D-036

データ仮想化システムにおけるマルチデータベース間パーティショニングの一手法
A Partitioning Method for Multidatabase in a Data Virtualization System

齋藤 和広† 渡辺 泰之† 村松 茂樹† 小林 亜令†
Kazuhiro Saito† Yasuyuki Watanabe† Shigeki Muramatsu† Arei Kobayashi†

1. はじめに

管理対象となるデータを日付等で分割（パーティショニング）し，利用頻度の高い小規模データをインメモリデータベースシステム等の高速なデータベースシステム（DB）に，利用頻度の低い大規模データを分散 DB 等の大容量な DB に保存することで，安価に高速利用することが可能となる．

ここでデータ仮想化システム[5]を利用することで，このように複数の DB にパーティショニングされたデータを，仮想的に単一のデータとして利用することができる．しかし，データ仮想化システムは対象のデータのスキーマ情報を持つだけであり，実際のデータの中身を把握していない．そのため，クエリの処理対象データが特定の DB にしか存在しない場合でも，データ仮想化システムはパーティショニングされたデータを持つ全ての DB に対してクエリを投稿し，低速な DB に性能が依存する課題がある．

そこで筆者らは，特徴の異なる複数の DB にパーティショニングされたデータを，データ仮想化システムから効率的にクエリ処理するためのモデリング方式を提案する．本稿では，このモデリング方式の概要と，これを実現するための実装方式について述べ，Apache Hive[1] 及び PostgreSQL[2]を対象 DB として，データ仮想化システム JBoss Data Virtualization[3]（JDV）による評価結果を述べる．

2. 提案方式

2.1 概要

データ仮想化システムを利用したパーティショニングの実現方式の概要図とモデリングを図 1 に示す．クライアントであるアドホックに利用するユーザや定常クエリを実行するアプリケーション等は，複数ある DB にデータ仮想化システムを介してアクセスする．ここでは，大容量 DB と高速 DB の二つの DB が接続されていて，同じスキーマのデータをそれぞれ保持し，X という範囲のデータを大容量 DB が，Y という範囲のデータを高速 DB が保持していることを表す．データ仮想化システム上には，それぞれの DB が持つデータのスキーマと接続情報を表す物理モデルが定義されていて，仮想スキーマはこれらの物理モデルを結合処理（Union）した結果が一つのテーブルとなるモデルを表現している．

また仮想スキーマには，パーティショニング対象のデータの分割条件 X と Y がそれぞれどの DB に配置されているかの情報が紐付いている．これによってデータ仮想化システムは分割されたデータの配置を知ることができ，効率的なクエリ処理が可能となる．クライアントからクエリが投稿された際に，パーティショニング情報とクエリの内容から，どの物理モデルにクエリを投稿するかを判断し，更に

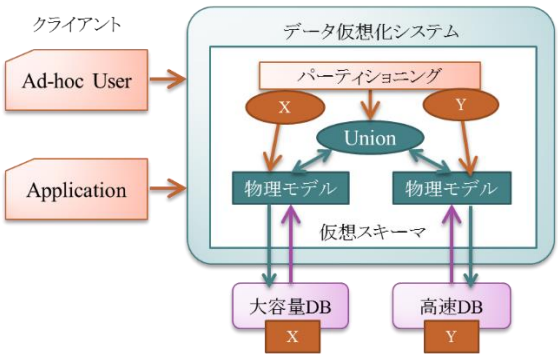

図 1 データ仮想化システムにおけるマルチデータベース間のパーティショニング例

結合処理の必要性を判断する．その後，各 DB へ投稿するクエリを生成する．

2.2 実装

データ仮想化システムを利用したパーティショニングを実現する上で必要となる要素は，パーティショニング情報の定義と，パーティショニング情報を考慮してユーザクエリを解釈し，実行する物理モデルを選択する機能である．

複数の DB へのパーティショニングに必要となるパーティショニング情報の定義は以下のとおりである．

- 対象の仮想スキーマ
- 対象となる物理モデルとテーブル
- パーティショニング方法（水平／垂直）
- パーティショニング条件の対象属性
- パーティショニング条件式（範囲／リスト／ハッシュ関数）

実行する物理モデルの選択は，ユーザクエリを解釈し，上記のパーティショニング情報の定義と照らし合わせることで行われる．パーティショニングを利用している仮想スキーマへユーザクエリが投稿されると，データ仮想化システムはユーザクエリを解釈し，対象の仮想スキーマのパーティショニング情報を抽出する．次にユーザクエリ内に含まれる WHERE 句等の条件式を解釈し，処理対象となる属性や条件式と，抽出したパーティショニング情報を照らし合わせる．これらが部分一致したパーティショニング情報の物理モデルを，投稿対象の物理モデルとして選択する．

3. 評価

3.1 評価実験

2 つの DB に対する仮想スキーマをデータ仮想化システム上にモデリングし，パーティショニングを実現する提案方式を適用する．そして一つの DB のみの実行となる範囲のユーザクエリを投稿し，提案方式を用いない場合との実

† 株式会社 KDDI 研究所，KDDI R&D Laboratories, Inc.

表 1　ソフトウェアバージョン

OS	CentOS release 6.3 (Final)
Java	jdk 1.7.0_51
JDV	6.0.0.GA-redhat-4
Hadoop	2.3.0-cdh5.0.0 (MapReduce1)
Hive	0.12.0-cdh5.0.0
PostgreSQL	8.4.20

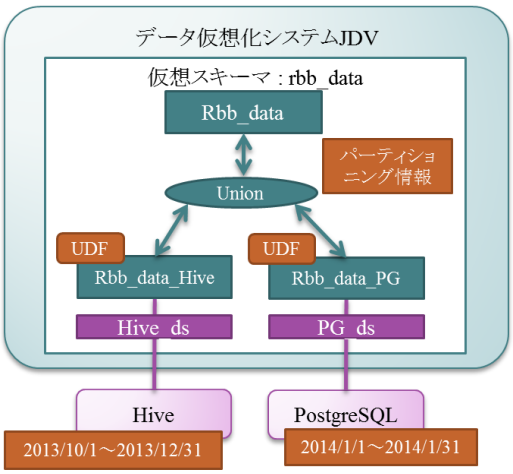

図 2　JDV 上のモデリング

行時間を比較することで，提案方式の有効性を評価する．

2 つの DB には，Apache Hive[1]と PostgreSQL[2]を用い，データ仮想化システムには JBoss Data Virtualization[3]（JDV）を利用した．利用したソフトウェアのバージョン情報を表 1 に示す．システム構成は，JDV 及び PostgreSQL 用サーバが 1 台，Hive サーバが 1 台，Hadoop クラスタが 4 台（Apache Hadoop の管理ノード 1 台，処理ノード 3 台）である．

評価に用いたデータはモバイル通信速度の計測サービスを提供している RBB TODAY SPEED TEST[4]のスマートフォン版計測ログである．これを計測時刻に基づいて分割し，2013/10/1～2013/12/31 の 3 ヶ月分の計測データを Hive に挿入し，2014/1/1～2014/1/30 の 1 ヶ月分の計測データを PostgreSQL に挿入した．Hive にあるデータの容量は 239MB で，PostgreSQL にあるデータは 61MB である．

提案方式を JDV 上でモデリングした構成が図 2 である．各 DB の物理モデル（Hive_ds, PG_ds）に対して，1 対 1 で繋がるビューモデル（Rbb_data_Hive，Rbb_data_PG）は，ユーザ定義関数（UDF）によってユーザクエリを投稿するかどうかを判断する．JDV ではビューモデルを SQL で表現し，WHERE 句に明示的に FALSE を組み込むことで対象の DB にクエリを投稿しない仕様となっている．これを利用し，UDF はパーティション情報を基にユーザクエリを解析して，対象外のデータソースの場合は FALSE を返す．クライアントに対してはこれら二つのビューモデルを結合（UNION）したビューモデル rbb_data を仮想スキーマ（テーブル）として提供する．

3.2 評価結果

実装した UDF とパーティション情報を利用した提案方式と，利用しない従来方式における実行時間を比較した．本評価

```
SELECT version_cli, COUNT(version_cli)
FROM rbb_data GROUP BY version_cli LIMIT 1
```

図 3　クエリ all

```
SELECT version_cli, COUNT(version_cli)
FROM rbb_table WHERE date > '2014/1/1'
GROUP BY version_cli LIMIT 1
```

図 4　クエリ 2014/1

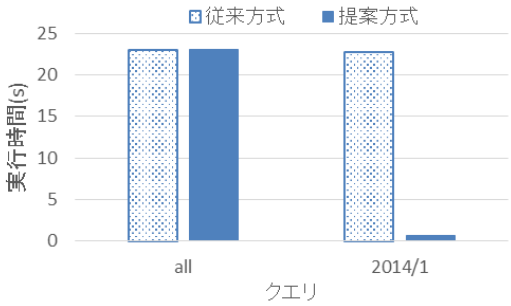

図 5　2 種のクエリの実行結果

で利用したクエリは図 3 の全てのデータに対するクエリ（all）と，図 4 の PostgreSQL のみが持つデータへのクエリ（2014/1）の 2 つである．これらのクエリの実行時間を計測した．結果を図 5 に示す．all クエリはどちらの方式も Hive と PostgreSQL 両方にクエリを投稿することから，ほぼ同じ結果となっている．一方で，従来方式の 2014/1 クエリは all クエリとほぼ同じ結果となっているが，提案方式の 2014/1 クエリは非常に短時間で完了していることがわかる．これは従来方式が PostgreSQL だけでなく Hive にもクエリを投稿しており，提案方式が PostgreSQL にのみクエリを投稿していることを示している．このことから，提案方式を用いることで，データ仮想化システムを用いて適切な DB にのみクエリを投稿できると言える．

4. おわりに

本稿では，特徴の異なる複数の DB にパーティショニングされたデータを，データ仮想化システムから効率的にクエリ処理するためのモデリング方式を提案し，これに必要な UDF を実装した．更に，2 つの DB とデータ仮想化システムを利用して実際に適切な DB へとクエリを投稿することを確認した．今後，提案方式を含む複数の仮想スキーマを組み合わせたクエリにおけるクエリ最適化処理と，データ挿入及びデータ移動を含めたパーティショニング範囲の自動最適化について検討したい．

参考文献

[1] Apache Hive, http://hive.apache.org/
[2] PostgreSQL, http://www.postgresql.org/
[3] JBoss Data Virtualization, http://www.jboss.org/products/datavirt.html
[4] RBB TODAY SPEED TEST, http://speed.rbbtoday.com/
[5] Rick F. van der Lans, “Data Virtualization for Business Intelligence Systems”, Morgan Kaufmann, 2012