D-014

オンライン対応型データ転送フレームワークの試作と評価

Evaluation of Prototype of Online Data Transfer Framework

米田 貴雄† 山足 光義†
Takao Yoneta Mitsuyoshi Yamatari

1. はじめに

企業の IT 化が進んだ今日では，既存システムとのデータ連携による業務の効率化や，データ統合による個別データ管理コストの削減など，システム間でのデータ連携のニーズが高まっている．

システム間データ連携では，既存システムへの影響が少ない ETL（Extract/Transform/Load）ツールの活用が一般的である．しかし，業務的な要件により，ETL ツールが苦手とするリアルタイム性や複雑性が求められる場合には，ETL ツールが導入できない場合があった．この場合，業務要件を満たすための個別のデータ連携プログラムを開発する必要があり，開発に要する期間や費用が大きくなる，また，柔軟性が低くなるという課題があった．

上記課題に対し，柔軟性のあるシステム構築が可能な ESB（Enterprise Service Bus）をアーキテクチャに組込み，設定だけでデータ転送プログラムの開発や改修が可能なフレームワークを提案した．本稿では，提案方式を実装したプロトタイプとその生産性について評価した結果について述べる．

2. 関連技術

2.1 システム間のデータ連携基盤

システム間のデータ連携を実現する基盤として ESB がある．ESB は，システム間のインタフェースの違いを吸収するためのプロトコル変換の機構を備えている．また，ルーティングと呼ばれるサービスの実行順序を定義することで，サービス間をメッセージ転送して複数サービスの連携を制御する機構を備えている．

ESB はオープンソースでも開発されており，ServiceMix や MuleESB といった製品が知られている[1][2]．これらのコミュニティから連携用の標準的な部品が数多く提供されており，これを組合せることで様々なインタフェースを持つシステムとの連携が容易に実現できる．

2.2 データベース操作のサービス化基盤

当社では，SQL 文などの定義を記載することでデータベース操作をサービス化する基盤（以下，DB サービス化基盤）を独自に開発して保有している．DB サービス化基盤の特長は，複数の SQL を組合せて一つのトランザクションで実行することが可能な点である．本稿では，この基盤を活用することを前提に記載しており，この基盤で生成されたサービスのことを本稿では DB サービスと呼ぶ．

一般的にデータベース操作をサービス化する方法としては，JDBC などを用いたデータアクセスロジックをサーブレット化する方法などがある．データ操作の実行結果を XML 形式のメッセージとして SOAP や REST などの通信方式で交換する方法などが知られている[3][4]．

3. 課題

システム間でリアルタイムなデータ転送が求められる場合には，転送先にとって意味のある単位のデータを，転送先データベースへのトランザクションを短くして更新する必要がある．このため，データ連携の数が増えた場合には，データ転送プログラムの流用を図ったとしても，次のような課題が残る．

- 転送先のデータ構造に合わせて更新順序などの処理を再プログラミングしなければならない
- 転送先が増えた場合には，追加分の更新処理をプログラミングしなければならない

以上より，連携の数が増えるにつれて改修範囲が大きくなるという課題があった．

4. 解決策

上記課題に対し，ESB を用いてデータ連携プログラムを開発するフレームワークを提案する．提案方式の構成を図 1に示す．提案方式は，ESB と DB サービス化基盤により構成される．

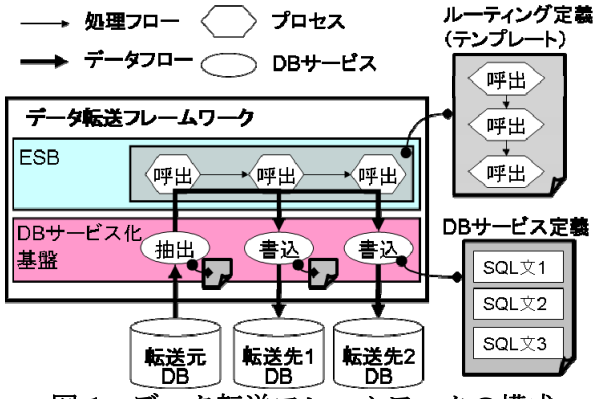

図 1 データ転送フレームワークの構成

提案方式の特長は次の通りである．

(1) 定義でサービスのデータアクセスを自由に変更

DB サービス化基盤は，DB サービス定義に記述された SQL や SQL の実行順序の定義を基にデータアクセスを実行するサービスを生成することができる．このため，データ構造に変更が合った場合にも，要件に合わせてデータの抽出や書込み処理を自由に変更することができる．

(2) データ転送用のルーティングテンプレートを利用

あらかじめ抽出用と書込み用の DB サービスを用いてデータ転送を行うルーティング定義のテンプレートを用意しておく．こうすることで，転送先などに変更がある場合には，テンプレートの修正で対応することが可能となる．

† 三菱電機株式会社, Mitsubishi Electric Corporation

5. 実現方式

データ転送フレームワークのプロトタイプを実装した内容について述べる．

5.1 ルーティングテンプレートの設計

ルーティング定義のテンプレートを図 2に示す．今回はタイマーを用いてデータ転送をコントロールする設計とした．

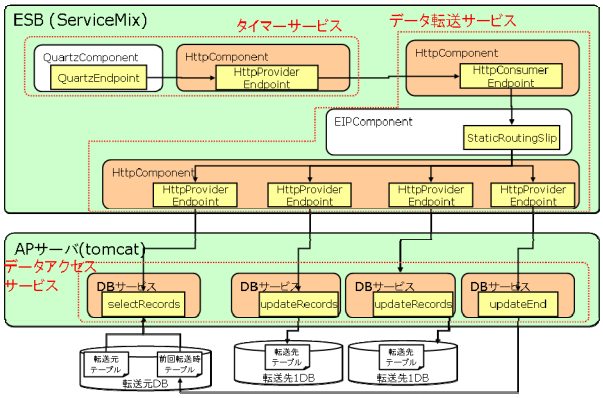

図 2 ルーティング定義

5.2 処理シーケンス

処理シーケンスを図 3に示す．

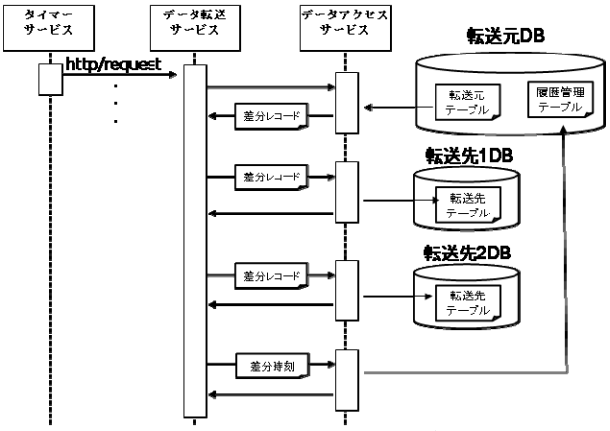

図 3 処理シーケンス概要

処理の手順は次の通りである．

1.タイマーから定期的にデータ転送サービスを呼出す
2.転送元から前回差分のデータを抽出する
3.転送先 1，2 に差分データを書込む
4.転送元の履歴管理テーブルを更新する．

なお，今回は転送元 DB に履歴管理のテーブルを設けて，転送対象データを抽出する方式とした．

6. 評価

フレームワークを適用した場合の生産性を評価した結果を述べる．

6.1 評価方法

評価は，従来方法と提案方法を用いた場合での新規開発時，データ構造変更時，転送先追加時にかかる開発量を比較する．従来方法は，Java でデータ転送を行うプログラムを開発した場合とする．基本のデータ転送は図 4に示すデータ構造 1 の構造を用いることとする．データ構造の変更については，データ構造 1 から 2 への変更とする．

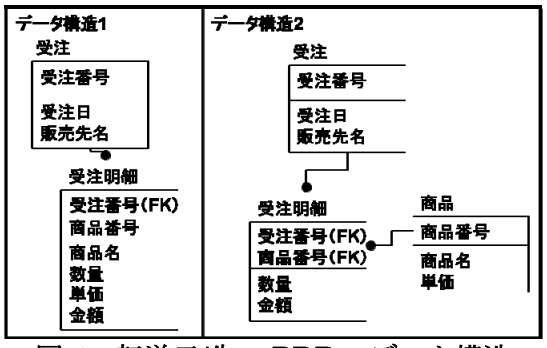

図 4 転送元/先の RDB のデータ構造

6.2 評価結果

開発量を比較した結果を表 1に示す．Java コードと定義ではあるが単純に新規・改修部分にかかる行数を比較すると，提案方法では従来方式に比べ新規開発時は 11.3%，データ構造変更時は 66.7%と削減できることがわかった．一方，転送先の追加では同程度となることが分かった．

表 1 開発量の比較

内容	従来[行] (Java)	提案[行] (定義)	比率[%]
新規開発時	新規：186	改修：21	11.3
データ構造変更時	改修：21 流用：168	改修：14	66.7
転送先追加時	改修：31 流用：186	改修：32	103.2

※実ステップ数のみをカウント

6.3 考察

今回は単純なデータ構造を持つ DB 間の転送を題材としたが，テーブル数や項目数が多くなるにつれて従来方式の方が，開発量が多くなる傾向が強かった．実システムへの適用においては複雑さが増すため，生産性の差はさらに顕著になると考える．

7. おわりに

本稿では，ESB を用いてデータ転送プログラムを開発するフレームワークを提案し試作した．生産性について評価した結果，プログラミングで開発する場合に対し，新規開発時とデータ構造変更時でそれぞれ生産性を向上できることが分かった．今後は転送パターンの拡充による対応要件の拡大や，設計ツールの開発による更なる開発の容易化を検討していく予定である．

【参考文献】

[1] The Apache Software Foundation,「ServiceMix」, http://servicemix.apache.org/
[2] MULESOFT.ORG,「Mule ESB」, http://www.mulesoft.com/mule-esb-open-source-esb
[3] IBM,「WSAD で Web サービスを作ってみよう！：第 6 回 DB2 から Web サービスを作る」, http://www.ibm.com/developerworks/jp/webservices/library/j_ws-wsad6/index.html, 2006.
[4] Oracle,「Oracle Application Server Web Services 開発者ガイド」, http://download.oracle.com/docs/cd/E18355_01/web.1013/B31868-01/devdbase.htm, 2006.