1D-2

データベース接続層における SQL 変換による DBMS 置き換え支援

藤山 健一郎　　喜田 弘司　　中村 暢達
ＮＥＣ　インターネットシステム研究所

1. はじめに

近年、稼動しているシステムのデータベース管理システム（DBMS）を、異なる種類の DBMS に置き換えたいというニーズがある。例えば、オープンソース DBMS で構築したシステムを、業務拡大のために高機能な商用 DBMS に移行する場合や、逆に、運用コスト削減等の理由から商用 DBMS をオープンソース DBMS に移行する場合などである。このような要求に対し、従来は旧 DBMS 用のアプリケーションプログラム（AP）を新 DBMS に対応するように改造することで対処してきた。しかし、複雑な AP の一部を、他に影響なく安全に改造するのは容易ではなく、またライセンス上、改造が不可能な場合もある。さらに、改造した AP を用いた置き換え後のシステムが、安定して動作するか検証することも困難である。

そこで本稿では、AP を変更するのではなく、AP が発行したデータ処理要求を途中で変換することで、DBMS 置き換えを支援する方式について述べる。また、DBMS 置き換え後のシステムの動作を検証する方式についても議論する。

2. 課題

AP が DBMS に対して行うデータ処理とは、DBMS 固有の「プロトコル」で DBMS に接続し、データ処理要求である「SQL」を投入し、「データ」を操作することである。故に DBMS を置き換える際には、このプロトコル、SQL、データを旧 DBMS 用から新 DBMS 用に変換する必要がある。それぞれの変換の方法について以下に述べる。

データ：多くの DBMS がデータのエクスポート/インポート機能を備える。また、一部の DBMS においてはデータ移行ツールが提供されているため、変換は比較的容易である。

プロトコル：多くの AP は、ODBC、JDBC 等の DB 接続層を介して DBMS に接続している。DB 接続層は、AP に標準的な接続インタフェースを提供し、AP が標準インタフェース対応の標準プロトコルで発行した SQL を、DBMS 固有のプロトコルに変換する。プロトコル変換機能はライブラリとして入れ替えられるので、変換は容易である。

SQL：ISO SQL99 等、標準仕様が規定されているため、理想的には変換を行わなくても問題ない。しかし、実際には DBMS 独自に拡張された SQL が使われることが多く、異なる DBMS では解釈できない。そのため、SQL の変換も必要となるが、SQL の変換は容易ではない。AP が発行する SQL は、一般に AP 内に散在してハードコーディングされたロジックにより生成されるため、ソースコードレベルで、それら生成ロジックを適切に書き換える必要があるからである。

以上のように、DBMS 置き換えでは、DBMS 独自 SQL の変換が必要だが、実現するのが困難であることが問題となる。それだけでなく、変換した SQL が正しく動作しているか検証できないという問題もある。なぜなら、変換した SQL を投入した新 DBMS が、エラーを起こさず応答を返しても、それが旧 DBMS と同じ応答であるか、実運用環境において検証することができないからである。

3. 方式提案

以上の問題を鑑み、我々は DB 接続層における DBMS アクセス制御技術[1]を応用した、SQL 変換方式、および SQL 検証方式を提案する。

3.1. SQL 変換方式

新 DBMS 用 SQL を発行するように AP を変更するのではなく、図 1 に示すように、AP が発行した旧 DBMS 用の SQL を、途中経路である DB 接続層において、新 DBMS 用 SQL に変換する。従来は DB 接続層で、プロトコル変換ライブラリを用いて、プロトコルのみを DBMS 固有のプロトコルに変換していたが、その手前でデータ処理要求をフックし、SQL を DBMS 固有の SQL に変換するように DB 接続層を拡張する。変換処理そのものは、特定の SQL 文字列を置換する変換ルールとして定義する。変換ルールのパターンとしては以下の３パターンが考えられる。

予約語変換：DBMS 固有の予約語を変換するパターン。単純にその文字列を置換する。例えば、ある DBMS 独自のデータ型である「LONG」という予約語を、他の DBMS で同様の意味を持つ予約語である「TEXT」に変換する。

ロジック変換：予約語だけなく、パラメータを含むロジックを変換するパターン。例えば、ある DBMS 独自の拡張関数 DECODE を用いた

Database Replacement Method by SQL Conversion in Database Connection Layer.
Kenichiro FUJIYAMA, Koji KIDA, and Nobutatsu NAKAMURA.
Internet Systems Research Laboratories, NEC Corporation.

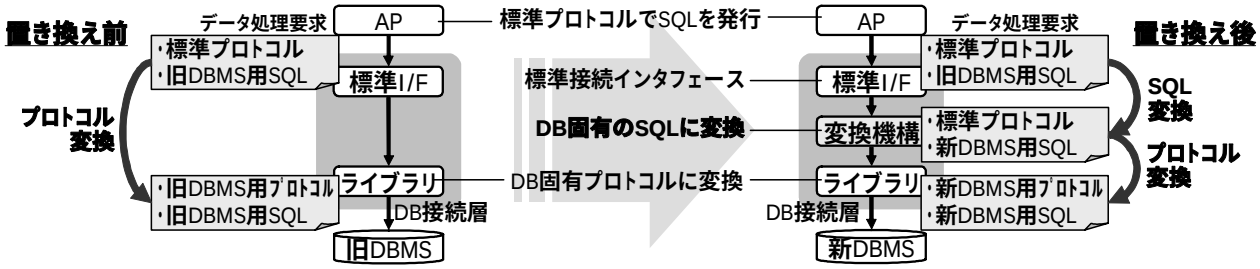

図 1：提案方式

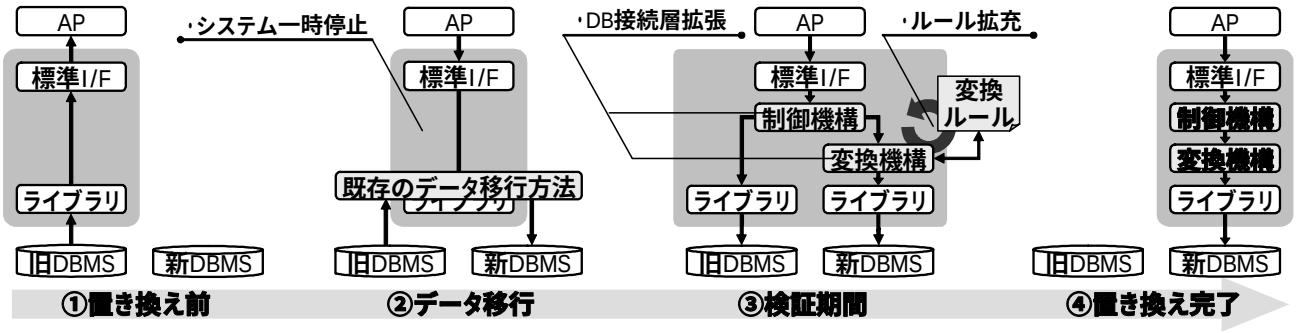

図 2：DBMS 置き換え手順

「 SELECT DECODE(key,val,res1,res2) label FROM table」という SQL を、標準関数 CASE を用いた「SELECT CASE WHEN key=val THEN res1 ELSE res2 END label FROM table」に変換する。

プリペアド型変換：事前に定義したプリペアドオブジェクトを呼び出す SQL を変換するパターン。この場合、SQL 変換だけでなく、呼び出されるプリペアドオブジェクトも、別途、事前に変換して登録しておく必要がある。

3.2. SQL 検証方式

変換後の SQL を新 DBMS に投入して正常に動作するかを検証するためには、旧 DBMS の SQL 実行結果を正解とし、これと新 DBMS の変換後 SQL 実行結果を比較すればよい。そこで、SQL 変換の前に、さらに SQL を多重化できるように DB 接続層を拡張する。多重化した SQL は、一方をそのまま旧 DBMS に投入し、もう一方の SQL を変換した後、新 DBMS に投入し、双方の処理結果を比較することで検証を行う。ただし、常に旧 DBMS を併用していては DBMS 置き換えとならないため、検証を含めた置き換えの手順は図 2 のようになる。

① 置き換え前：旧 DBMS で動作している状態。

② データ移行：DBMS 内のデータを移行。

③ 検証期間：SQL 変換、検証方式を導入し、旧 DBMS と新 DBMS を併用して、SQL 変換に問題が無いか検証。問題があれば変換ルールを修正。

④ 置き換え完了：検証期間を経て、安定性を確認した後、旧 DBMS を切り離す。

なお、提案方式は、AP、DBMS 両方から独立した DB 接続層を拡張しているため、AP を変更することなく透過的に、かつ、DBMS の種類に依存せず汎用的に適用することが可能である。

4. 考察

本方式では、SQL を変換する変換ルールを、如何に適切に設定するかが重要となる。しかし、最初から完全な変換ルールを設定することは困難である。そこで、基本的な変換ルールをベースに、検証期間中に検証結果を元にルールを修正、拡充していくという手順を検討している。さらに、検証済みの変換ルールを収集し、変換ナレッジとして他のシステムに活用することも検討している。例えば、汎用的に利用できる変換ルールセットや、変換ルール設定の指針となるガイドライン、設定候補をユーザに提示することで設定作業を支援するツール等である。以上のように、変換ナレッジを活用することで、変換ルールを効率的に設定できると考える。

5. まとめ

本稿では、DB 接続層を拡張することで、AP の発行する SQL を制御し、AP を変更することなく DBMS の置き換えを支援する方式について述べた。今後は、提案方式を実装、実システムに適用し、変換負荷等の評価を行う予定である。

参考文献

[1] 藤山他,”データベース同期複製のための Java API の拡張”, 情処 67 全大, 2005.