PostgreSQL を活用した仮想データ統合基盤の実現

Realization of Virtual Data Integration Platform using PostgreSQL

中 山　陽 太 郎

要　約　企業においてデータ統合に対する要求が高まっている一方で，既存業務システムの一部のデータベースは存続させたい場合や，段階的なシステム移行にともないデータを部分的に統合したい場合など，全面的なデータの物理統合では解決できないケースが生じている．既存データベースの統合と既存データベースを継続して運用したいという要求に対して，物理的なデータ統合によらず仮想データ統合を適用することで，統合 DB と既存 DB の共存を実現することが可能となる．

2010 年 12 月に構築した仮想データ統合基盤では，対象とする複数の既存データベースを変更せず仮想的に統合し，一元的なデータベースにすることが可能である．仮想データ統合基盤として，オープンソースの RDBMS である PostgreSQL の外部データ管理機能を用いて PostgreSQL の連邦型データベースシステム（PostgreSQL Federation Database System: PGFDBS）を開発した．また，分散問合せの最適化を改善することで，Oracle，SQL Server を論理的に統合し，異種 RDB に対する問合せが効率的に実行されることを確認した．本稿では，連邦データベースシステム PGFDBS の実証検証について報告する．

Abstract　While the demand on data integration has increased in companies, the cases that the physical integration of full range of company data cannot resolve occur. Those cases include demands to keep some databases of existing business system alive, and to integrate data partially through gradual system migration. Towards a request both of the integration of existing databases, and continuing use of existing databases, the application of virtual data integration regardless of physical data integration enables realization of the coexistence of integrated DB and existing DB.

The virtual data integration, which we implemented as experimental proof in December 2010, enabled to build the integrated database by virtual consolidation without any changes on target databases for integration. As the virtual data integration platform, we developed the PostgreSQL Federated Database System (PGFDBS) using open source RDBMS PostgreSQL. By improving the optimization of distributed query, we confirmed that the query to heterogeneous RDBs was performed efficiently integrating of Oracle and Microsoft SQL Server logically.

In this report, we describe the implementation of proof of concept about the federated database system PGFDBS.

1. は　じ　め　に

拠点や部門を越えた情報活用が求められる中で，異なるシステムに散在するデータを必要なときにリアルタイムに活用したいという要求が高まっている．しかしながら，業務や部門ごとにデータベースが運用されており，業務統合や企業統合において異なるシステムのデータを簡単に統合できないという課題がある．既存のデータベースを統合してシステムを更改する場合

であっても，既存データベースを業務や部門ごとに運用したり，業務データベースに跨ってデータを参照するなど，既存データベースを維持しつつ新たな統合データベースと共存させたいという要求に対する解決が望まれている．

異種データベースを統合して新たなデータベースを構築し，同時に統合元の既存データベースは変更することなく継続して利用するためには，異種データベースや異なるプラットフォームに対してデータを透過的に統合連結するための仕組みが必要である．異なるデータベースを統合するための方法として，物理的にデータを集約し統合する ETL や DWH があるが，遅延のないリアルタイムなデータ統合連携の技術として，仮想データ統合の技術がある．仮想データ統合では，異種データベースを仮想的にリアルタイムで統合し，あたかも実体のある一つのデータベースとして利用することを可能とする．

2010 年 10 月から 12 月にかけて，異種データベースを仮想的に統合する仮想データ統合基盤を実装し，その有効性を検証した．仮想データベースを実現するアーキテクチャとして連邦型データベースシステム[1]を採用し，統合対象とするデータベースの自律性を保障しつつ，場所やプラットフォームの違いを吸収した．連邦型データベースシステムの実装には，オープンソース（OSS）のデータベースである PostgreSQL[2]を用い，PostgreSQL Federated Database System（PGFDBS）の研究と開発を行った[3]．

以下，2 章ではデータ統合における仮想データ統合の概略について述べる．3 章と 4 章では，PostgreSQL による連邦データベースシステム（PGFDBS）のアーキテクチャと機能について説明し，分散問合せにおける効率改善の実装の考慮について解説する．5 章でデータ統合における連邦型データベースの現状と動向について概説し，6 章で全体のまとめを述べる．

2. 連邦型データベースシステムによる仮想データ統合

連邦型データベースシステム PGFDBS では，自律分散する外部の異種データベースをあたかも PGFDBS のデータベースとして透過的に参照することを可能とする．データベース利用者は，仮想的に定義された一元的な仮想スキーマ[*1]を通して，外部のデータベースを意識することなくアクセスすることが可能となる．仮想データ統合では，物理的なデータ統合を行わないため，データベース構築やハードウェアに掛かるコストを削減できる．また，既存のデータベース資産を維持しながら，新たなビジネス要求に対応可能な柔軟なデータ利用という要求に応える．本章では，PGFDBS の特徴である透過性と，分散問合せの課題について述べる．

2.1 仮想データ統合の特徴

連邦型データベースシステム PGFDBS で構築する仮想的な統合データベースは，統合対象とする異種データベースに対して，データアクセスの一元化を可能とするため，次の三つの透過性を実現している．

- データ分散に対する透過性
  データベース・サーバの位置透過性とアクセス透過性，および統合されたデータベースの構成情報からの透過性を保障する．
- プラットフォームに対する透過性
  統合対象とする DB システムのハードウェアやオペレーティングシステムの違い，文字コードの違いなどを吸収し，ユーザに意識させない．

- データソースの異種性に対する透過性

  DB システムの違い，データモデルの違いを吸収し，SQL によるデータアクセスの透過性を保障する．

図 1 に PGFDBS の全体のイメージを示す．統合の対象とする既存のデータベース群は自律的に独立して稼働するものであり，PGFDBS を通して，それらの全てまたは一部のデータに対して透過的にアクセスすることを可能とする．

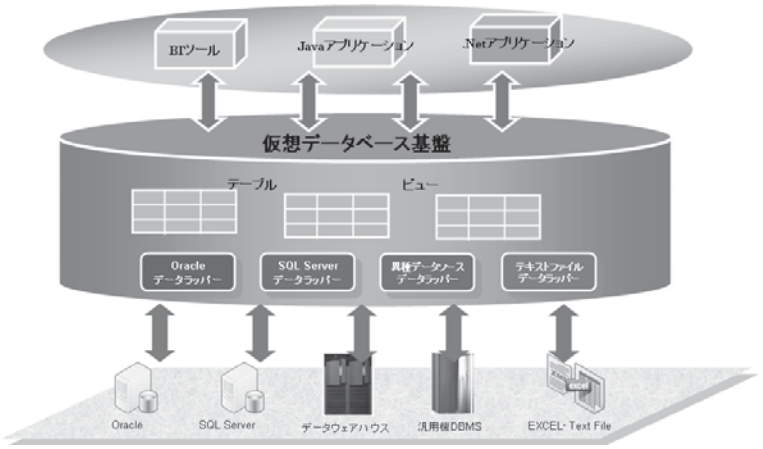

図 1　連邦型データベース PGFDBS の全体イメージ

2. 2　分散問合せ処理における効率の課題

一般に分散データベースにおける分散問合せでは，データ通信を効率化することが重要であり，分散問合せの最適化の問題として研究されている[4]．データ通信の処理コストは，問合せ結果のデータ量や問合せで発生するサーバ間の通信回数に比例して増加するため，検索条件によるフィルタリングや必要なカラムの選択によるレコードサイズの削減，及び結合演算において適切な結合アルゴリズムを適用することによって処理の繰り返し回数を削減することが重要となる．特に表の結合演算は，データ通信が増加する大きな要因となるため，問合せの結果集合を小さくすることや，結合アルゴリズムによりサーバ間の通信回数を低減することが求められる．これらは分散問合せにおける実行計画の最適化に関連する技術である．

分散クエリの最適化としては，問合せの選択条件によるフィルター処理を外部の RDB 側に実行させる条件プッシュダウン（Condition Pushdown），外部テーブルの結合処理におけるテーブル結合順序の最適化，また結合処理を外部 DB サーバ側に行わせる副問合せプッシュダウン（Sub Query Pushdown）などの方法により，外部 RDB と連邦サーバ間のデータ通信のオーバーヘッドをどれだけ抑えられるかが重要な課題となる．今回の実装では，PostgreSQL の外部データラッパー（Foreign Data Wrapper: FDW）機能[5]を利用した．

3.　連邦型データベースシステムの実装

本章では，連邦型データベースシステムのアーキテクチャと分散問合せについて述べる．

3.1　アーキテクチャと基本機能

PGFDBS による連邦型データベース・アーキテクチャは，PGFDBS 統合管理サーバと，そこからアクセスされる外部 DB サーバで構成される（図 2）．今回の検証では，外部 DB サーバとして Oracle, Microsoft SQL Server を各一台ずつとしているが，各複数台の外部 DB サーバを追加することが可能である．

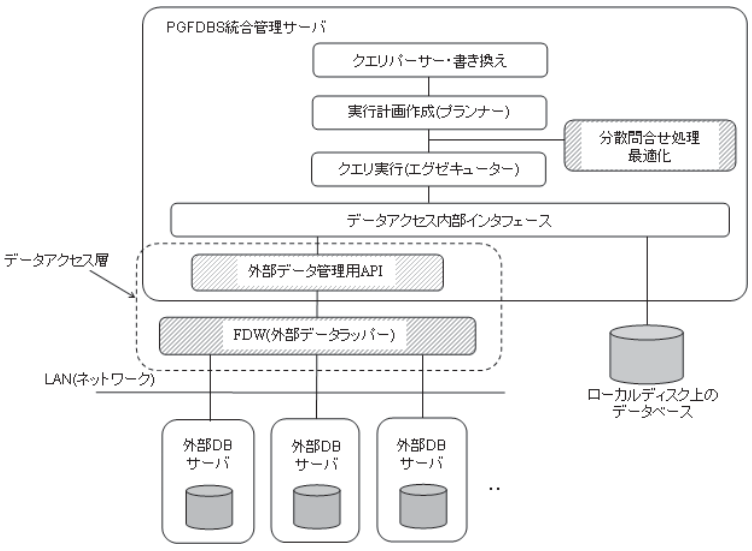

図 2　PGFDBS のアーキテクチャ

3.1.1　異種 DB サーバに対する透過性

外部 DB サーバの RDB で定義されているテーブルは，外部テーブルとして PGFDBS 統合管理サーバである PostgreSQL に定義することで，利用者は外部 DB サーバの異種性を意識することなく，あたかも PostgreSQL 自体のテーブルとして参照することが可能である．外部テーブルの定義のため，PostgreSQL の機能である SQL/MED（Management of External Data：外部データ管理）[6]によるデータ定義を使用する．

3.1.2　SQL/MED による外部テーブルの定義

外部のデータソースにアクセスするため，PostgreSQL では，標準 SQL の SQL/MED に一部準拠する外部データソースを定義するための構文が提供されている．具体的には，CREATE FOREIGN TABLE 文によって，外部 DB サーバ上の参照するテーブルを定義する．DDL 構文では，CREATE FOREIGN TABLE のサーバパラメタに，CREATE SERVER 文で定義したサーバ名を指定する．外部テーブルの定義に関連する外部データラッパーの DDL 構文を表 1 に示す．

利用者は，PostgreSQL で定義された外部テーブルに対して，通常のテーブルと同様に SQL による問合せを実行する．外部テーブルに対する SQL は検索のみであり，更新 SQL については対応していない．

表 1　PostgreSQL の SQL/MED 対応 DDL

```
CREATE FOREIGN DATA WRAPPER name
    [ HANDLER handler_function | NO HANDLER ]
    [ VALIDATOR validator_function | NO VALIDATOR ]
    [ OPTIONS ( option 'value' [, ... ] ) ]

CREATE SERVER server_name [ TYPE 'server_type' ] [ VERSION 'server_version' ]
    FOREIGN DATA WRAPPER fdw_name
    [ OPTIONS ( option 'value' [, ... ] ) ]

CREATE USER MAPPING FOR { user_name | USER | CURRENT_USER | PUBLIC }
    SERVER server_name
    [ OPTIONS ( option 'value' [ , ... ] ) ]

CREATE FOREIGN TABLE [ IF NOT EXISTS ] table_name ( [
  { column_name data_type [ NULL | NOT NULL ] } [, ... ] ] )  SERVER server_name
    [ OPTIONS ( option 'value' [, ... ] ) ]
```

3. 1. 3　対象データソース

SQL/MED では，外部テーブルの対象データソースとして，RDBMS に限定せず表形式として参照可能なデータソース一般を対象とすることが可能である．例えば，CSV ファイルも対象とすることができる．

3. 1. 4　PGFDBS 統合管理サーバ拡張

PGFDBS 統合管理サーバは，PostgreSQL を本体として，次の二つの機能対応から成る．一つはデータアクセス層であり，もう一つは分散問合せ最適化機構である．

（1）データアクセス層

ここでは，外部の異種 RDB に対するコネクション管理と，異種 RDB に対応した SQL 生成を行う．PostgreSQL から外部 DB へのアクセスには，標準 SQL を用いる．但しベンダーによっては，標準 SQL に準拠しない拡張 SQL 構文もあり，ベンダーの違いに対応した SQL の生成が必要である．外部の異種 RDB へのアクセスインタフェースは ODBC を用いた．PostgreSQL 内部では，SQL はパース木の形式で保持されているため，パース木と実行計画から分散問合せ用の SQL を生成する．

（2）分散問合せ処理の最適化

分散問合せの高速化のため，SQL 文の WHERE 条件を外部の RDB 側に置く条件プッシュダウン（Condition Pushdown）と，外部表の結合処理を最適化する．また，PostgreSQL の既存の問合せ最適化の後に分散問合せのための最適化を追加する．これにより，既存の最適化処理を変更することなく，分散問合せの最適化が可能となる．分散最適化では，結合順序，結合アルゴリズム，結合処理のプッシュダウン（結合処理をサブクエリ化して外部の RDB 側で実行させる），集合演算のプッシュダウンなど，分散問合せ用の実行計画を生成する．

PostgreSQL では，標準 SQL 規格 SQL/MED に一部準拠する外部データラッパー（FDW）機能が提供されている．PostgreSQL　FDW 外部機能は，表形式としてアクセス可能な外部のデータソースに対して，SQL によるアクセスを行うための仕組みを提供している[*2]．

3. 2　分散問合せによる外部データベースへのアクセス

ここでは，PostgreSQL の外部データ管理機能の利用とデータ処理の概略について述べる．

ユーザから発行されたSQLクエリは，パーサーによって内部形式を解析され，SQL構文木（SQLパーサー木）に変換される．実行計画（プランナー）では，問合せ実行のためにアクセスメソッドを割り当て，コストベースの最適化処理として，統計情報（行数，行の平均長，索引の分散度数等の情報）によって実行計画のコストを計算し，最も効率のよい実行計画を決定し，最終的なプラン木（Plan Tree：実行計画木）を生成する（図3）．生成されたプラン木は，クエリ実行器（エグゼキュータ）によって実行される．今回の実装範囲では，外部RDBの統計情報を取得管理することができないため，外部RDBの統計情報については，いずれの外部表も同じ値となるように内部的に設定している．

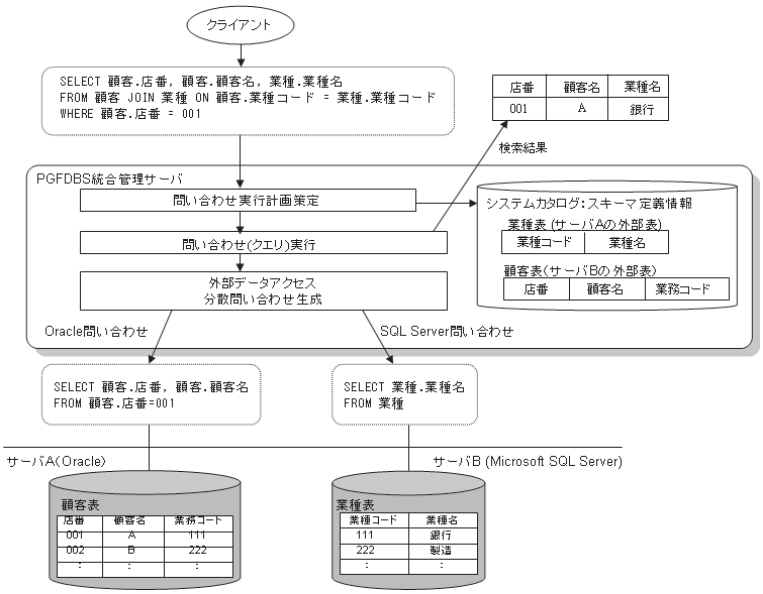

図3　PostgreSQL外部データ管理の構造

外部のRDBのテーブルからのデータは，PostgreSQLのレコード処理用の領域に格納され，PostgreSQL自体のテーブルに対するデータ処理と同じ関数で処理される．すなわち，PostgreSQLの既存のタプル（レコード）処理であるイテレータ（Iterator）アクセス方式によって処理される

3.3　分散問合せにおけるSQL変換

PostgreSQLから外部の異種データベースへの問合せは，SQLを生成しODBCドライバによるアクセスとした．PostgreSQLの内部では，エンドユーザからのSQLは内部形式としてパース木に変換されているため，外部DBに対してSQLを発行するためには，分散問合せとパース木から，異種DBに対するSQLの生成が必要である．これは逆パース処理と呼ばれ，de-parse関数によって実装される．また，ベンダーごとにSQLの違いがあるため，SQLの違いに対応した逆パース処理が必要である．以下に，クライアントが発行したSQLが，どのように変換されて外部のRDBに出されるかの例を示す．前提として，テーブルft1は，カラムcol1，col2，col3，col4が定義されているものとする．

ユーザから実行された SQL が(1)のように SELECT の選択リストにカラムが一つだけ指定された場合でも，現状の PostgreSQL では，(2)のようにテーブルに定義された全てのカラムを指定した SELECT 文が生成される．(2)の実行の結果，外部の RDB から検索対象としないカラムも返されることになり，データ通信量が増加し負荷となる．

（1）select col1 from ft1;

（2）select col1, col2, col3, col4 from ft1;

このため，PGFDBS では，外部 DB サーバに対する SQL でも，ユーザから発行された SQL 文(1)と同じ選択リストのカラムだけを生成するように改善した．

3.4　分散問合せにおける関数の実行

関数の実行については，関数の特性を考慮して，外部 DB サーバ側に処理させるか（プッシュダウン），PGFDBS 統合管理サーバ上で処理するか決定する．関数の分類として，集合関数，RDB で提供される関数，そして使用者定義関数がある．集合関数の場合と集合関数でない場合とについて，実行計画の違いを述べる．

集合関数（SUM，MAX，MIN，AVG，COUNT）を外部 DB サーバにプッシュダウンしない場合，外部 DB サーバから全データを取得し，PGFDBS 統合管理サーバ上にて集約処理を行う必要があるが，外部 DB サーバにプッシュダウンして集合関数を外部 DB サーバで実行させることにより，受け取る通信データ量は集約実行後のデータとなり，データ通信量を大幅に削減できる．但し，今回の実装においては，集合関数のプッシュダウンの適用は，SQL で単一の外部テーブルを参照する場合のみ対応する．

集合関数以外の関数，例えば RDB で提供される関数や使用者定義関数の場合は，数値の演算などの“不変な関数”（immutable function）の場合と，time 関数のような環境で値が異なる“可変な関数”（non immutable function）の場合とで処理が異なる．time 関数のように値が不定な場合は，結果がプラットフォームに依存するため，サーバ間で結果が同期される保障がない．そのような“可変な関数”の場合，実行条件を統一するため，全て PGFDBS 統合管理サーバ側で実行することで結果を保証する．例えば，問合せ文“select * from 外部 DB サーバ where now (col2) > 10”では，now 関数の実行条件を統一するため，PGFDBS 統合管理サーバで実行する必要がある．

3.5　分散問合せ最適化処理の改善方法

分散問合せの最適化処理の拡張方針として，PostgreSQL における既存の最適化処理を変更せず，関数を追加することで，分散問合せ処理モジュールの独立性を高めることを目標とした．そのため，PostgreSQL の既存の最適化処理に影響を与えずに分散問合せの最適化を実現可能にする箇所を特定し，「分散問合せにおける 2 段階の最適化方式」として開発した（図 4）．

同一の外部 DB サーバ上の外部表の結合は，その外部 DB サーバにプッシュダウンすることによって，それぞれの外部表のデータを，PGFDBS 統合管理サーバ側で参照しながら結合処理することにより，データの転送に掛かる処理負荷が軽減される．

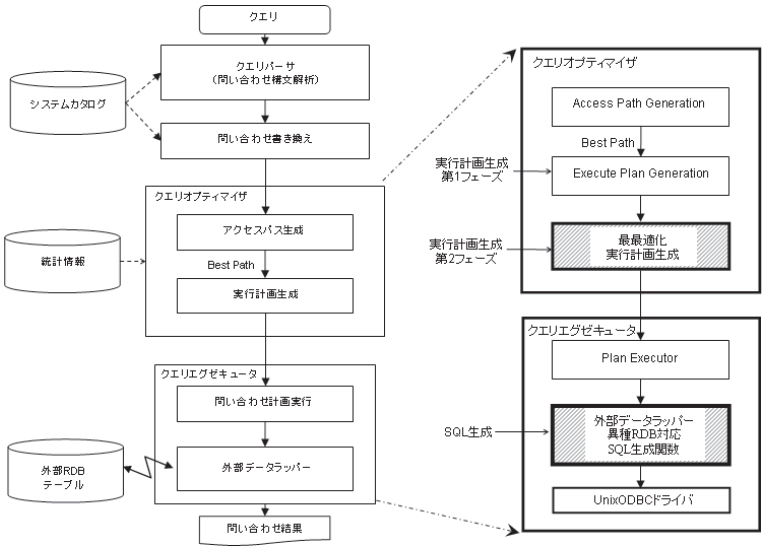

図 4　分散問合せにおける 2 段階の最適化

分散問合せにおける結合処理の最適化は次のとおりである．

1）　同一外部 DB サーバ上の 2 表の結合処理の最適化
同一サーバに属する外部テーブル同士の結合の場合，結合処理をサブクエリとしてプッシュダウンし，外部 DB サーバで結合処理を行い，その結果を PGFDBS 統合管理サーバ側で参照することにより，データ転送量を削減する．

2）　異なる外部 DB サーバに属する 2 表のハッシュ結合対応
異なる外部テーブル同士の結合は PGFDBS 統合管理サーバ側で行われるが，既存の PostgreSQL では入れ子ループのみが選択される．そのため，統計情報を設定し，ハッシュ結合が選択されるように拡張した[*3]．

最適化の実装においては，1)の適用を，外部テーブルに対するスキャン処理（ForeignScan）がリーフノードにある場合のみに限定している．また結合対象の一方の外部表の検索結果が，もう一方の外部表の選択条件となるような外部表同士で依存性のある結合の最適化については考慮しない．図 5 に分散問合せの結合処理の最適化の概略を示す．左側が最適化前の外部表の結合処理，右側が最適化後の結合処理イメージである．図中の FS は，外部テーブルに対するプランタイプとして追加された ForeignScan を示す．SQL として，表 1，表 2，表 3 の結合があり，表 2 と表 3 は，サーバ B 上に存在する．最適化を実施しない場合，表 2，表 3 は，それぞれ独立した外部 DB サーバとして扱われ，結合処理は PGFDBS 統合管理サーバ側で行われる．それに対して，分散問合せ処理の最適化を適用した場合，表 2，表 3 は，外部 DB サーバ B 上のテーブルとして，表 2 と表 3 を結合する一つの SQL が組み立てられ，サーバ B に対して実行されるように最適化される．これによって，表 2，表 3 の結合処理は，PGFDBS 統合管理サーバ側ではなく，外部 DB サーバ B で行われる．表 2，表 3 をそれぞれ PGFDBS 統合管理サーバ側から結合ループ処理によって繰り返しアクセスすることに比べ，外部 DB サーバ B で結合のみ実行した方が，接続とデータ転送の負荷を大幅に軽減し効率が向上する．

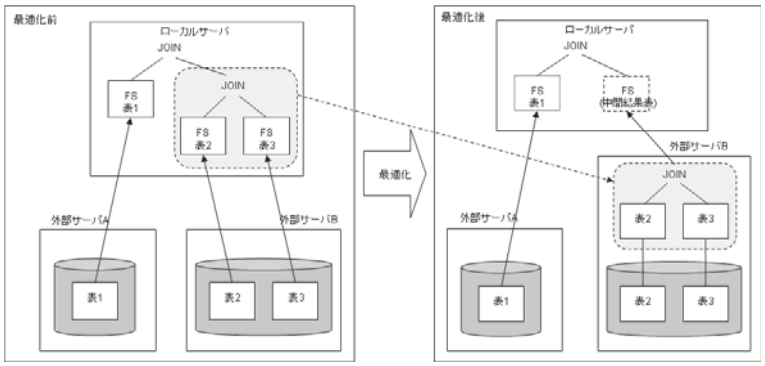

図 5　分散問合せの結合処理の最適化

4.　分散問合せ最適化改善の検証

本章では，外部 RDB へのデータアクセス，及び異種 RDB 用の SQL 生成，また分散問合せの最適化の仕組みと検証結果について解説する．

4. 1　検証環境

連邦型データベースに対して，統合対象とするデータベースとして，Oracle，Microsoft SQL Server，PostgreSQL を使用した．サーバの構成図を図 6 に示す．PGFDBS から Oracle，SQL Server へは，UnixODBC[*4] を用いてアクセスしている（図 7）．

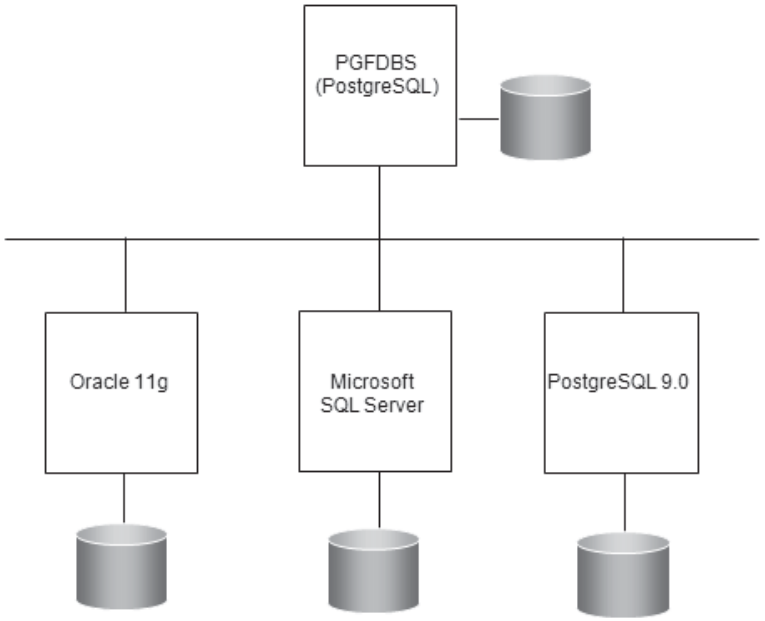

図 6　PGFDBS 統合管理サーバ環境

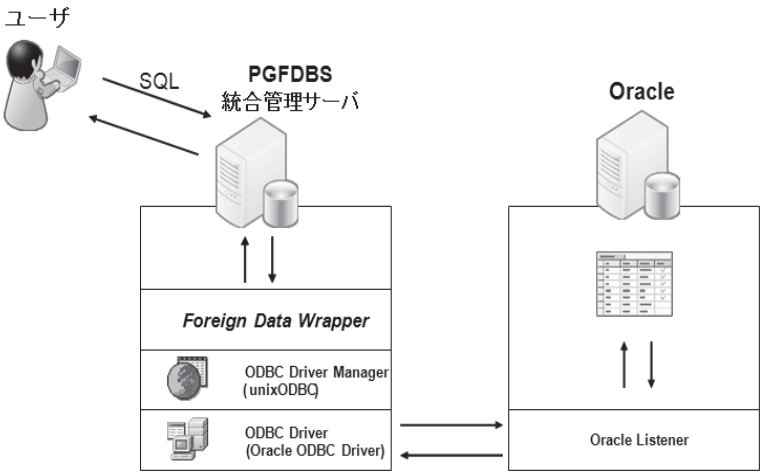

図 7　ODBC によるアクセス

4. 2　分散問合せ最適化処理改善の検証

2 段階の最適化方式の検証として，実際に問合せによる実行計画を生成し，SQL の実行計画が書き換えられることを確認した．現状では，既存の最適化によって生成された実行計画木を入力として，リーフノードに外部表の結合がある場合，外部表が同一のサーバに属するものであれば，外部 DB サーバに結合を行う一つの SQL として実行するように実行計画を書き換える．

4. 2. 1　結合処理の最適化

実行計画木（Execute Plan Tree）は，その木構造のパターンによって，左側が深い木（Left Depth Tree），右側が深い木（Right Depth Tree），ブッシュ型の木（Bushy Tree）がある．図 8 は，左側が深い木の実行計画木の，最適化前（図 8 左側）と最適化後（図 8 右側）の対比

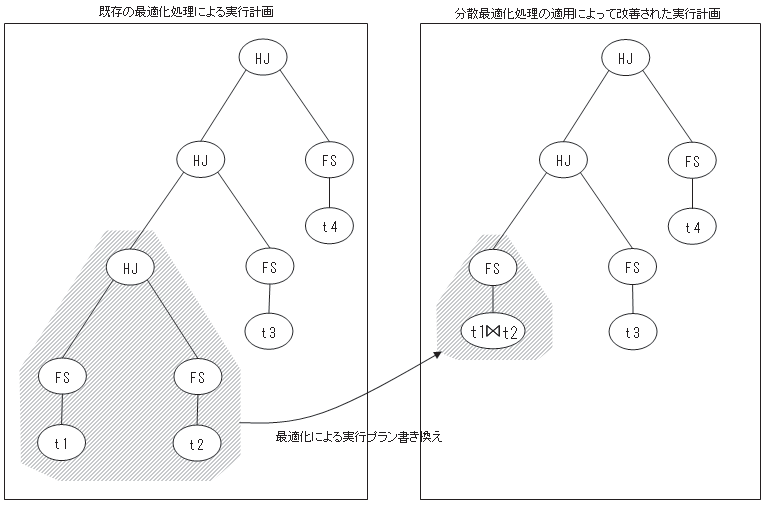

図 8　左側が深い木の最適化の例

である．同様に，右側が深い木，及びブッシュ型の木についても，同じデータベース上の 2 表の結合処理をまとめて一つの SQL としてプッシュダウンし，外部 DB サーバ側で実行するように最適化されることを確認した．このように，外部テーブルスキャン（FS）を一時的に一つにして，クエリを外部 DB にプッシュダウンすることにより，FS ごとの不要なデータ転送を削減することが可能となる．

4. 2. 2　集合関数の最適化

次の集合関数を含む SQL が実行される場合の最適化の例を図 9 に示す．

Select AVG(c1) from 外部表 where c2 > 10000 ;

通常の PGFDBS 統合管理サーバ側の実行では，集合関数は，実行計画として最後に評価される（図 9 左側）．外部表に対しては，外部 DB サーバ側で集合関数（この場合は AVG）のプッシュダウンを行う（図 9 右側）．外部 DB サーバで集合関数を実行させることによって，外部 DB サーバからのデータ転送回数を削減し，効率化を図ることができる．

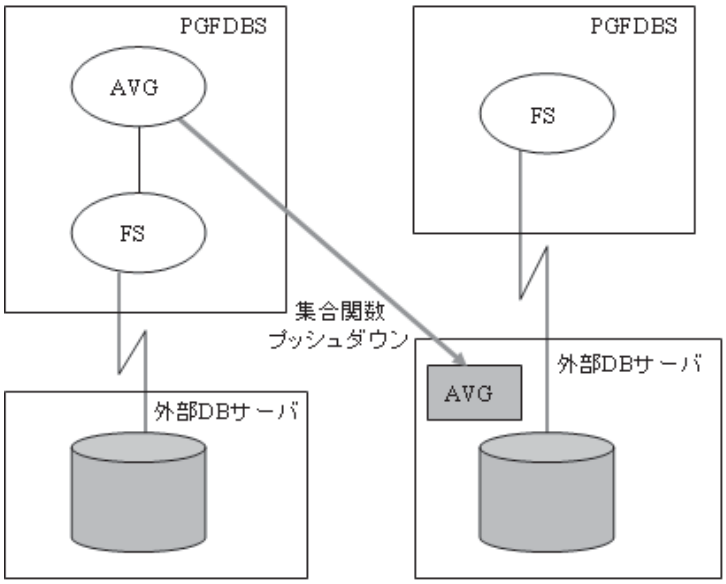

図 9　集合関数の最適化の例

5.　仮想データ統合の現状と今後

5. 1　EII としてのデータ連携と連邦型データベース

仮想的なデータ連携やデータ統合への要求は，アプリケーションにおける 3 層アーキテクチャの広がりとともに高まってきたと考えることができる．クライアントは，アプリケーション層を介してデータ層にアクセスすることで，データベース層の独立性が保障されるため，データベースを跨るデータアクセスの自由度が高まるとともに，データ層におけるデータ連携の重要性が認識されるようになってきた[7]．

仮想的なデータ連携を実現する方法としては，連邦型データベースの他にデータ交換（Data Exchange）による方法がある．データ交換では，データソースに応じたデータアクセスイン

タフェースを利用するが，異なるデータソースへのアクセスインタフェースをアプリケーション側に組み込むことが必要となる．連邦型データベースは，データ管理システムであり，問合せ言語である SQL によって複数のデータソースに一元的にアクセスする．単純なデータ参照はデータ交換が適しており，複雑なデータ操作，例えばデータの複合化や集約処理など異なるデータソースを連携させるような処理では連邦型データベースが適すると考えられる[8][9]．システム統合の視点から見た場合，アプリケーション統合（EAI: Enterprise Application Integration）ではデータ交換を用い，情報統合（EII: Enterprise Information Integration）におけるデータ統合では統一的なデータ処理と管理が可能である連邦型データベースが有効であると見ることもできる．また EAI におけるデータ共通化インタフェースとして，連邦型データベースをデータ統合共有基盤とするシステムも考えられる．最近では，DWH や BI による集約処理結果に対して，リアルタイムの基幹データを参照するための補完的な機能として連邦型データベースを用いる提案もされている．

5. 2　連邦型データベースの利用と技術課題

連邦型データベースにおける分散問合せは，実用的には成熟した段階と考えられるが，情報統合における異種データベース連携において連邦型データベース技術の適用は，普及している状況にはない．理由として，企業内のデータベースが同一ベンダーである場合が多く，同一ベンダーの DBMS で提供される同種データソース間のデータ連携機能で充分であること，また，連邦型データベース自体の課題として，大量データの集計やバッチ処理において実用的な性能を満たすことが難しいケースが多いことが考えられる．

連邦型データベースでは，少量のデータのリアルタイム・アクセスに有効であるが，大量データに対する集約処理ではデータ転送の負荷が重くなるため，いかに実効的な性能を担保できるかが問題となる．しかしながら，近年のハードウェア性能の向上，またネットワークの高速化によって，これまで課題であった大量データに対する分散問合せの性能の向上が期待される．

大量データを効率的に処理する方法として，例えば PGFDBS では，外部からのデータを PGFDBS 統合管理サーバのデータベース上のテーブルに保持することが可能であり，例えば，複雑な集計処理を事前に実行し，一時的な結果として保存することができる．このように，PGFDBS では，リアルタイムな問合せとバッチ処理的な集計処理とを使い分け，それぞれの結果を連携させることで，実用的な処理性能を満たすことも可能である．

6. お　わ　り　に

PostgreSQL を用いて連邦型データベースシステム PGFDBS を開発し，仮想データ統合基盤の構築を実証した．外部データベースとして Oracle，Microsoft SQL Server に対して，外部データラッパーを開発し，さらに分散問合せを最適化するための機構を組み込んだ．この機構は，既存のプランナーによって生成された実行計画を入力として，分散問合せを効率化するよう最適化していることが特長である．これによって，既存の最適化処理を変更せず，独自に拡張した分散最適化を適用することが可能である．実証検証では，分散問合せの最適化によって，分散する異種データベースを意識した効率的な実行計画が生成されることを確認した．

PGFDBS の外部表更新と分散トランザクションの機能拡張については今後の課題である．また，多様化するデータ統合の要求において，ETL や DWH を補完するデータ統合ソリュー

ションの位置づけとして展開するとともに，クラウド間の連携まで視野にいれた将来の連邦型データベースの課題について解決すべく検討を進めていきたい．

＊1　仮想スキーマとは，異種 RDB の外部テーブルを定義するスキーマを意味する．
＊2　PostgreSQL 外部データラッパー（FDW）の機能については，PostgreSQL のドキュメント（参考文献[5]）を参照のこと．
＊3　既存の PostgreSQL では，結合アルゴリズムとして，入れ子ループ結合，ハッシュ結合，マージソート結合がある．
＊4　オープンソースの ODBC ドライバ．UNIX や Linux 上で動作する ODBC ドライバを提供している．http://www.unixodbc.org/

参考文献

［1］ Amit P. Sheth, James A. Larson, “Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases”. ACM Computing Surveys, Vol.22, No.3, 1990.
［2］ PostgreSQL Global Development Group, http://www.postgresql.org/（2012 年 2 月確認）
［3］ Yotaro Nakayama, “Real Federation Database System leveraging PostgreSQL FDW”, PostgreSQL Conference PGCon2011, Canada, 2011.
［4］ Donald Kossmann, “The State of the Art in Distributed Query Processing”, ACM Computing Surveys, Vol.32, No.4, 2000.
［5］ PostgreSQL 9.1.2 Documentation, PostgreSQL Global Development Group, http://www.postgresql.org/docs/9.1/static/index.html（2012 年 2 月確認）
［6］ Information technology – Database languages – SQL – Part 9: Management of External Data (SQL/MED), ISO/IEC JTC 1/SC 32/WG 3, The United States of America (ANSI), 2006
［7］ Michael Stonebraker, “Too much middleware”, ACM SIGMOD Record, Vol.31, No.1, 2002.
［8］ Alon Y. Halevy, Anand Rajaraman, Joann J. Ordille: “Data Integration: The Teenage Years”, Proc. of VLDB’06, 2006.
［9］ Alon Y. Halevy, et al., “Enterprise information integration: successes, challenges and controversies”, ACM SIGMOD Conference, 2005.

執筆者紹介　中 山　陽 太 郎（Yotaro Nakayama）
1988 年日本ユニシス（株）入社．Unisys 汎用機データベース管理システム主管業務を経て，オープンソース RDB の評価・開発に従事．現在，総合技術研究所インキュベーションラボに所属し，データベース技術に関する調査研究を担当．