
path	binary varying(1024000)
point	binary varying(57)
polygon	binary varying(1024000)
real	real

PostgreSQL **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
smallint	smallint
time(*precision*) with time zone, 0 <= p <= 6	timetz
time(*precision*) without time zone, 0 <= p <= 6	time
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp with time zone
timestamp(*precision*) without time zone, 0 <= p <= 6	timestamp
tsrange	character varying(63)
tstzrange	character varying(75)
uuid	character(36)
xml	character varying(65535)

LOB の制限事項

タスクウィザードの［**ソース**］ページにある［**詳細**］の下の［**LOB を含める**］オプションを指定した場合、いずれかのロードタイプを使用するデータベース取り込み初期ロードジョブと増分ロードジョブは、PostgreSQL BYTEA、TEXT、XML、およびその他のラージオブジェクトカラムから Amazon Redshift ターゲットにデータをレプリケートできます。LOB カラムデータは、LOB タイプによって異なるバイト制限よりもサイズが大きい場合、ターゲットに書き込まれる前に切り詰められます。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

初期ロードジョブの場合、一括取り込みデータベースでは次の PostgreSQL データ型はサポートされていません。

- ABSTIME
- 配列型
- NAME
- オブジェクト識別子型
- PG_LSN
- RELTIME
- テキスト検索型:

  - TSQUERY

  - TSVECTOR
- ユーザー定義型

増分ロードジョブの場合、一括取り込みデータベースでは、初期ロードジョブでサポートされていないものに加えて、次の PostgreSQL データ型はサポートされていません。

- 空間タイプ

  - Box

  - Circle

- Line
- LSeg
- Path
- Point
- Polygon

- 無制限のさまざまなタイプ

データベース統合ジョブは、これらのデータ型を持つカラムにはデプロイしたり null をプロパゲートしたりすることはできません。

PostgreSQL ソースと Databricks Delta ターゲット

次の表は、PostgreSQL ソースと Databricks Delta ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

PostgreSQL **ソースデータ型**	Databricks Delta Lake **ターゲットデータ型**
bigint	bigint
bit varying(*precision*), 1 <= p <= 83886080	binary
bit(*precision*), 1 <= p <= 83886080	binary
boolean	boolean
box	binary
character varying(*size*), 1 <= size <= 10485760	string
character(*size*), 1 <= size <= 10485760	string
cidr	string
circle	binary
date	string
daterange	string
double precision	double
inet	string
int4range	string
int8range	string
integer	int
json	string
jsonb	string

PostgreSQL **ソースデータ型**	Databricks Delta Lake **ターゲットデータ型**
line	binary
lseg	binary
macaddr	string
macaddr8	string
money	decimal(19,2)
numeric	string
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 1000, 39 <= s <= 1000	string
numrange	string
path	binary
point	binary
polygon	binary
real	float
smallint	int
time(*precision*) with time zone, 0 <= p <= 6	string
time(*precision*) without time zone, 0 <= p <= 6	string
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*) without time zone, 0 <= p <= 6	timestamp
tsrange	string
tstzrange	string
uuid	string
xml	string

LOB の制限事項

タスクウィザードの**［ソース］**ページにある**［詳細］**の下の**［LOB を含める］**オプションを指定した場合、いずれかのロードタイプを使用するデータベース取り込み初期ロードジョブと増分ロードジョブは、PostgreSQL BYTEA、TEXT、XML、およびその他のラージオブジェクトカラムから Databricks Delta ターゲットにデータをレプリケートできます。LOB カラムデータは、LOB タイプによって異なるバイト制限よりもサイズが大きい場合、ターゲットに書き込まれる前に切り詰められます。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

初期ロードジョブの場合、一括取り込みデータベースでは次の PostgreSQL データ型はサポートされていません。

- ABSTIME
- 配列型
- NAME
- 精度よりスケールが大きい NUMERIC
- オブジェクト識別子型
- PG_LSN
- RELTIME
- テキスト検索型:
  - TSQUERY
  - TSVECTOR
- ユーザー定義型

増分ロードジョブの場合、一括取り込みデータベースでは、初期ロードジョブでサポートされていないものに加えて、次の PostgreSQL データ型はサポートされていません。

- 空間タイプ
  - Box
  - Circle
  - Line
  - LSeg
  - Path
  - Point
  - Polygon
- 無制限のさまざまなタイプ

データベース統合ジョブは、これらのデータ型を持つカラムにはデプロイしたり null をプロパゲートしたりすることはできません。

PostgreSQL ソースと Google BigQuery ターゲット

次の表は、PostgreSQL ソースと Google BigQuery ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

PostgreSQL **ソースデータ型**	Google BigQuery **ターゲットデータ型**
bigint	int64
bit varying(*precision*), 1 <= p <= 83886080	bytes
bit(*precision*), 1 <= p <= 83886080	bytes
boolean	bool
box	bytes

PostgreSQL **ソースデータ型**	Google BigQuery **ターゲットデータ型**
character varying(*size*), 1 <= size <= 10485760	string
character(*size*), 1 <= size <= 10485760	string
cidr	string
circle	bytes
date	date
daterange	string
double precision	float64
inet	string
int4range	string
int8range	string
integer	int64
json	string
jsonb	string
line	bytes
lseg	bytes
macaddr	string
macaddr8	string
money	numeric
numeric	string
numeric(1,1)	numeric
numeric(10,10)	bignumeric
numeric(11,s), 10 <= s <= 11	bignumeric
numeric(12,s), 10 <= s <= 12	bignumeric
numeric(13,s), 10 <= s <= 13	bignumeric
numeric(14,s), 10 <= s <= 14	bignumeric
numeric(15,s), 10 <= s <= 15	bignumeric
numeric(16,s), 10 <= s <= 16	bignumeric

PostgreSQL **ソースデータ型**	Google BigQuery **ターゲットデータ型**
numeric(17,s), 10 <= s <= 17	bignumeric
numeric(18,s), 10 <= s <= 18	bignumeric
numeric(19,s), 10 <= s <= 19	bignumeric
numeric(2,s), 1 <= s <= 2	numeric
numeric(20,s), 10 <= s <= 20	bignumeric
numeric(21,s), 10 <= s <= 21	bignumeric
numeric(22,s), 10 <= s <= 22	bignumeric
numeric(23,s), 10 <= s <= 23	bignumeric
numeric(24,s), 10 <= s <= 24	bignumeric
numeric(25,s), 10 <= s <= 25	bignumeric
numeric(26,s), 10 <= s <= 26	bignumeric
numeric(27,s), 10 <= s <= 27	bignumeric
numeric(28,s), 10 <= s <= 28	bignumeric
numeric(3,s), 1 <= s <= 3	numeric
numeric(38,s), 10 <= s <= 38	bignumeric
numeric(4,s), 1 <= s <= 4	numeric
numeric(5,s), 1 <= s <= 5	numeric
numeric(6,s), 1 <= s <= 6	numeric
numeric(7,s), 1 <= s <= 7	numeric
numeric(8,s), 1 <= s <= 8	numeric
numeric(*precision*), 1 <= p <= 18	int64
numeric(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
numeric(p,s), 29 <= p <= 30, 10 <= s <= 29	bignumeric
numeric(p,s), 30 <= p <= 31, 1 <= s <= 30	bignumeric
numeric(p,s), 31 <= p <= 32, 1 <= s <= 31	bignumeric
numeric(p,s), 32 <= p <= 33, 1 <= s <= 32	bignumeric
numeric(p,s), 33 <= p <= 34, 1 <= s <= 33	bignumeric

PostgreSQL **ソースデータ型**	Google BigQuery **ターゲットデータ型**
numeric(p,s), 34 <= p <= 35, 1 <= s <= 34	bignumeric
numeric(p,s), 35 <= p <= 36, 1 <= s <= 35	bignumeric
numeric(p,s), 36 <= p <= 37, 1 <= s <= 36	bignumeric
numeric(p,s), 37 <= p <= 38, 1 <= s <= 37	bignumeric
numeric(p,s), 39 <= p <= 1000, 39 <= s <= 1000	string
numeric(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
numrange	string
path	bytes
point	bytes
polygon	bytes
real	float64
smallint	int64
time(*precision*) with time zone, 0 <= p <= 6	string
time(*precision*) without time zone, 0 <= p <= 6	time
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*) without time zone, 0 <= p <= 6	datetime
tsrange	string
tstzrange	string
uuid	string
xml	string

LOB の制限事項

タスクウィザードの［**ソース**］ページにある［**詳細**］の下の［**LOB を含める**］オプションを指定した場合、データベース取り込み初期ロードジョブと増分ロードジョブは、PostgreSQL BYTEA、TEXT、XML、およびその他のラージオブジェクトカラムから Google BigQuery ターゲットにデータをレプリケートできます。LOB カラムデータは、LOB タイプによって異なるバイト制限よりもサイズが大きい場合、ターゲットに書き込まれる前に切り詰められます。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

初期ロードジョブの場合、一括取り込みデータベースでは次の PostgreSQL データ型はサポートされていません。

- ABSTIME

- 配列型
- NAME
- オブジェクト識別子型
- PG_LSN
- RELTIME
- テキスト検索型:
  - TSQUERY
  - TSVECTOR
- ユーザー定義型

増分ロードジョブの場合、一括取り込みデータベースでは、初期ロードジョブでサポートされていないものに加えて、次の PostgreSQL データ型はサポートされていません。

- 空間タイプ
  - Box
  - Circle
  - Line
  - LSeg
  - Path
  - Point
  - Polygon
- 無制限のさまざまなタイプ

データベース統合ジョブは、これらのデータ型を持つカラムにはデプロイしたり null をプロパゲートしたりすることはできません。

PostgreSQL ソースと Microsoft Azure Synapse Analytics ターゲット

次の表は、PostgreSQL ソースと Microsoft Azure Synapse Analytics ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

PostgreSQL **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
bigint	bigint
bit varying(1)	varbinary(*size*), 1 <= size <= max
bit varying(*precision*), 2 <= p <= 83886080	varbinary(*size*), 1 <= size <= max
bit(1)	bit
bit(*precision*), 2 <= p <= 64000	binary(*size*), 1 <= size <= 8000
bit(*precision*), 64001 <= p <= 83886080	varbinary(max)
boolean	bit

PostgreSQL **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
box	varbinary(115)
character varying(*size*), 1 <= size <= 10485760	varchar(*size*), 4 <= size <= max
character(*size*), 1 <= size <= 10485760	varchar(*size*), 4 <= size <= max
cidr	varchar(45)
circle	varbinary(87)
date	date
daterange	varchar(29)
double precision	float
inet	varchar(45)
int4range	varchar(25)
int8range	varchar(43)
integer	int
json	varchar(max)
jsonb	varchar(max)
line	varbinary(85)
lseg	varbinary(117)
macaddr	varchar(17)
macaddr8	varchar(23)
money	decimal(19,2)
numeric	number
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 1000, 39 <= s <= 1000	varchar(*size*), 42 <= s <= 1003
numrange	varchar(max)
path	varbinary(max)
point	varbinary(57)
polygon	varbinary(max)
real	real

PostgreSQL **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
smallint	smallint
time(*precision*) with time zone, 0 <= p <= 6	datetimeoffset(*precision*), 0 <= p <= 6
time(*precision*) without time zone, 0 <= p <= 6	time(*precision*), 0 <= p <= 6
timestamp(*precision*) with time zone, 0 <= p <= 6	datetimeoffset(*precision*), 0 <= p <= 6
timestamp(*precision*) without time zone, 0 <= p <= 6	datetime2(*precision*), 0 <= p <= 6
tsrange	varchar(63)
tstzrange	varchar(75)
uuid	uniqueidentifier
xml	varchar(max)

LOB の制限事項

タスクウィザードの［**ソース**］ページにある［**詳細**］の下の［**LOB を含める**］オプションを指定した場合、データベース取り込み初期ロードジョブと増分ロードジョブは、PostgreSQL BYTEA、TEXT、XML、およびその他のラージオブジェクトカラムから Microsoft Azure Synapse Analytics ターゲットにデータをレプリケートできます。LOB カラムデータは、LOB タイプによって異なるバイト制限よりもサイズが大きい場合、ターゲットに書き込まれる前に切り詰められます。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

初期ロードジョブの場合、一括取り込みデータベースでは次の PostgreSQL データ型はサポートされていません。

- ABSTIME
- 配列型
- NAME
- オブジェクト識別子型
- PG_LSN
- RELTIME
- テキスト検索型:

  - TSQUERY

  - TSVECTOR
- ユーザー定義型

増分ロードジョブの場合、一括取り込みデータベースでは、初期ロードジョブでサポートされていないものに加えて、次の PostgreSQL データ型はサポートされていません。

- 空間タイプ

  - Box

- Circle
- Line
- LSeg
- Path
- Point
- Polygon

- 無制限のさまざまなタイプ

データベース統合ジョブは、これらのデータ型を持つカラムにはデプロイしたり null をプロパゲートしたりすることはできません。

PostgreSQL ソースと Oracle ターゲット

次の表に、PostgreSQL ソースと Oracle ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示します。

PostgreSQL **ソースデータ型**	Oracle **ターゲットデータ型**
bigint	number(19)
bit varying(*precision*), 1 <= p <= 83886080	blob
bit(*precision*), 1 <= p <= 83886080	blob
boolean	char(1 char)
box	blob
character varying(*size*), 1 <= s <= 1501	char(s char), 4 <= s <= 2000
character varying(*size*), 2001 <= s <= 10485760	clob
character(*size*), 1 <= s <= 2000	char(s char), 4 <= s <= 2000
character(*size*), 2100 <= s <= 10485760	clob
cidr	char(45 char)
circle	blob
date	date
daterange	char(29 char)
double precision	binary_double
inet	char(45 char)
int4range	char(25 char)
int8range	char(43 char)
integer	number(10)

PostgreSQL ソースデータ型	Oracle ターゲットデータ型
json	clob
jsonb	clob
line	blob
lseg	blob
macaddr	char(17 char)
macaddr8	char(23 char)
money	number(19,2)
numeric	clob
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	number(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 1000, 39 <= s <= 1000	char(s char), 42 <= s <= 1003
numrange	clob
path	blob
point	blob
polygon	blob
real	binary_double
smallint	number(5)
time(*precision*) with time zone, 0 <= p <= 6	char(s char), 48 <= s <= 55
time(*precision*) without time zone, 0 <= p <= 6	timestamp(*precision*), 0 <= p <= 6
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp(*precision*) with time zone, 0 <= p <= 6
timestamp(*precision*) without time zone, 0 <= p <= 0	date
timestamp(*precision*) without time zone, 1 <= p <= 6	timestamp(*precision*), 1 <= p <= 6
tsrange	char(63 char)
tstzrange	char(75 char)
uuid	char(36 char)
xml	clob

LOB の制限事項

タスクウィザードの［**ソース**］ページにある［**詳細**］の下の［**LOB を含める**］オプションを指定した場合、データベース取り込み初期ロードジョブは、PostgreSQL BYTEA、TEXT、XML、およびその他のラージオブジェ

クトカラムから Oracle ターゲットにデータをレプリケートできます。LOB カラムデータは、LOB タイプによって異なるバイト制限よりもサイズが大きい場合、ターゲットに書き込まれる前に切り詰められます。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

初期ロードジョブの場合、一括取り込みデータベースでは次の PostgreSQL データ型はサポートされていません。

- ABSTIME
- 配列型
- NAME
- オブジェクト識別子型
- PG_LSN
- RELTIME
- テキスト検索型:
  - TSQUERY
  - TSVECTOR
- ユーザー定義型

増分ロードジョブ、初期および増分ロードジョブの場合、一括取り込みデータベースでは、初期ロードジョブでサポートされていないものに加えて、次の PostgreSQL データ型はサポートされていません。

- 空間タイプ
  - Box
  - Circle
  - Line
  - LSeg
  - Path
  - Point
  - Polygon
- 無制限のさまざまなタイプ

データベース統合ジョブは、これらのデータ型を持つカラムにはデプロイしたり null をプロパゲートしたりすることはできません。

PostgreSQL ソースと Snowflake ターゲット

次の表は、PostgreSQL ソースと Snowflake ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

PostgreSQL **ソースデータ型**	Snowflake **ターゲットデータ型**
bigint	number
bit varying(*precision*), 1 <= p <= 83886080	binary(*size*), 1 <= size <= 6291457
bit(*precision*), 1 <= p <= 83886080	binary(*size*), 1 <= size <= 6291457

PostgreSQL **ソースデータ型**	Snowflake **ターゲットデータ型**
boolean	boolean
box	binary(115)
character varying(*size*), 1 <= size <= 10485760	varchar(*size*), 4 <= size <= 16776004
character(*size*), 1 <= size <= 10485760	varchar(*size*), 4 <= size <= 16776004
cidr	varchar(45)
circle	binary(87)
date	date
daterange	varchar(29)
double precision	float
inet	varchar(45)
int4range	varchar(25)
int8range	varchar(43)
integer	integer
json	variant
jsonb	variant
line	binary(85)
lseg	binary(117)
macaddr	varchar(17)
macaddr8	varchar(23)
money	number(19,2)
numeric	varchar（131074）
numeric(p,s), 1 <= p <= 38, 0 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
numeric(p,s), 38 <= p <= 1000, 38 <= s <= 1000	varchar(*size*), 41 <= size <= 1003
numrange	varchar(294917)
path	binary
point	binary(57)
polygon	binary

PostgreSQL **ソースデータ型**	Snowflake **ターゲットデータ型**
real	float
smallint	number
time(*precision*) with time zone, 0 <= p <= 6	timestamp_tz(*precision*), 0 <= p <= 6
time(*precision*) without time zone, 0 <= p <= 6	time(*precision*), 0 <= p <= 6
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp_tz(*precision*), 0 <= p <= 6
timestamp(*precision*) without time zone, 0 <= p <= 6	timestamp_ntz(*precision*), 0 <= p <= 6
tsrange	varchar(63)
tstzrange	varchar(75)
uuid	varchar
xml	variant

LOB の制限事項

タスクウィザードの［**ソース**］ページにある［**詳細**］の下の［**LOB を含める**］オプションを指定した場合、いずれかのロードタイプを使用するデータベース取り込みジョブは、PostgreSQL BYTEA、TEXT、XML、およびその他のラージオブジェクトカラムから Snowflake ターゲットにデータをレプリケートできます。LOB カラムデータは、LOB タイプによって異なるバイト制限よりもサイズが大きい場合、ターゲットに書き込まれる前に切り詰められます。詳細については、「ソースの設定」 (ページ 102)の［LOB を含める］に関する説明を参照してください。

サポートされていないソースデータ型

初期ロードジョブの場合、一括取り込みデータベースでは次の PostgreSQL データ型はサポートされていません。

- ABSTIME
- 配列型
- NAME
- オブジェクト識別子型
- PG_LSN
- RELTIME
- テキスト検索型:

  - TSQUERY

  - TSVECTOR
- ユーザー定義型

増分ロードジョブ、初期および増分ロードジョブの場合、一括取り込みデータベースでは、初期ロードジョブでサポートされていないものに加えて、次の PostgreSQL データ型はサポートされていません。

- 空間タイプ

  - Box

- Circle
- Line
- LSeg
- Path
- Point
- Polygon

- 無制限のさまざまなタイプ

データベース統合ジョブは、これらのデータ型を持つカラムにはデプロイしたり null をプロパゲートしたりすることはできません。

SAP HANA ソースと Amazon Redshift ターゲット

次の表は、SAP HANA または SAP HANA Cloud ソースと Amazon Redshift ターゲットを使用した一括取り込みデータベース設定のデフォルトのデータ型マッピングを示しています。

SAP HANA **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
alphanum(*precision*), 1 <= p <= 127	character(*size*), 1 <= size <= 127
array	binary varying(1024000)
bigint	bigint
binary(*size*), 1 <= size <= 2000	binary varying(*size*), 1 <= size <= 2000
boolean	boolean
char(*size*), 1 <= size <= 2000	character(*size*), 1 <= size <= 2000
date	date
decimal	character varying(255)
decimal(38,38)	character varying(41)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
double	double precision
float	double precision
integer	integer
nchar(*size*), 1 <= size <= 2000	character varying(*size*), 4 <= size <= 8000
nvarchar(*size*), 1 <= size <= 5000	character varying(*size*), 4 <= size <= 20000
real	real
seconddate	timestamp without time zone
shorttext(*precision*), 1 <= p <= 5000	character varying(*size*), 1 <= size <= 5000

SAP HANA **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
smalldecimal	character varying(255)
smallint	smallint
st_geometry	binary varying(1024000)
st_point	binary varying(1024000)
time	time without time zone
timestamp	character varying(27)
timestamp	timestamp without time zone
tinyint	smallint
varbinary(*size*), 1 <= size <= 5000	binary varying(*size*), 1 <= size <= 5000
varchar(*size*), 1 <= size <= 5000	character varying(*size*), 1 <= size <= 5000

サポートされていないソースデータ型:

- 英数字（SAP HANA Cloud のみ）
- array
- bintext
- blob
- clob
- char（SAP HANA Cloud のみ）
- nclob
- st_geometry
- st_point
- text

一括取り込みデータベースは、これらのサポートされていないデータ型を持つカラムには null のみをレプリケートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

SAP HANA ソースと Databricks Delta ターゲット

次の表は、SAP HANA または SAP HANA Cloud ソースと Databricks Delta ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

SAP HANA **ソースデータ型**	Databricks Delta **ターゲットデータ型**
alphanum(*precision*), 1 <= p <= 127	string
array	binary
bigint	long
binary(*size*), 1 <= size <= 2000	binary
boolean	boolean
char(*size*), 1 <= s <= 2000	string
date	string
decimal	string
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
double	double
float	double
integer	integer
nchar(*size*), 1 <= size <= 2000	string
nvarchar(*size*), 1 <= size <= 5000	string
real	float
seconddate	timestamp
shorttext(*precision*), 1 <= p <= 5000	string
smalldecimal	string
smallint	integer
st_geometry	binary
st_point	binary
time	string
timestamp	string
timestamp	timestamp
tinyint	integer

SAP HANA **ソースデータ型**	Databricks Delta **ターゲットデータ型**
varbinary(*size*), 1 <= size <= 5000	binary
varchar(*size*), 1 <= size <= 5000	string

サポートされていないソースデータ型:

- 英数字（SAP HANA Cloud のみ）
- array
- bintext
- blob
- clob
- char（SAP HANA Cloud のみ）
- 精度よりスケールが大きい decimal
- nclob
- st_geometry
- st_point
- text

一括取り込みデータベースは、これらのサポートされていないデータ型を持つカラムには null のみをレプリケートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

SAP HANA ソースと Google BigQuery ターゲット

次の表は、SAP HANA または SAP HANA Cloud ソースと Google BigQuery ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

SAP HANA **ソースデータ型**	Google BigQuery **ターゲットデータ型**
alphanum(*precision*), 1 <= p <= 127	string
array	bytes
bigint	int64
binary(*size*), 1 <= size <= 2000	bytes
boolean	bool
char(*size*), 1 <= size <= 2000	string
date	date
decimal	string

SAP HANA **ソースデータ型**	Google BigQuery **ターゲットデータ型**
decimal(1,1)	numeric
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(2,s), 1 <= s <= 2	numeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(3,s), 1 <= s <= 3	numeric
decimal(38,s), 10 <= s <= 38	bignumeric
decimal(4,s), 1 <= s <= 4	numeric
decimal(5,s), 1 <= s <= 5	numeric
decimal(6,s), 1 <= s <= 6	numeric
decimal(7,s), 1 <= s <= 7	numeric

SAP HANA **ソースデータ型**	Google BigQuery **ターゲットデータ型**
decimal(8,s), 1 <= s <= 8	numeric
decimal(p,0), 1 <= p <= 18	int64
decimal(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
decimal(p,s), 29 <= p <= 30, 0 <= s <= 29	bignumeric
decimal(p,s), 30 <= p <= 31, 0 <= s <= 30	bignumeric
decimal(p,s), 31 <= p <= 32, 0 <= s <= 31	bignumeric
decimal(p,s), 32 <= p <= 33, 0 <= s <= 32	bignumeric
decimal(p,s), 33 <= p <= 34, 0 <= s <= 33	bignumeric
decimal(p,s), 34 <= p <= 35, 0 <= s <= 34	bignumeric
decimal(p,s), 35 <= p <= 36, 0 <= s <= 35	bignumeric
decimal(p,s), 36 <= p <= 37, 0 <= s <= 36	bignumeric
decimal(p,s), 37 <= p <= 38, 0 <= s <= 37	bignumeric
decimal(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
double	float64
float	float64
integer	int64
nchar(*size*), 1 <= size <= 2000	string
nvarchar(*size*), 1 <= size <= 5000	string
real	float64
seconddate	datetime
shorttext(*precision*), 1 <= p <= 5000	string
smalldecimal	string
smallint	int64
st_geometry	bytes
st_point	bytes
time	time
timestamp	datetime

SAP HANA **ソースデータ型**	Google BigQuery **ターゲットデータ型**
timestamp	string
tinyint	int64
varbinary(*size*), 1 <= size <= 5000	bytes
varchar(*size*), 1 <= size <= 5000	string

サポートされていないソースデータ型:

- 英数字（SAP HANA Cloud のみ）
- array
- bintext
- blob
- clob
- char（SAP HANA Cloud のみ）
- nclob
- st_geometry
- st_point
- text

一括取り込みデータベースは、これらのサポートされていないデータ型を持つカラムには null のみをレプリケートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

SAP HANA ソースと Microsoft Azure Synapse Analytics ターゲット

次の表は、SAP HANA または SAP HANA Cloud ソースと Microsoft Azure Synapse Analytics ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

SAP HANA **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
alphanum(*precision*), 1 <= p <= 127	char(*size*), 1 <= size <= 127
array	varbinary(max)
bigint	bigint
binary(*size*), 1 <= size <= 2000	binary(*size*), 1 <= size <= 2000
boolean	bit
char(*size*), 1 <= size <= 2000	char(*size*), 1 <= size <= 2000

SAP HANA ソースデータ型	Microsoft Azure Synapse Analytics ターゲットデータ型
date	date
decimal	varchar（255）
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
double	float
float	float
integer	int
nchar(*size*), 1 <= size <= 2000	char(*size*), 4 <= size <= 8000
nvarchar(*size*), 1 <= size <= 5000	varchar(*size*), 4 <= size <= max
real	real
seconddate	datetime2(0)
shorttext(*precision*), 1 <= p <= 5000	varchar(*size*), 1 <= s <= 5000
smalldecimal	varchar（255）
smallint	smallint
st_geometry	varbinary(max)
st_point	varbinary(max)
time	time(0)
timestamp	datetime2(*precision*), 0 <= p <= 7
tinyint	tinyint
varbinary(*size*), 1 <= size <= 5000	varbinary(*size*), 1 <= size <= 5000
varchar(*size*), 1 <= size <= 5000	varchar(*size*), 1 <= size <= 5000

サポートされていないソースデータ型:

- 英数字（SAP HANA Cloud のみ）
- array
- bintext
- blob
- clob
- char（SAP HANA Cloud のみ）
- nclob
- st_geometry

- st_point
- text

一括取り込みデータベースは、これらのサポートされていないデータ型を持つカラムには null のみをレプリケートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

SAP HANA ソースと Microsoft SQL Server ターゲット

次の表に、SAP HANA または SAP HANA Cloud ソースと Microsoft SQL Server ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示します。

SAP HANA **ソースデータ型**	Microsoft SQL Server **ターゲットデータ型**
alphanum(*precision*), 1 <= p <= 127	varchar(*size*), 1 <= size <= 127
array	varbinary(max)
bigint	bigint
binary(*size*), 1 <= size <= 2000	varbinary(*size*), 1 <= size <= 2000
boolean	bit
char(*size*), 1 <= size <= 2000	varchar(*size*), 1 <= size <= 2000
date	date
decimal	char(255)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
double	float
float	float
integer	int
nchar(*size*), 1 <= size <= 2000	nvarchar(*size*), 4 <= size <= 8000
nvarchar(*size*), 1 <= size <= 5000	nvarchar(*size*), 4 <= size <= max
real	real
real	real
seconddate	datetime2(0)
shorttext(*precision*), 1 <= p <= 5000	varchar(*size*), 1 <= size <= 5000
smalldecimal	char(255)
smallint	smallint

SAP HANA ソースデータ型	Microsoft SQL Server ターゲットデータ型
st_geometry	varbinary(max)
st_point	varbinary(max)
time	time(0)
timestamp	datetime2(*precision*), 0 <= p <= 7
tinyint	tinyint
varbinary(*size*), 1 <= size <= 5000	varbinary(*size*), 1 <= size <= 5000
varchar(*size*), 1 <= size <= 5000	varchar(*size*), 1 <= size <= 5000

サポートされていないソースデータ型:

- 英数字（SAP HANA Cloud のみ）
- array
- bintext
- blob
- clob
- char（SAP HANA Cloud のみ）
- nclob
- st_geometry
- st_point
- text

一括取り込みデータベースは、これらのサポートされていないデータ型を持つカラムには null のみをレプリケートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

SAP HANA ソースと Oracle ターゲット

次の表に、SAP HANA または SAP HANA Cloud ソースと Oracl ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示します。

SAP HANA ソースデータ型	Oracle ターゲットデータ型
alphanum(*precision*), 1 <= p <= 127	varchar2(s char), 1 <= s <= 127
array	blob
bigint	number(19)
binary(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000

SAP HANA ソースデータ型	Oracle ターゲットデータ型
boolean	char(1 char)
char(*size*), 1 <= size <= 2000	varchar2(s byte), 1 <= s <= 2000
date	date
decimal	char(255 char)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	number(p,s), 1 <= p <= 38, 0 <= s <= 38
double	binary_double
float	binary_double
integer	number(10)
nchar(*size*), 1 <= size <= 1000	nvarchar2(s char), 1 <= s <= 1000
nchar(*size*), 1001 <= size <= 2000	char(2000 char)
nvarchar(*size*), 1 <= size <= 901	nvarchar2(s char), 1 <= s <= 901
nvarchar(*size*), 1001 <= size <= 1901	char(2000 char)
nvarchar(*size*), 2001 <= size <= 5000	clob
real	binary_float
real	binary_float
seconddate	date
shorttext(*precision*), 1 <= p <= 3901	varchar2(s char), 1 <= s <= 3901
shorttext(*precision*), 4001 <= p <= 5000	clob
smalldecimal	char(255 char)
smallint	number(5)
st_geometry	blob
st_point	blob
time	timestamp(0)
timestamp	date
timestamp	timestamp(*precision*), 1 <= p <= 7
tinyint	number(3)
varbinary(*size*), 1 <= size <= 1501	raw(*size*), 1 <= size <= 1501

SAP HANA **ソースデータ型**	Oracle **ターゲットデータ型**
varchar(*size*), 1 <= size <= 3901	varchar2(s byte), 1 <= s <= 3901
varchar(*size*), 4001 <= size <= 5000	clob

サポートされていないソースデータ型:

- 英数字（SAP HANA Cloud のみ）
- array
- bintext
- blob
- clob
- char（SAP HANA Cloud のみ）
- nclob
- st_geometry
- st_point
- text

一括取り込みデータベースは、これらのサポートされていないデータ型を持つカラムには null のみをレプリケートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

SAP HANA ソースと Snowflake ターゲット

次の表は、SAP HANA または SAP HANA Cloud ソースと Snowflake ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

SAP HANA **ソースデータ型**	Snowflake **ターゲットデータ型**
alphanum(*precision*), 1 <= p <= 127	char(*size*), 1 <= size <= 127
array	binary
bigint	integer
binary(*size*), 1 <= size <= 2000	binary(*size*), 1 <= size <= 2000
boolean	boolean
char(*size*), 1 <= size <= 2000	char(*size*), 1 <= size <= 2000
date	date
decimal	char(255)
decimal(38,38)	char(41)

SAP HANA ソースデータ型	Snowflake ターゲットデータ型
decimal(p,s), 1 <= p <= 38, 0 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
double	float
float	float
integer	integer
nchar(*size*), 1 <= size <= 2000	char(*size*), 4 <= size <= 8000
nvarchar(*size*), 1 <= size <= 5000	varchar(*size*), 4 <= size <= 20000
real	float
seconddate	timestamp_ntz(0)
shorttext(*precision*), 1 <= p <= 5000	varchar(*size*), 1 <= size <= 5000
smalldecimal	char(255)
smallint	integer
st_geometry	binary
st_point	binary
time	time(0)
TIMESTAMP	timestamp_ntz(*precision*), 0 <= p <= 7
tinyint	integer
varbinary(*size*), 1 <= size <= 5000	binary(*size*), 1 <= size <= 5000
varchar(*size*), 1 <= size <= 5000	varchar(*size*), 1 <= size <= 5000

サポートされていないソースデータ型:

- 英数字（SAP HANA Cloud のみ）
- array
- bintext
- blob
- clob
- char（SAP HANA Cloud のみ）
- nclob
- st_geometry
- st_point
- text

一括取り込みデータベースは、これらのサポートされていないデータ型を持つカラムには null のみをレプリケートします。

注: サポートされていないデータ型がデフォルトのマッピングに表示される場合があります。ただし、これらのマッピングでは null がレプリケートされます。

Teradata ソースと Amazon Redshift ターゲット

次の表は、Teradata ソースと Amazon Redshift ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Teradata **ソースデータ型**	Amazon Redshift **ターゲットデータ型**
array	binary varying(1024000)
bigint	bigint
blob	binary varying(1024000)
byte(*precision*), 1 <= p <= 64000	binary varying(*size*), 1 <= size <= 64000
byteint	smallint
char(*size*), 1 <= size <= 64000	character varying(*size*), 4 <= size <= 65535
clob	character varying(65535)
date	date
decimal(p,s), 1 <= p <= 38, 1 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
float	double precision
integer	integer
interval day(*precision*) to hour, 1 <= p <= 4	character varying(*size*), 5 <= size <= 8
interval day(*precision*) to minute, 1 <= p <= 4	character varying(*size*), 8 <= size <= 11
interval day(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	character varying(*size*), 12 <= size <= 21
interval day(*precision*), 1 <= p <= 4	character varying(*size*), 2 <= size <= 5
interval hour(*precision*) to minute, 1 <= p <= 4	character varying(*size*), 5 <= size <= 8
interval hour(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	character varying(*size*), 9 <= size <= 18
interval hour(*precision*), 1 <= p <= 4	character varying(*size*), 2 <= size <= 5
interval minute(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	character varying(*size*), 9 <= size <= 18
interval minute(*precision*), 1 <= p <= 4	character varying(*size*), 2 <= size <= 5
interval month(*precision*), 1 <= p <= 4	character varying(*size*), 2 <= size <= 5
interval second(p,s), 1 <= p <= 4, 0 <= s <= 6	character varying(*size*), 3 <= size <= 12

Teradata ソースデータ型	Amazon Redshift ターゲットデータ型
interval year(*precision*) to month, 1 <= p <= 4	character varying(*size*), 5 <= size <= 8
interval year(*precision*), 1 <= p <= 4	character varying(*size*), 2 <= size <= 5
json	character varying(65535)
mbr	binary varying(256)
number(*,s), 0 <= s <= 37	character varying(255)
number(p,s), 1 <= p <= 38, 1 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
period(date)	character varying(28)
period(time(*precision*) with time zone), 0 <= p <= 6	character varying(*size*), 36 <= size <= 50
period(time(*precision*)), 0 <= p <= 6	character varying(*size*), 24 <= size <= 38
period(timestamp(*precision*) with time zone), 0 <= p <= 6	character varying(*size*), 58 <= size <= 72
period(timestamp(*precision*)), 0 <= p <= 6	character varying(*size*), 46 <= size <= 60
smallint	smallint
time(*precision*) with time zone, 0 <= p <= 6	timetz
time(*precision*), 0 <= p <= 6	time without time zone
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp with time zone
timestamp(*precision*), 0 <= p <= 6	timestamp without time zone
varbyte(*precision*), 1 <= p <= 64000	binary varying(*size*), 1 <= size <= 64000
varchar(*size*), 1 <= size <= 64000	character varying(*size*), 4 <= size <= 65535
varray	binary varying(1024000)
xml	character varying(65535)

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Teradata データ型はサポートされません。

- ARRAY
- Blob
- CLOB
- JSON
- ST_GEOMETRY
- XML

注: サポートされていないデータ型のほとんどがデフォルトのマッピングに表示されます。ただし、これらのマッピングでは null がレプリケートされます。

Teradata ソースと Databricks Delta ターゲット

次の表は、Teradata ソースと Databricks Delta ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Teradata **ソースデータ型**	Databricks Delta **ターゲットデータ型**
array	binary
bigint	long
blob	binary
byte(*precision*), 1 <= p <= 64000	binary
byteint	integer
char(*size*), 1 <= size <= 64000	string
clob	string
date	string
decimal(p,s), 1 <= p <= 38, 1 <= s <= 37	decimal(p,s), 1 <= p <= 38, 1 <= s <= 37
float	double
integer	integer
interval day(*precision*) to hour, 1 <= p <= 4	string
interval day(*precision*) to minute, 1 <= p <= 4	string
interval day(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	string
interval day(*precision*), 1 <= p <= 4	string
interval hour(*precision*) to minute, 1 <= p <= 4	string
interval hour(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	string
interval hour(*precision*), 1 <= p <= 4	string
interval minute(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	string
interval minute(*precision*), 1 <= p <= 4	string
interval month(*precision*), 1 <= p <= 4	string
interval second(p,s), 1 <= p <= 4, 0 <= s <= 6	string
interval year(*precision*) to month, 1 <= p <= 4	string

Teradata ソースデータ型	Databricks Delta ターゲットデータ型
interval year(*precision*), 1 <= p <= 4	string
json	string
mbr	binary
number(*,s), 0 <= s <= 37	string
number(p,s), 1 <= p <= 38, 1 <= s <= 37	decimal(p,s), 1 <= p <= 38, 1 <= s <= 37
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	decimal(p,s), 1 <= p <= 38, 1 <= s <= 37
period(date)	string
period(time(*precision*) with time zone), 0 <= p <= 6	string
period(time(*precision*)), 0 <= p <= 6	string
period(timestamp(*precision*) with time zone), 0 <= p <= 6	string
period(timestamp(*precision*)), 0 <= p <= 6	string
smallint	integer
time(*precision*) with time zone, 0 <= p <= 6	string
time(*precision*), 0 <= p <= 6	string
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*), 0 <= p <= 6	timestamp
varbyte(*precision*), 1 <= p <= 64000	binary
varchar(*size*), 1 <= size <= 64000	string
varray	binary
xml	string

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Teradata データ型はサポートされません。

- ARRAY
- Blob
- CLOB
- 精度よりスケールが大きい DECIMAL、NUMBER、または NUMERIC
- JSON
- ST_GEOMETRY
- XML

注: サポートされていないデータ型のほとんどがデフォルトのマッピングに表示されます。ただし、これらのマッピングでは null がレプリケートされます。

Teradata ソースと Google BigQuery ターゲット

次の表は、Teradata ソースと Google BigQuery ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Teradata **ソースデータ型**	Google BigQuery **ターゲットデータ型**
array	bytes
bigint	int64
blob	bytes
byte(*precision*), 1 <= p <= 64000	bytes
byteint	int64
char(*size*), 1 <= size <= 64000	string
clob	string
date	date
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric

Teradata ソースデータ型	Google BigQuery ターゲットデータ型
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(29,s), 10 <= s <= 29	bignumeric
decimal(*precision*), 1 <= p <= 18	int64
decimal(*precision*), 19 <= p <= 29	numeric
decimal(*precision*), 30 <= p <= 38	bignumeric
decimal(p,s), 1 <= p <= 38, 1 <= s <= 9	numeric
decimal(p,s), 30 <= p <= 31, 1 <= s <= 30	bignumeric
decimal(p,s), 31 <= p <= 32, 1 <= s <= 31	bignumeric
decimal(p,s), 32 <= p <= 33, 1 <= s <= 32	bignumeric
decimal(p,s), 33 <= p <= 34, 1 <= s <= 33	bignumeric
decimal(p,s), 34 <= p <= 35, 1 <= s <= 34	bignumeric
decimal(p,s), 35 <= p <= 36, 1 <= s <= 35	bignumeric
decimal(p,s), 36 <= p <= 37, 1 <= s <= 36	bignumeric
decimal(p,s), 37 <= p <= 38, 1 <= s <= 37	bignumeric
float	float64
integer	int64
interval day(*precision*) to hour, 1 <= p <= 4	string
interval day(*precision*) to minute, 1 <= p <= 4	string
interval day(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	string
interval day(*precision*), 1 <= p <= 4	string
interval hour(*precision*) to minute, 1 <= p <= 4	string
interval hour(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	string
interval hour(*precision*), 1 <= p <= 4	string
interval minute(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	string

Teradata **ソースデータ型**	Google BigQuery **ターゲットデータ型**
interval minute(*precision*), 1 <= p <= 4	string
interval month(*precision*), 1 <= p <= 4	string
interval second(p,s), 1 <= p <= 4, 0 <= s <= 6	string
interval year(*precision*) to month, 1 <= p <= 4	string
interval year(*precision*), 1 <= p <= 4	string
json	string
mbr	bytes
number(*,s), 0 <= s <= 37	string
number(10,10)	bignumeric
number(11,s), 10 <= s <= 11	bignumeric
number(12,s), 10 <= s <= 12	bignumeric
number(13,s), 10 <= s <= 13	bignumeric
number(14,s), 10 <= s <= 14	bignumeric
number(15,s), 10 <= s <= 15	bignumeric
number(16,s), 10 <= s <= 16	bignumeric
number(17,s), 10 <= s <= 17	bignumeric
number(18,s), 10 <= s <= 18	bignumeric
number(19,s), 10 <= s <= 19	bignumeric
number(20,s), 10 <= s <= 20	bignumeric
number(21,s), 10 <= s <= 21	bignumeric
number(22,s), 10 <= s <= 22	bignumeric
number(23,s), 10 <= s <= 23	bignumeric
number(24,s), 10 <= s <= 24	bignumeric
number(25,s), 10 <= s <= 25	bignumeric
number(26,s), 10 <= s <= 26	bignumeric
number(27,s), 10 <= s <= 27	bignumeric
number(28,s), 10 <= s <= 28	bignumeric

Teradata **ソースデータ型**	Google BigQuery **ターゲットデータ型**
number(29,s), 10 <= s <= 29	bignumeric
number(*precision*), 1 <= p <= 18	int64
number(*precision*), 19 <= p <= 29	numeric
number(*precision*), 30 <= p <= 36	bignumeric
number(p,s), 1 <= p <= 38, 1 <= s <= 9	numeric
number(p,s), 30 <= p <= 31, 1 <= s <= 30	bignumeric
number(p,s), 31 <= p <= 32, 1 <= s <= 31	bignumeric
number(p,s), 32 <= p <= 33, 1 <= s <= 32	bignumeric
number(p,s), 33 <= p <= 34, 1 <= s <= 33	bignumeric
number(p,s), 34 <= p <= 35, 1 <= s <= 34	bignumeric
number(p,s), 35 <= p <= 36, 1 <= s <= 35	bignumeric
number(p,s), 36 <= p <= 37, 1 <= s <= 36	bignumeric
number(p,s), 37 <= p <= 38, 1 <= s <= 37	bignumeric
numeric(10,10)	bignumeric
numeric(11,s), 10 <= s <= 11	bignumeric
numeric(12,s), 10 <= s <= 12	bignumeric
numeric(13,s), 10 <= s <= 13	bignumeric
numeric(14,s), 10 <= s <= 14	bignumeric
numeric(15,s), 10 <= s <= 15	bignumeric
numeric(16,s), 10 <= s <= 16	bignumeric
numeric(17,s), 10 <= s <= 17	bignumeric
numeric(18,s), 10 <= s <= 18	bignumeric
numeric(19,s), 10 <= s <= 19	bignumeric
numeric(20,s), 10 <= s <= 20	bignumeric
numeric(21,s), 10 <= s <= 21	bignumeric
numeric(22,s), 10 <= s <= 22	bignumeric
numeric(23,s), 10 <= s <= 23	bignumeric

Teradata ソースデータ型	Google BigQuery ターゲットデータ型
numeric(24,s), 10 <= s <= 24	bignumeric
numeric(25,s), 10 <= s <= 25	bignumeric
numeric(26,s), 10 <= s <= 26	bignumeric
numeric(27,s), 10 <= s <= 27	bignumeric
numeric(28,s), 10 <= s <= 28	bignumeric
numeric(29,s), 10 <= s <= 29	bignumeric
numeric(*precision*), 1 <= p <= 18	int64
numeric(*precision*), 19 <= p <= 29	numeric
numeric(*precision*), 30 <= p <= 36	bignumeric
numeric(p,s), 1 <= p <= 38, 1 <= s <= 9	numeric
numeric(p,s), 30 <= p <= 31, 1 <= s <= 30	bignumeric
numeric(p,s), 31 <= p <= 32, 1 <= s <= 31	bignumeric
numeric(p,s), 32 <= p <= 33, 1 <= s <= 32	bignumeric
numeric(p,s), 33 <= p <= 34, 1 <= s <= 33	bignumeric
numeric(p,s), 34 <= p <= 35, 1 <= s <= 34	bignumeric
numeric(p,s), 35 <= p <= 36, 1 <= s <= 35	bignumeric
numeric(p,s), 36 <= p <= 37, 1 <= s <= 36	bignumeric
numeric(p,s), 37 <= p <= 38, 1 <= s <= 37	bignumeric
period(date)	string
period(time(*precision*) with time zone), 0 <= p <= 6	string
period(time(*precision*)), 0 <= p <= 6	string
period(timestamp(*precision*) with time zone), 0 <= p <= 6	string
period(timestamp(*precision*)), 0 <= p <= 6	string
smallint	int64
time(*precision*) with time zone, 0 <= p <= 6	timestamp
time(*precision*), 0 <= p <= 6	time
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp

Teradata **ソースデータ型**	Google BigQuery **ターゲットデータ型**
timestamp(*precision*), 0 <= p <= 6	datetime
varbyte(*precision*), 1 <= p <= 64000	bytes
varchar(*size*), 1 <= size <= 64000	string
varray	bytes
xml	string

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Teradata データ型はサポートされません。

- ARRAY
- Blob
- CLOB
- JSON
- ST_GEOMETRY
- XML

注: サポートされていないデータ型のほとんどがデフォルトのマッピングに表示されます。ただし、これらのマッピングでは null がレプリケートされます。

Teradata ソースと Microsoft Azure Synapse Analytics ターゲット

次の表は、Teradata ソースと Microsoft Azure Synapse Analytics ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Teradata **ソースデータ型**	Microsoft Azure Synapse Analytics **ターゲットデータ型**
array	varbinary(max)
bigint	bigint
blob	varbinary(max)
byte(*precision*), 1 <= p <= 7501	binary(*size*), 1 <= size <= 7501
byte(*precision*), 8001 <= p <= 64000	varbinary(max)
byteint	smallint
char(*size*), 1 <= size <= 8000	char(*size*), 1 <= size <= 8000
char(*size*), 8001 <= size <= 64000	varchar(max)
clob	varchar(max)

Teradata ソースデータ型	Microsoft Azure Synapse Analytics ターゲットデータ型
date	date
decimal(p,s), 1 <= p <= 38, 1 <= s <= 37	decimal(p,s), 1 <= p <= 38, 1 <= s <= 37
float	float
integer	int
interval day(*precision*) to hour, 1 <= p <= 4	varchar(*size*), 5 <= size <= 8
interval day(*precision*) to minute, 1 <= p <= 4	varchar(*size*), 8 <= size <= 11
interval day(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 12 <= size <= 21
interval day(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval hour(*precision*) to minute, 1 <= p <= 4	varchar(*size*), 5 <= size <= 8
interval hour(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 9 <= size <= 18
interval hour(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval minute(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 9 <= size <= 18
interval minute(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval month(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval second(p,s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 3 <= size <= 12
interval year(*precision*) to month, 1 <= p <= 4	varchar(*size*), 5 <= size <= 8
interval year(*precision*), 1 <= p <= 4	varchar(*size*), 2 <=size <= 5
json	varchar(max)
mbr	varbinary(256)
number(*,s), 0 <= s <= 37	varchar（255）
number(p,s), 1 <= p <= 38, 1 <= s <= 37	decimal(p,s), 1 <= p <= 38, 1 <= s <= 37
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	decimal(p,s), 1 <= p <= 38, 1 <= s <= 37
period(date)	varchar(28)
period(time(*precision*) with time zone), 0 <= p <= 6	varchar(*size*), 36 <= size <= 50
period(time(*precision*)), 0 <= p <= 6	varchar(*size*), 24 <= size <= 38

Teradata ソースデータ型	Microsoft Azure Synapse Analytics ターゲットデータ型
period(timestamp(*precision*) with time zone), 0 <= p <= 6	varchar(*size*), 58 <= size <= 72
period(timestamp(*precision*)), 0 <= p <= 6	varchar(*size*), 46 <= size <= 60
smallint	smallint
time(*precision*) with time zone, 0 <= p <= 6	datetimeoffset(*precision*), 0 <= p <= 6
time(*precision*), 0 <= p <= 6	time(*precision*), 0 <= p <= 6
timestamp(*precision*) with time zone, 0 <= p <= 6	datetimeoffset(*precision*), 0 <= p <= 6
timestamp(*precision*), 0 <= p <= 6	datetime2(*precision*), 0 <= p <= 6
varbyte(*precision*), 1 <= p <= 64000	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 64000	varchar(*size*), 1 <= size <= max
varray	varbinary(max)
xml	varchar(max)

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Teradata データ型はサポートされません。

- ARRAY
- Blob
- CLOB
- JSON
- ST_GEOMETRY
- XML

注: サポートされていないデータ型のほとんどがデフォルトのマッピングに表示されます。ただし、これらのマッピングでは null がレプリケートされます。

Teradata ソースと Oracle ターゲット

次の表に、Teradata ソースと Oracle ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示します。

Teradata ソースデータ型	Oracle ターゲットデータ型
array	blob
bigint	number(19)
blob	blob

Teradata ソースデータ型	Oracle ターゲットデータ型
byte(*precision*), 1 <= p <= 64000	blob
byteint	number(3)
char(*size*), 1 <= size <= 2000	char(s byte), 1 <= s <= 2000
char(*size*), 2001 <= size <= 64000	clob
clob	clob
date	date
decimal(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
float	binary_double
integer	number(10)
interval day(*precision*) to hour, 1 <= p <= 4	char(s char), 5 <= s <= 8
interval day(*precision*) to minute, 1 <= p <= 4	char(s char), 8 <= s <= 11
interval day(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	char(s char), 12 <= s <= 21
interval day(*precision*), 1 <= p <= 4	char(s char), 2 <= s <= 5
interval hour(*precision*) to minute, 1 <= p <= 4	char(s char), 5 <= s <= 8
interval hour(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	char(s char), 9 <= s <= 18
interval hour(*precision*), 1 <= p <= 4	char(s char), 2 <= s <= 5
interval minute(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	char(s char), 9 <= s <= 18
interval minute(*precision*), 1 <= p <= 4	char(s char), 2 <= s <= 5
interval month(*precision*), 1 <= p <= 4	char(s char), 2 <= s <= 5
interval second(p,s), 1 <= p <= 4, 0 <= s <= 6	char(s char), 3 <= s <= 12
interval year(*precision*) to month, 1 <= p <= 4	char(s char), 5 <= s <= 8
interval year(*precision*), 1 <= p <= 4	char(s char), 2 <= s <= 5
json	clob
mbr	blob
number(*,s), 0 <= s <= 37	char(255 char)
number(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37

Teradata ソースデータ型	Oracle ターゲットデータ型
period(date)	char(28 char)
period(time(*precision*) with time zone), 0 <= p <= 6	char(s char), 36 <= s <= 50
period(time(*precision*)), 0 <= p <= 6	char(s char), 24 <= s <= 38
period(timestamp(*precision*) with time zone), 0 <= p <= 6	char(s char), 58 <= s <= 72
period(timestamp(*precision*)), 0 <= p <= 6	char(s char), 46 <= s <= 60
smallint	number(5)
time(*precision*) with time zone, 0 <= p <= 6	timestamp(*precision*) with time zone, 0 <= p <= 6
time(*precision*), 0 <= p <= 6	timestamp(*precision*), 0 <= p <= 6
timestamp(0)	date
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp(*precision*) with time zone, 0 <= p <= 6
timestamp(*precision*), 1 <= p <= 6	timestamp(*precision*), 1 <= p <= 6
varbyte(*precision*), 1 <= p <= 64000	blob
varchar(*size*), 1 <= size <= 1501	char(s byte), 1 <= s <= 1501
varchar(*size*), 2001 <= size <= 64000	clob
varray	blob
xml	clob

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Teradata データ型はサポートされていません。

- ARRAY
- Blob
- CLOB
- JSON
- ST_GEOMETRY
- XML

注: サポートされていないデータ型のほとんどがデフォルトのマッピングに表示されます。ただし、これらのマッピングでは null がレプリケートされます。

Teradata ソースと Snowflake ターゲット

次の表は、Teradata ソースと Snowflake ターゲットを使用した一括取り込みデータベース設定の推奨データ型マッピングを示しています。

Teradata **ソースデータ型**	Snowflake **ターゲットデータ型**
array	binary
bigint	integer
blob	binary
byte(*precision*), 1 <= p <= 64000	binary(*size*), 1 <= size <= 64000
byteint	integer
char(*size*), 1 <= size <= 64000	varchar(*size*), 4 <= size <= 256000
clob	varchar
date	date
decimal(*precision*), 1 <= p <= 38	integer
decimal(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
float	float
integer	integer
interval day(*precision*) to hour, 1 <= p <= 4	varchar(*size*), 5 <= size <= 8
interval day(*precision*) to minute, 1 <= p <= 4	varchar(*size*), 8 <= size <= 11
interval day(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 12 <= size <= 21
interval day(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval hour(*precision*) to minute, 1 <= p <= 4	varchar(*size*), 5 <= size <= 8
interval hour(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 9 <= size <= 18
interval hour(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval minute(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 9 <= size <= 18
interval minute(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval month(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval second(p,s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 3 <= size <= 12
interval year(*precision*) to month, 1 <= p <= 4	varchar(*size*), 5 <= size <= 8
interval year(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5

Teradata ソースデータ型	Snowflake ターゲットデータ型
json	varchar
mbr	binary(256)
number(*,s), 0 <= s <= 37	char(255)
number(*precision*), 1 <= p <= 36	integer
number(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
numeric(*precision*), 1 <= p <= 36	integer
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
period(date)	varchar(28)
period(time(*precision*) with time zone), 0 <= p <= 6	varchar(*size*), 36 <= size <= 50
period(time(*precision*)), 0 <= p <= 6	varchar(*size*), 24 <= size <= 38
period(timestamp(*precision*) with time zone), 0 <= p <= 6	varchar(*size*), 58 <= size <= 72
period(timestamp(*precision*)), 0 <= p <= 6	varchar(*size*), 46 <= size <= 60
smallint	integer
time(*precision*) with time zone, 0 <= p <= 6	timestamp_tz(*precision*), 0 <= p <= 6
time(*precision*), 0 <= p <= 6	time(*precision*), 0 <= p <= 6
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp_tz(*precision*), 0 <= p <= 6
timestamp(*precision*), 0 <= p <= 6	timestamp_ntz(*precision*), 0 <= p <= 6
varbyte(*precision*), 1 <= p <= 64000	binary(*size*), 1 <= size <= 64000
varchar(*size*), 1 <= size <= 64000	varchar(*size*), 4 <= size <= 256000
varray	binary
xml	varchar

注: Snowflake TIMESTAMP_TZ データ型には、ソースの Teradata TIME WITH TIME ZONE データ型に含まれていないデフォルトの日付が含まれています。たとえば、データベース取り込みジョブは、ソース値「12:59:59」を「1900-01-01 12:59:59」としてレプリケートします。

サポートされていないソースデータ型

一括取り込みデータベースでは、次の Teradata データ型はサポートされません。

- ARRAY
- Blob
- CLOB

- JSON
- ST_GEOMETRY
- XML

注: サポートされていないデータ型のほとんどがデフォルトのマッピングに表示されます。ただし、これらのマッピングでは null がレプリケートされます。

索引

A

Amazon Redshift
  データベース取り込みのターゲットに関する考慮事項 66
Amazon Redshift ターゲット
  SQL Server ソースを使用したマッピング 205
  DB2 for i ソースを使用したマッピング 169
  DB2 for Linux、UNIX、および Windows ソースを使用したマッピング 185
  DB2 for z/OS ソースを使用したマッピング 196
  Microsoft SQL Server ソースを使用したマッピング 205
  MongoDB ソースを使用したマッピング 221
  MySQL ソースを使用したマッピング 224
  Netezza ソースを使用したマッピング 241
  Oracle ソースを使用したマッピング 250
  PostgreSQL ソースを使用したマッピング 265
  SAP HANA ソースを使用したマッピング 282
  Teradata ソースを使用したマッピング 295
Amazon S3
  データベース取り込みのターゲットに関する考慮事項 66
Amazon S3 ターゲット
  Parquet データ型から Microsoft SQL Server 型へのマッピング 213
Apache Kafka
  データベース取り込みのターゲットに関する考慮事項 73
Azure Event Hubs
  データベース取り込みのターゲットに関する考慮事項 73
Azure Managed Instance
  ソースの準備と使用 27
Azure SQL データベース
  ソースの準備と使用 27

C

Cloud アプリケーション統合コミュニティ
  URL 7
Cloud 開発者コミュニティ
  URL 7
Confluent Kafka
  データベース取り込みのターゲットに関する考慮事項 73

D

Databricks Delta
  データベース取り込みのターゲットに関する考慮事項 68
Databricks Delta ターゲット
  DB2 for i ソースを使用したマッピング 171
  DB2 for Linux、UNIX、および Windows ソースを使用したマッピング 187
  DB2 for z/OS ソースを使用したマッピング 197
  Microsoft SQL Server ソースを使用したマッピング 206
  MongoDB ソースを使用したマッピング 222
  MySQL ソースを使用したマッピング 226
  Netezza ソースを使用したマッピング 242
  Oracle ソースを使用したマッピング 253
Databricks Delta ターゲット *(続く)*
  PostgreSQL ソースを使用したマッピング 268
  SAP HANA ソースを使用したマッピング 284
  Teradata ソースを使用したマッピング 297
DB2 for i ソース
  Amazon Redshift ターゲットを使用したマッピング 169
  Databricks Delta ターゲットを使用したマッピング 171
  Google BigQuery ターゲットを使用したマッピング 172
  Microsoft Azure Synapse Analytics ターゲットを使用したマッピング 178
  Microsoft SQL Server ターゲットを使用したマッピング 180
  Oracle ターゲットを使用したマッピング 181
  PostgreSQL ターゲットを使用したマッピング 183
  Snowflake ターゲットを使用したマッピング 184
DB2 for Linux、UNIX、および Windows ソース
  Amazon Redshift ターゲットを使用したマッピング 185
  Databricks Delta ターゲットを使用したマッピング 187
  Google BigQuery ターゲットを使用したマッピング 188
  Microsoft Azure Synapse Analytics ターゲットを使用したマッピング 190
  Microsoft SQL Server ターゲットを使用したマッピング 192
  Oracle ターゲットを使用したマッピング 193
  Snowflake ターゲットを使用したマッピング 194
Db2 for z/OS ソース
  CDC のストアドプロシージャのセットアップ 20
DB2 for z/OS ソース
  ソースの準備と使用に関する考慮事項 19
  Amazon Redshift ターゲットを使用したマッピング 196
  Databricks Delta ターゲットを使用したマッピング 197
  Google BigQuery ターゲットを使用したマッピング 198
  Microsoft Azure Synapse Analytics ターゲットを使用したマッピング 201
  Oracle ターゲットを使用したマッピング 202
  Snowflake ターゲットを使用したマッピング 203

G

Google BigQuery
  データベース取り込みのターゲットに関する考慮事項 70
Google BigQuery ターゲット
  DB2 for i ソースを使用したマッピング 172
  DB2 for Linux、UNIX、および Windows ソースを使用したマッピング 188
  DB2 for z/OS ソースを使用したマッピング 198
  Microsoft SQL Server ソースを使用したマッピング 208
  MongoDB ソースを使用したマッピング 222
  MySQL ソースを使用したマッピング 228
  Netezza ソースを使用したマッピング 243
  Oracle ソースを使用したマッピング 254
  PostgreSQL ソースを使用したマッピング 270
  SAP HANA ソースを使用したマッピング 285
  Teradata ソースを使用したマッピング 299
Google Cloud Storage
  データベース取り込みのターゲットに関する考慮事項 66

Google Cloud Storage ターゲット
Parquet データ型から Microsoft SQL Server 型へのマッピング 213

I

Informatica Intelligent Cloud Services
Web サイト 7
Informatica グローバルカスタマサポート
連絡先情報 8

M

Microsoft Azure Data Lake Storage
データベース取り込みのターゲットに関する考慮事項 66
Microsoft Azure Data Lake Storage Gen2 ターゲット
Parquet データ型から Microsoft SQL Server 型へのマッピング 213
Microsoft Azure SQL Database ソース
Microsoft Azure Synapse Analytics ターゲットを使用したマッピング 214
Snowflake ターゲットを使用したマッピング 220
Microsoft Azure Synapse Analytics
データベース取り込みのターゲットに関する考慮事項 76
Microsoft Azure Synapse Analytics ターゲット
DB2 for i ソースを使用したマッピング 178
DB2 for Linux、UNIX、および Windows ソースを使用したマッピング 190
DB2 for z/OS ソースを使用したマッピング 201
Microsoft Azure SQL Database ソースを使用したマッピング 214
Microsoft SQL Server ソースを使用したマッピング 214
MySQL ソースを使用したマッピング 232
Netezza ソースを使用したマッピング 245
Oracle ソースを使用したマッピング 258
PostgreSQL ソースを使用したマッピング 274
SAP HANA ソースを使用したマッピング 288
Teradata ソースを使用したマッピング 304
Microsoft Fabric OneLake
データベース取り込みのターゲットに関する考慮事項 66
Microsoft Fabric OneLake ターゲット
Parquet データ型から Microsoft SQL Server 型へのマッピング 213
Microsoft SQL Server
ソースの準備と使用 27
データベース取り込みのターゲットに関する考慮事項 77
Microsoft SQL Server ソース
Amazon Redshift ターゲットを使用したマッピング 205
Databricks Delta ターゲットを使用したマッピング 206
Google BigQuery ターゲットを使用したマッピング 208
Microsoft Azure Synapse Analytics ターゲットを使用したマッピング 214
Oracle ターゲットを使用したマッピング 218
Parquet 出力を使用したターゲットへのデータ型マッピング 213
Snowflake ターゲットを使用したマッピング 220
変更キャプチャメカニズム 31
Microsoft SQL Server ターゲット
DB2 for i ソースを使用したマッピング 180
DB2 for Linux、UNIX、および Windows ソースを使用したマッピング 192
MongoDB ソースを使用したマッピング 223
MySQL ソースを使用したマッピング 234
Netezza ソースを使用したマッピング 246
Oracle ソースを使用したマッピング 259
SAP HANA ソースを使用したマッピング 290
MongoDB ソース
Azure Synapse Analytics ターゲットを使用したマッピング 222
Amazon Redshift ターゲットを使用したマッピング 221
Databricks Delta ターゲットを使用したマッピング 222
Google BigQuery ターゲットを使用したマッピング 222
MongoDB ソース *(続く)*
Microsoft SQL Server ターゲットを使用したマッピング 223
Oracle ターゲットを使用したマッピング 223
Snowflake ターゲットを使用したマッピング 223
MySQL ソース
Amazon Redshift ターゲットを使用したマッピング 224
Databricks Delta ターゲットを使用したマッピング 226
Google BigQuery ターゲットを使用したマッピング 228
Microsoft Azure Synapse Analytics ターゲットを使用したマッピング 232
Microsoft SQL Server ターゲットを使用したマッピング 234
Oracle ターゲットを使用したマッピング 237
Snowflake ターゲットを使用したマッピング 239

N

Netezza ソース
Amazon Redshift ターゲットを使用したマッピング 241
Databricks Delta ターゲットを使用したマッピング 242
Google BigQuery ターゲットを使用したマッピング 243
Microsoft Azure Synapse Analytics ターゲットを使用したマッピング 245
Microsoft SQL Server ターゲットを使用したマッピング 246
Oracle ターゲットを使用したマッピング 247
Snowflake ターゲットを使用したマッピング 249

O

Oracle
データベース取り込みのターゲットに関する考慮事項 78
Oracle Cloud Object Storage
データベース取り込みのターゲットに関する考慮事項 66
Oracle ソース
Amazon Redshift ターゲットを使用したマッピング 250
Databricks Delta ターゲットを使用したマッピング 253
Google BigQuery ターゲットを使用したマッピング 254
Microsoft Azure Synapse Analytics ターゲットを使用したマッピング 258
Oracle Data Guard プライマリおよびスタンバイデータベースソース 55
Oracle ターゲットを使用したマッピング 261
PostgreSQL ターゲットを使用したマッピング 262
Snowflake ターゲットを使用したマッピング 264
SQL Server ターゲットを使用したマッピング 259
アーカイブログの保持 57
Oracle ターゲット
DB2 for i ソースを使用したマッピング 181
DB2 for Linux、UNIX、および Windows ソースを使用したマッピング 193
DB2 for z/OS ソースを使用したマッピング 202
Microsoft SQL Server ソースを使用したマッピング 218
MongoDB ソースを使用したマッピング 223
MySQL ソースを使用したマッピング 237
Netezza ソースを使用したマッピング 247
Oracle ソースを使用したマッピング 261
PostgreSQL ソースを使用したマッピング 277
SAP HANA ソースを使用したマッピング 291
Teradata ソースを使用したマッピング 306

P

PopstgreSQL ターゲット
DB2 for i ソースを使用したマッピング 183
PostgreSQL
データベース取り込みのターゲットに関する考慮事項 80

PostgreSQL ソース
Amazon Redshift ターゲットを使用したマッピング 265
Databricks Delta ターゲットを使用したマッピング 268
Google BigQuery ターゲットを使用したマッピング 270
Microsoft Azure Synapse Analytics ターゲットを使用したマッピング 274
Oracle ターゲットを使用したマッピング 277
Snowflake ターゲットを使用したマッピング 279
PostgreSQL ターゲット
Oracle ソースを使用したマッピング 262

S

SAP HANA ソース
Amazon Redshift ターゲットを使用したマッピング 282
Databricks Delta ターゲットを使用したマッピング 284
Google BigQuery ターゲットを使用したマッピング 285
Microsoft Azure Synapse Analytics ターゲットを使用したマッピング 288
Microsoft SQL Server ターゲットを使用したマッピング 290
Oracle ターゲットを使用したマッピング 291
Snowflake ターゲットを使用したマッピング 293
Snowflake
データベース取り込みのターゲットに関する考慮事項 81
Snowflake ターゲット
DB2 for i ソースを使用したマッピング 184
DB2 for Linux、UNIX、および Windows ソースを使用したマッピング 194
DB2 for z/OS ソースを使用したマッピング 203
Microsoft Azure SQL Database ソースを使用したマッピング 220
Microsoft SQL Server ソースを使用したマッピング 220
MongoDB ソースを使用したマッピング 223
MySQL ソースを使用したマッピング 239
Netezza ソースを使用したマッピング 249
Oracle ソースを使用したマッピング 264
PostgreSQL ソースを使用したマッピング 279
SAP HANA ソースを使用したマッピング 293
Teradata ソースを使用したマッピング 309

T

Teradata ソース
Amazon Redshift ターゲットを使用したマッピング 295
Databricks Delta ターゲットを使用したマッピング 297
Google BigQuery ターゲットを使用したマッピング 299
Microsoft Azure Synapse Analytics ターゲットを使用したマッピング 304
Oracle ターゲットを使用したマッピング 306
Snowflake ターゲットを使用したマッピング 309

W

Web サイト 7

あ

アップグレード通知 8

し

システムステータス 8

す

ステータス
Informatica Intelligent Cloud Services 8

た

ターゲット、データベース取り込み
Amazon S3 66
Apache Kafka 73
Confluent Kafka 73
Google Cloud Storage 66
Kafka 対応 Azure Event Hubs 73
Microsoft Azure Data Lake Storage 66
Microsoft Azure Synapse Analytics 76
Microsoft Fabric OneLake 66
Oracle Cloud Object Storage 66
Snowflake 81
フラットファイル 66

て

データベース取り込み
アーキテクチャ 12
概要 9
使用例 9
データベース取り込みジョブ
CDC のストアドプロシージャのセットアップ 20
ジョブのデプロイ解除 167
ジョブの再デプロイ 166
ジョブの再開 164
スキーマドリフトオプションのオーバーライド 165
スケジュールに基づいたジョブの実行 163
ソースオブジェクトとターゲットオブジェクトの再同期 168
監視からのデプロイされたジョブの実行 163
停止 163
データベース取り込みタスク
Azure SQL Database のソースに関する考慮事項 27
Amazon Redshift ターゲットのプロパティ 119
Amazon S3 ターゲットのプロパティ 120
Azure Managed Instance ソースに関する考慮事項 27
Databricks Delta ターゲットのプロパティ 124
DB2 for i ソース 15
DB2 for LUW ソースの準備と考慮事項 18
DB2 for z/OS ソース 19
Google BigQuery ターゲットのプロパティ 129
Google Cloud Storage ターゲットのプロパティ 132
Kafka ターゲットのプロパティ 136
Microsoft Azure Data Lake Storage ターゲットプロパティ 139
Microsoft Azure Synapse Analytics ターゲットのプロパティ 143
Microsoft Fabric OneLake ターゲットプロパティ 144
Microsoft SQL Server ソースに関する考慮事項 27
Microsoft SQL Server ターゲットのプロパティ 146
MongoDB ソース 34
MySQL ソース 35
Netezza ソースに関する考慮事項 37
Oracle ソースに関する考慮事項 38
Oracle ソース権限 45
Oracle ターゲットのプロパティ 147
PostgreSQL ソースに関する考慮事項 58
PostgreSQL ターゲットプロパティ 153
RDS for SQL Server ソースに関する考慮事項 27
SAP HANA ソース 62
Snowflake ターゲットのプロパティ 153
Teradata ソースに関する考慮事項 65
ソースタイプ 10

データベース取り込みタスク *(続く)*
ソースに関する考慮事項 15
ソースの設定 102
ターゲットテーブルの名前変更 116
ターゲットの考慮事項 66
ターゲットの設定 115
タスクの基本情報の定義 101
テーブル選択ルールの例 115
フラットファイルターゲットのプロパティ 126
ランタイムオプションの設定 157
取り込みタスクのデプロイ 162
制限 14
設定タスクフロー 100
データベース取り込みのターゲットのプロパティ
Amazon Redshift ターゲットのプロパティ 119
Amazon S3 ターゲットのプロパティ 120
Databricks Delta ターゲットのプロパティ 124
Google BigQuery ターゲットのプロパティ 129
Google Cloud Storage ターゲットのプロパティ 132
Kafka のプロパティ 136
Microsoft Azure Data Lake Storage ターゲットプロパティ 139
Microsoft Azure Synapse Analytics ターゲットのプロパティ 143
Microsoft Fabric OneLake ターゲットプロパティ 144
Microsoft SQL Server ターゲットのプロパティ 146
Oracle ターゲットのプロパティ 147
データベース取り込みのターゲットのプロパティ *(続く)*
PostgreSQL ターゲットプロパティ 153
Snowflake ターゲットのプロパティ 153
フラットファイルターゲットのプロパティ 126
データ型マッピング
デフォルトマッピングのカスタマイズ 117

ふ

フラットファイル
データベース取り込みのターゲットに関する考慮事項 66

め

メンテナンスの停止 8

り

リスタートおよびリカバリ
データベース取り込み増分ロードジョブ 169