OpenStaging 技術解説書

R 1.1.0

OpenStaging 技術解説書

R 1.1.0

商標

本書に記載されている社名および商品名は、それぞれ各社の商標または登録商標です。

なお、本文中には TM、® マークは明記しておりません。

目次

1 技術解説書

1.1 対応データベースの追加

OpenStaging では、標準で以下の 3 つのデータベースに対応しています。

- PostgreSQL 8.0
- MySQL 5.0
- Oracle Database 10g

またいくつかの XML ファイル、ストアドプロシージャ、Java プログラムを作成し、設定ファイルに項目を追加することで、既存のソースコードを改変することなく対応するデータベースを新たに追加することが可能です。

以下の手順で、データベースを追加することができます。

1. JDBC ドライバのインストール
2. SQL 定義ファイルの作成
3. ストアドプロシージャ・テンプレートの作成
4. SQLCreatorDAO 実装サブクラスの作成
5. DBUtil 実装クラスの作成
6. 設定ファイルに対応データベースの定義を追加
7. データベース間のデータ型変換を定義

1.1.1 OpenStaging における ETL 処理概要

OpenStaging では高速化のため、ETL 処理にストアドプロシージャを利用しています。ETL ジョブをコンパイルするときに、用意されているテンプレートから最終的なストアドプロシージャのコードを生成します。

テンプレートはデータベースごとに、1 つの全体テンプレートと、複数の個別テンプレートから構成されます。

コンパイルのとき、全体テンプレート内のキーワード「% ～ %」に対応する、①個別テンプレートのコード、② SQL 文を組み込んでストアドプロシージャのコードを生成します（テンプレートの詳細については『1.1.4 ストアドプロシージャ・テンプレートの作成』を参照）。

1.1.2 JDBC ドライバのインストール

OpenStaging に新しくデータベースを追加するには、そのデータベース用の JDBC ドライバが必要です。ベンダのサイトからダウンロードする等の方法で JDBC ドライバを入手します。入手した JDBC ドライバは「/WEB-INF/lib」ディレクトリの直下にコピーします。

1.1.3 SQL 定義ファイルの作成

データベースを追加する場合、SQL の構文はデータベースによって異なるため、以下の手順で SQL 定義ファイルを作成する必要があります。

1. 追加するデータベースの構文を定義した XML ファイル（SQL 定義ファイル）を「/WEB-INF/conf/sql」ディレクトリの直下に作成します。
2. 既存の SQL 定義ファイル（「pg_sql.xml」等）を参考に、SQL 定義ファイルを作成します。
3. 任意のファイル名で保存します。

1.1.4 ストアドプロシージャ・テンプレートの作成

ストアドプロシージャの原型となる全体テンプレートと個別テンプレートを作成します。テンプレートはデータベースごとに、1 つの全体テンプレートと複数の個別テンプレートで構成されています。データベースを追加する場合は、既存のデータベースの中で追加するデータベースに最も近いテンプレートをベースにカスタマイズを行うのが良いでしょう。

・ 最も近いデータベースの選択基準

	パターン	データベース
A	フェッチのときにレコード変数が使用できない場合	MySQL
B	A 以外の場合で、コミット可能な場合	Oracle
C	A、B 以外の場合	PostgreSQL

ベースとなるデータベースのプロシージャを、追加するデータベース用にカスタマイズします。
例)PostgreSQL で定義しているテンプレート

・ 全体テンプレート ：「pg_etl_template.sql」

・ 個別テンプレート：

処理	ファイル名
ルックアップ処理	pg_etl_lookup_template.sql
ルックアップ除外処理	pg_etl_lookup_remove_template.sql
Load 処理	pg_etl_load_template.sql
DimensionLoad 処理	pg_etl_load_dimension_template.sql
FactLoad 処理	pg_etl_load_fact_template.sql

1.1.4.1 全体テンプレート

全体テンプレートは、ETL処理の中心的役割を果たすテンプレートです。いくつかのプロシージャコードのほかに、正しい順序で個別テンプレートを組み込むためのキーワード「%～%」が指定されています。データベースを追加する場合、ベースとなるデータベースの全体テンプレートを参考に全体テンプレートを作成します。

1.1.4.2 個別テンプレート

個別テンプレートは、主に、全体テンプレート中のキーワード「%～%」に組み込まれるストアドプロシージャの部品です。Loadやルックアップ等、処理内容によって個別に作成します。

データベースを追加する場合、ベースとなるデータベースの個別テンプレートを参考に個別テンプレートを作成します。

1.1.5 SQLCreatorDAO実装サブクラスの作成

作成したテンプレートは、SQLCreatorDAOによってキーワードの置換等が行われ、最終的なストアドプロシージャが生成されます。

SQLCreatorDAOクラスは抽象クラスであり、対応データベースごとに実装サブクラスの作成が必要です。

[実装するメソッド]

- protected void prepareProperties throws Exception
  受け取った引数のMapに、全体テンプレートに含まれるキーワードと、対応するコードやSQLとのマッピングを、引数の情報から生成し、格納します。

(引数)

型	説明
ETLJob	パースされたETLジョブ
TableViewWorkObject[]	ソースとなるテーブルを表すTableViewWorkObjectの配列
ExtractWorkObject	Extract処理を表すExtractWorkObject
TransformWorkObject	Transform処理を表すTransformWorkObject
LoadWorkObject	Loadを表すLoadWorkObject
LookupWorkObject[]	ルックアップ処理を表すLookupWorkObjectの配列
ConnectObject	データベースへの接続情報を表すConnectObject
SQLCreatorTemplates	SQL定義情報を保持する、データベースごとに実装されたSQLCreatorTemplates
Map<String, String>	キーワードと置換文字列のマッピングを格納する、空のMap

(戻り値)なし

- protected String getProcVarName throws Exception
  「transform/lookup/load」クラスのindexで定義される変数に対して、プロシージャ内で実際に使用される変数名を返します。

(引数)

型	説明
ExtractWorkObject	Extract 処理を表す ExtractWorkObject
TransformWorkObject[]	Transform 処理を表す TransformWorkObject
int	Transform の走査対象位置 例えば、最初の transform で出現した場合は 0、 Load/ ルックアップ処理で出現した場合は transforms.length
int	Job の XML に出現する変数を表すインデックス
ConnectObject	データベースへの接続情報を表す ConnectObject
SQLCreatorTemplates	SQL 定義情報を保持する、データベースごとに実装された SQLCreatorTemplates

(戻り値)

型	説明
String	プロシージャ内で実際に使用される変数名

- protected String getProcVarName throws Exception
  上記 transform 内で変数が定義されていない場合、extract に必ず存在するはずなので、その変数名を返します。主に、上記の getProcVarName メソッドから呼ばれます。

(引数)

型	説明
ExtractWorkObject	抽出処理を表す ExtractWorkObject
SelectItem	ExtractWorkObject 内の SelectItem
ConnectObject	データベースへの接続情報を表す ConnectObject
SQLCreatorTemplates	SQL 定義情報を保持する、データベースごとに実装された SQLCreatorTemplates

(戻り値)

型	説明
String	プロシージャ内で実際に使用される変数名

- protected void registProcedure throws Exception
  プロシージャのコンパイルを行います。コンパイルに失敗した場合、必ずエラーメッセージを含む SQLException をスローします。

(引数)

型	説明
ExtractWorkObject	抽出処理を表す ExtractWorkObject
SelectItem	ExtractWorkObject 内の SelectItem
ConnectObject	データベースへの接続情報を表す ConnectObject
SQLCreatorTemplates	SQL 定義情報を保持する、データベースごとに実装された SQLCreatorTemplates

(戻り値)なし

1.1.6 DBUtil実装クラスの作成

DBUtilは、データベースごとに固有のデータベース接続、テーブル情報取得等の処理を行うメソッドを宣言したインターフェースです。また、SQLCreatorDAOの実装クラスのオブジェクトや、SQL定義情報を保持するSQLCreatorTemplatesのオブジェクトの取得も、このインターフェースを通じて行います。
対応するデータベースを追加するには、このインターフェースを実装したクラスが必要です。

[実装するメソッド]

・ public String void getJDBCDriverName
JDBCドライバクラスの完全修飾名を返します。

(引数)なし
(戻り値)

型	説明
String	JDBCドライバクラスの完全修飾名

・ public Connection getConnection throws SQLException, Exception
引数の情報から、データベースへの接続を行い、Connectionを返します。

(引数)

型	説明
ConnectObject	データベースへの接続情報を表すConnectObject

(戻り値)

型	説明
Connection	データベースのConnection

・ public DBRepository getDBRepository throws SQLException, Exception
引数の情報からデータベースへの接続を行い、現在のテーブル、カラム、インデックスの構成を取得し、結果をDBRepositoryオブジェクトに格納して返します。

(引数)

型	説明
ConnectObject	データベースへの接続情報を表すConnectObject

(戻り値)

型	説明
DBRepository	テーブル、カラム、インデックスの構成を格納した、DBRepositoryオブジェクト

・ public boolean isTableExist throws SQLException
データベース上に、引数で指定された名前のテーブルもしくはビューが存在するかどうかを確認し、存在すればtrueを返します。

（引数）

型	説明
String	存在を確認するテーブル名
Connection	データベース接続オブジェクト。この Connection を利用して判定を行う。
ConnectObject	データベースへの接続情報を表す ConnectObject

（戻り値）

型	説明
boolean	該当する名前のテーブル（ビュー）が存在する場合 true

- public boolean isIndexExist throws SQLException
  引数で指定された Connection のデータベース上の指定された名前のテーブル上に、指定された名前のインデックスが存在するかどうかを確認し、存在すれば true を返します。

（引数）

型	説明
String	インデックスの存在を確認するテーブル名
String	存在を確認するインデックスのインデックス名
Connection	データベース接続オブジェクト。この Connection を利用して判定を行う。
ConnectObject	データベースへの接続情報を表す ConnectObject

（戻り値）

型	説明
boolean	該当するテーブルに、該当するインデックス名が存在する場合 true

- public SQLCreatorDAO getSQLCreatorDAO
  このデータベース用の、SQLCreatorDAO 実装サブクラスのオブジェクトを返します。

（引数）なし

（戻り値）

型	説明
SQLCreatorDAO	このデータベース用の、SQLCreatorDAO の実装サブクラスオブジェクト

- public Object convertSQLType throws SQLException
  このデータベース用 JDBC ドライバの、ResultSet.getObject() メソッドが返す可能性のある型のうち、JDBC 汎用型でないものを、JDBC 汎用型に変換して返します。

（引数）

型	説明
Object	ResultSet が getObject() で返す Object

（戻り値）

型	説明
Object	JDBC 汎用型に変換された Object 引数の Object がもともと JDBC 汎用型であれば、その Object がそのまま返される

なお、OpenStaging では、これらのメソッドのうち、原則共通と思われるメソッドのみを実装した抽象クラス GenericDBUtil クラスを用意していますので、通常はこのクラスを継承して、残りのメソッドの実装を行います。

1.1.7　設定ファイルに対応データベースの定義を追加

OpenStaging では、対応するデータベースを、「/WEB-INF/classes」ディレクトリ直下の「settings.properties」ファイルで定義しています。

データベースを識別するために、固有の名称を決定します。ここでは例として「NewDB」という名称を使用します。

1. DBUtil 実装クラス名を設定します。

   DBUtil.NewDB=[作成した DBUtil 実装クラスの完全修飾名]
2. SQLCreatorTemplates が読み込むファイル名を設定します。

   SQLCreatorTemplates.NewDB=[SQL 定義ファイルのファイル名]
3. クラスパスの設定に JDBC ドライバを追加します。
   以下の項目の末尾に、「:」（コロン）と、JDBC ドライバのライブラリ（JAR ファイル or ZIP ファイル）の絶対パスを追加します。

   JobExecuter.classPath=[既存の設定値]:[JDBC ドライバの絶対パス]

1.1.8　対応データベース間の型変換を定義

複数のデータベースを扱う以上、場合によっては、同じ型でも指定できるサイズの上限・下限が異なっていたり、サイズを文字数で指定していたりバイト数で指定していたりすることがあります。

そのため、あるデータベースから別のデータベースへデータを移行する場合に、移行先のデータベースで利用可能な型に変換する必要があります。

OpenStaging では、このような型変換のルールを、型変換定義ファイル（/WEB-INF/conf/sql/sqltypes.xml）に記述して定義します。

1. SQLTypes 要素を、型グループとなる「Group」要素の下に追加します。
   XML 内の各「Group」要素に、「SQLTypes」要素を追加します。「SQLTypes」要素の「db」属性には、『1.1.7 設定ファイルに対応データベースの定義を追加』 で決定したデータベースの名称（ここでは「NewDB」）を指定します。

例)<Group>
  <SQLTypes db=”NewDB”>
  </SQLTypes>
  <SQLTypes db=”PostgreSQL”>
        ……

2. SQLType 要素を、「SQLTypes」要素の下に追加します。
追加した「SQLTypes」要素の下に、変換後の型を優先順位の高い順に<SQLType>として追加します。
「SQLType」要素には、以下の属性を指定することができます。

属性名	説明
name（必須）	型の名称
length	特定のサイズの場合だけ変換を適用したい場合はサイズを、サイズの指定ができない型の場合は「false」を、いずれにも該当しない場合は属性を指定しません。
destonly	変換元としては利用せず、変換後の型としてだけ利用する場合に「true」を指定します。
sizeRatio	変換時の型のサイズ比率を指定します。デフォルトは「1」。 *length=「false」の場合は不要です。
maxLength	型のサイズの最大値。変換後、このサイズを越えてしまう場合は次の候補の型を探します。
minLength	型のサイズの最小値。変換後、このサイズを下回ってしまう場合は次の候補の型を探します。
unused	変換に利用しない型の場合に「true」を指定します。

例)PostgreSQL → NewDB に型変換する場合
ここでは*1・・・varchar、text の順に変換を試みるように指定、
*2・・・PostgreSQL が text 型の場合グループ１を使用せずグループ２を使用するように指定します。

```
<Group>・・・*1（グループ１）
<Name>可変長文字列</Name>
<SQLTypes db=”NewDB”>
  <SQLType name=”varchar” maxLength=”255” sizeRatio=”2”/>
  <SQLType name=”text” length=”false”/>
  <SQLType name=”char” maxLength=”255”/>
</SQLTypes>
<SQLTypes db=”PostgreSQL”>
  <SQLType name=”varchar”/>
  <SQLType name=”text” maxLength=”20” destonly=”true”/>
</SQLTypes>
<Group>
<Group>・・・*2（グループ２）
<Name>可変長文字列</Name>
<SQLTypes db=”NewDB”>
  <SQLType name=”varchar” maxLength=”255” sizeRatio=”2”/>
  <SQLType name=”text” length=”false”/>
</SQLTypes>
<SQLTypes db=”PostgreSQL”>
  <SQLType name=”varchar”/>
  <SQLType name=”text” length=”false”/>
</SQLTypes>
<Group>
```

*1：PostgreSQL の sizeRatio= デフォルト（「1」）、NewDB の sizeRatio=2、

PostgreSQL の maxLongth= 指定なし NewDB の maxLength=255
→ NewDB 側では、PostgreSQL の 2 倍のサイズが必要であり（sizeRatio）、最大値が 255。そのため PostgreSQL の varchar 型のカラムのサイズが 255 の半分の 127 以下であれば、NewDB の varchar 型に変換され、それ以上の場合は次の候補である text 型に変換されます。

*2：PostgreSQL のグループ 1 の text 型は「destonly="true"」のため変換元の型として扱うことはできません。そのため「destonly="true"」指定されていないグループ 2 を使用して型変換を行います。

1.2 ジョブで使用する SQL のカスタマイズ

OpenStaging の ETL ジョブおよび、テーブル移行ジョブでは、それぞれ実行する SQL 文を生成し、ファイルに保存しています。また、ETL ジョブの SQL は、GUI からのカスタマイズが可能です（『OpenStaging ユーザーマニュアル』の「4.10 プログラムコードを編集する」を参照）。

SQL が記述されたファイルを手動でカスタマイズする場合は、以下のファイルを編集します。

オブジェクト	ファイル名
ETL ジョブ	/WEB-INF/conf/sql/[ジョブ ID]/ *.sql
テーブル移行ジョブ	/WEB-INF/conf/job/expimp/[ジョブ ID]/*.sql *テーブル移行ジョブの場合は、実行時にファイルが生成されます。

1.3 データソース、ジョブ、スケジュールの各設定ファイルの保存場所

OpenStaging では、登録されたデータソース、ジョブ、スケジュールは、XML 形式に変換され、以下に保存されます。

オブジェクト	ファイル名
データソース	/WEB-INF/conf/datasource/[データソース ID].xml
ETL ジョブ	/WEB-INF/conf/job/etl/[ジョブ ID].xml
テーブル移行ジョブ	/WEB-INF/conf/job/expimp/[ジョブ ID].xml
スケジュール	/WEB-INF/conf/schedule/[スケジュール ID].xml

メモ： 「template.xml」をカスタマイズするとそれ以降に作成されるオブジェクトに対して、設定が有効になります。

1.4 SQL 定義ファイルおよびテンプレートの格納場所

PostgreSQL、MySQL、Oracle の各データベースにおける、SQL 定義ファイルとストアドプロシージャのテンプレートは以下に保存されます。

定義ファイル /テンプレート	保存場所
SQL 定義ファイル	/WEB-INF/conf/sql/[dbname]_sql.xml
全体テンプレート	/WEB-INF/conf/sql/[dbname]_etl_template.sql
個別テンプレート (ルックアップ処理)	/WEB-INF/conf/sql/[dbname]_etl_lookup_template.sql
個別テンプレート (ルックアップ除外処理)	/WEB-INF/conf/sql/ [dbname]_etl_lookup_remove_template.sql
個別テンプレート (Load 処理)	/WEB-INF/conf/sql/[dbname]_etl_load _template.sql
個別テンプレート (Dimension Load 除外処理)	/WEB-INF/conf/sql/ [dbname]_etl_load_dimension_template.sql
個別テンプレート (Fact Load 除外処理)	/WEB-INF/conf/sql/ [dbname]_etl_load_fact_template.sql

メモ： ［*dbname*］は、PostgreSQL の場合「pg」、MySQL の場合「my」、Oracle の場合「orcl」を指定します。