社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEEWRS

信学技報
TECHNICAL REPORT OF IEICE

分散データベース環境における複製データの仮想化による SQL 処理の最適化手法

大川 昌弘† 黒澤 亮二† 福田 剛志†

†日本アイ・ビー・エム（株）大和ソフトウェア開発研究所 〒242-8502 神奈川県大和市下鶴間 1623-14
E-mail: †{mohkawa, kurokuro, fukudat} @jp.ibm.com

あらまし 近年，多くの企業では，業務毎や部門毎に管理されたデータベースを個別に活用するだけでなく，これらの分散されたデータベースを統合し，データ分析などを行っている．データベースの統合手法としては，分散されたデータベースの表を仮想的に統合するフェデレーション技術や，統合サーバに分散されたデータベースの表を複製するレプリケーション技術などがある．これらの技術により，SQL 文を用いて分散された複数のデータベースに存在する表をジョインすることができる．しかしながら，フェデレーション技術では SQL のジョイン処理におけるネットワークのオーバーヘッドの問題が，レプリケーション技術では表の複製処理の負荷の問題や SQL のジョイン処理における統合サーバへの負荷集中の問題が考えられる．本論文では，これらの問題を解決するために，両者の技術を組み合わせた手法を提案する．提案する手法では，既存のフェデレーション技術を拡張し，仮想的な表と現実の表の１対多のマッピングを可能とすることで，レプリケーション技術と組み合わせて効率的で可用性のある SQL 処理を実現する．

キーワード 情報統合，分散 DB，異種 DB，複製，アクセスパス，問い合わせ処理

SQL processing optimization in virtualizing replicated data under distributed database environment

Masahiro OHKAWA† Ryoji KUROSAWA† and Takeshi FUKUDA†

†Yamato Software Laboratory, IBM Japan 1623-14 Shimotsuruma, Yamato, Kanagawa, 242-8502 Japan
E-mail: †{mohkawa, kurokuro, fukudat} @jp.ibm.com

Abstract In recent years, not only each department or each business operation uses its own database, but these databases are integrated and used for some applications such as data analysis. There are several technologies to integrate distributed databases. Federation technology is to virtually integrate tables on distributed databases. Replication technology is to replicate them. But federation technology may have network overhead while processing SQL. Replication technology may have processing overhead while replicating tables, and SQL processing overhead in integrated server. To solve these problems, this paper proposes a hybrid approach to use both technologies. For this approach, we extended federation technology to support 1-to-many mapping between virtual table and real table, then realize effective SQL processing with high availability.

Keyword Information Integration, Distributed Database, Heterogeneous Database, Replication, Access Path, Query Processing

1. まえがき

近年，多くの企業では，業務毎や部門毎に管理されたデータベースを個別に活用するだけでなく，これらの分散されたデータベースを統合し，データウェアハウスなどを構築してデータ分析を行うなど，情報の有効かつ効率的な利用を図り，企業の競争力を高めている．また企業の吸収・合併などにより，それぞれの企業が管理していたデータベースを統合する必要性もでてきている．

業務毎や部門毎に管理されたデータベースや，吸収・合併前のそれぞれの企業が所有するデータベースは異機種であるケースが多い．それらのデータベース

の統合方法としては，すべてのデータを 1 つのデータベースに移行し，それに合わせてシステムを刷新する方法があるが，業務アプリケーションは業務毎や部門毎に管理し続けたい場合や，既存のハードウェアやソフトウェアの資源を有効利用したい場合や，移行に伴う業務アプリケーションの再開発コストなどの問題からすぐに移行できない場合などがある．そのため，分散されたデータベースを仮想的に統合するフェデレーション技術や，分散されたデータベースの表を統合サーバに複製するレプリケーション技術などがよく用いられている．IBM では，前者を実現するために WebSphere Federation Server 製品[1]（以後，WFS と呼ぶ），後者を実現するために WebSphere Replication Server 製品[2]（以降，WRS と呼ぶ）を提供している．

WFS は IBM の RDBMS 製品である DB2[3]に組み込まれて動作し，さまざまな種類の分散されたデータの統合を可能とする．統合サーバとして DB2 のデータベースを利用し，そこに分散されたデータベース（リモートデータベース）の表（リモート表）を関連付けた仮想的な表（ニックネーム）が登録できる．SQL 文でニックネームを表のように使用することで，分散されたデータは，あたかも DB2 上に存在するかのように扱うことができる[4][5][6][7][8].

フェデレーション技術の問題点として，複数のデータベースに存在する表をジョインする場合が挙げられる．ジョインの演算を行う場合，その対象データが必要となるが，その対象データが別々のデータベースにある場合，演算に必要なデータをすべて統合サーバに集めて演算処理が行われるため，データを転送する時間がかかるなどの問題がある．

WRS は，あるデータベースの表（ソース表）を別のデータベースの表(ターゲット表)に複製する技術で，ソース表とターゲット表の同期処理が行われる．

レプリケーション技術の問題点として，ソース側とターゲット側の複製処理の負荷やソース・ターゲット間のネットワークの負荷，さらに，統合サーバに対する SQL 文の処理はすべて統合サーバで行われるため，負荷が集中することなどが挙げられる．

本論文では，これらの問題点を解決するために，フェデレーション技術とレプリケーション技術を組み合わせた手法を提案する．既存の WFS では，ニックネームとリモートデータベースの表を 1 対 1 で関連付けるが，これを 1 対多で関連付け，かつ，SQL 文の処理が最適になるような表を選択できるように拡張することで，レプリケーション技術と組み合わせた効率的なデータアクセスを実現する．

2. 従来技術とその問題点

2.1. フェデレーション技術とその問題点

2.1.1. フェデレーション技術の概要

ニックネームを参照する SQL 文が統合サーバに対して発行されると，統合サーバは通常のデータベースにおけるアクセスパスの最適化技術に加え，フェデレーション固有の分散最適化技術を用いて，SQL 文に含まれる各構文を統合サーバ上で実行するか，リモートデータベース上で実行するかを決定する．ここでは各分散データベースのマシンスペック（CPU 速度，ディスクアクセス速度）やネットワーク帯域の容量，また各分散データベースの言語特性（文字コード，照合順序）や SQL 処理の特性（実行できる関数の種類，データタイプの対応，NULL 値の扱い）などが勘案され，各分散データベースで処理する内容が決定される[9].

必要なリモート操作が決定したのち，統合サーバはそれを実行するためのリモートデータベース用の SQL 文を作成しリモートデータベースに処理要求する．INSET/UPDATE/DELETE 文の処理の場合は処理結果を，SELECT 文の処理の場合は処理結果に加え結果セットをリモートデータベースから受け取り，統合サーバ上で必要な処理を実行したのち，クライアントへ結果を返す．

図 1 の例では，2 つのリモートデータベースを統合し，リモート表 T1, T2, T3 をそれぞれニックネーム N1, N2, N3 に関連付けている．この構成で，N1, N2, N3 を参照する SQL 文が統合サーバに対して発行されると，統合サーバはリモートデータベースに対して最適な SQL 文を発行し，それらの結果から統合サーバに対して発行された SQL 文の結果セットを求める．

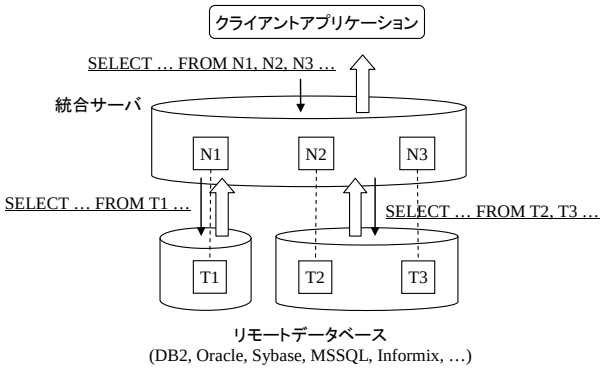

図 1. フェデレーション構成と SQL 処理例

アプリケーション開発者は統合サーバにアクセスするアプリケーションを作成することで，異機種混合のデータベース上のデータにアクセスできる．

2.1.2. フェデレーション技術の問題点

複数のリモートデータベースを統合した分散データベース環境で，別々のリモートデータベースにあるリモート表をジョインする SQL 文を処理する場合，必要なデータを取得するための SQL 文が各データベースに発行され，統合サーバ上でジョイン処理が行われる．一方，1 つのリモートデータベースにある複数のリモート表をジョインする場合，リモートデータベース上でジョインすることも可能となる．

一般的に，参照する表のレコード数は，ジョインした結果得られるレコード数よりはるかに多いため，統合サーバ上でジョインを実行する場合，より多くのデータをリモートデータベースから受信する必要があり，パフォーマンスが悪化する．

例えば図 2 のようにリモートデータベース 1 の表 T1 とリモートデータベース 2 の表 T2 にそれぞれ 1 万，１０万レコードがあり，それらの表のジョインの結果として 5000 レコード得られるとする．

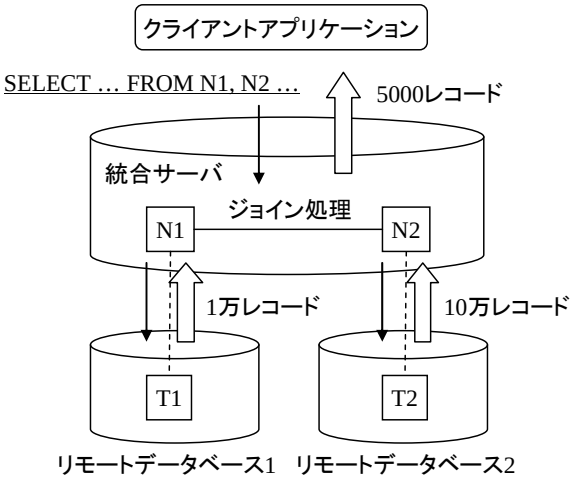

図 2. 2 つのリモートデータベースの表のジョイン

この場合，リモートデータベース 1 から 1 万レコード,リモートデータベース 2 から 10 万レコードを統合サーバに送信し，統合サーバでジョイン処理が行われる．また，統合サーバ上でのジョイン処理では，当然のことながら，リモートデータベースで作成されている索引などが使用できない．

図 3 のように，T1 と T2 が同じリモートデータベースにある場合には，リモートデータベースでジョインすることも可能となり，そちらの方がパフォーマンスが良いと見積られる場合，そのように処理される．その場合，索引などがあれば使用され効率よくジョイン処理が行われ，リモートデータベースからジョイン結果として 5000 レコードが統合サーバに送られる．

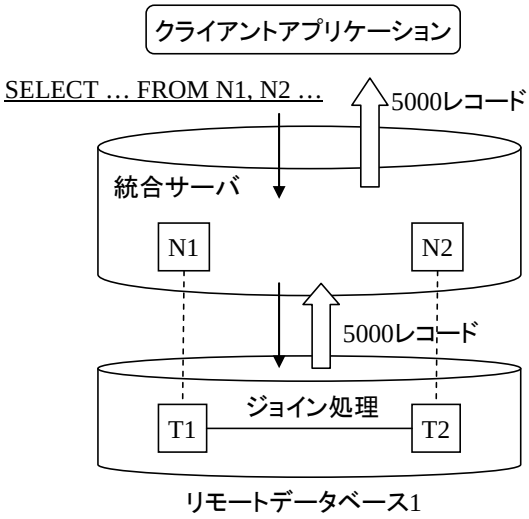

図 3. 1 つのリモートデータベースの表のジョイン

結果として，一般的に，図 3 の構成のパフォーマンスが図 2 と比べ，非常に良い結果となる．

2.2. レプリケーション技術とその問題点

2.2.1. レプリケーション技術の概要

レプリケーション技術には，ソース表からターゲット表に複製するほかに，複数の表の間で同期をとるピア・ツー・ピア型などがある．レプリケーション技術は分散されたデータを一箇所に集める目的，負荷を分散する目的，障害時の可用性を向上させる目的などに利用される．

複製するためには，ソース表の差分データを取得する必要があるが，その手法として，SQL 文のトリガーを使用する方法と，各データベースで独自に実装されているデータベース・リカバリーログ（トランザクションログとも呼ばれる）を使用する方法がある．前者は，SQL 文で差分データが取得できるため容易に実装できる反面，各業務のトランザクション処理の一環として行われるため，各業務にインパクトを与える．後者は，データベースに対して行われた変更が格納されたデータベース・リカバリーログを用いてソース表の差分データを取得するため，各業務のトランザクション処理に影響は与えないが，データベース製品毎に実装してゆく必要があるため，実装やその保守のコストがかかる．既存の WRS ではソース表が DB2 にある場合のみ，データベース・リカバリーログを使用してソース表の差分データを取得している．

ターゲット表への複製方法としては，ソース表の差分データを差分表に保持し，ターゲット・サーバから SQL 文で差分データを取得する方法（プル型）や，差

分データをメッセージキューなどでターゲット・サーバに送付する方法（プッシュ型）などがある．

2.2.2. レプリケーション技術の問題点

例として図 4 に示される構成を考える．図 4 では 2 つの業務システムがあり，1 つのシステムには表 T1 が，もう 1 つのシステムには表 T2 と T3 がある．これらを統合するために統合サーバを用意し，３つの表の複製を持つ．

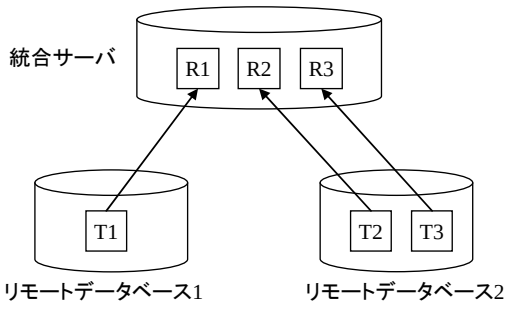

図 4. レプリケーション構成例

このようにレプリケーション技術を利用すると，すべての表が統合サーバに複製されるため，複製処理の負荷がかかる．また，統合サーバに対する SQL 文の処理は，すべて統合サーバで行われるため，SQL 文の処理の負荷が統合サーバにかかる．

この例では，2 つの業務アプリケーションを想定しており，各業務アプリケーションは，それぞれリモートデータベース 1 とリモートデータベース 2 にアクセスする．さらに，データ分析するためのアプリケーションが統合サーバに対してアクセスされる．

業務アプリケーションで使用される SQL 文はあるレコードの登録・更新・削除・参照といった比較的シンプルなものと考えられ，データ分析で使用される SQL 文は集約関数などを使用した処理に時間や負荷のかかる複雑なものと考えられる．したがって，業務アプリケーションになるべく影響を与えず，統合サーバに対する SQL 文の処理を効率的に行えるように，すべてのハードウェア資源を有効に利用する仕組みが望まれる．

3. 解決方法

２章で述べた問題を解決するために，フェデレーション技術とレプリケーション技術を組み合わせた構成を提案する．

例として図 4 と同様の業務システムを考える．データ分析として，T1 と T2 のジョインと T2 と T3 のジョインの 2 種類の SQL 文が必要であるとすると図 5 のように T2 をリモートデータベース 1 に複製し，T1, T2, T3 の各表のニックネームを統合サーバに作成する．T2 はリモートデータベース 1 とリモートデータベース 2 に存在し，それらを 1 つのニックネームで関連付ける．

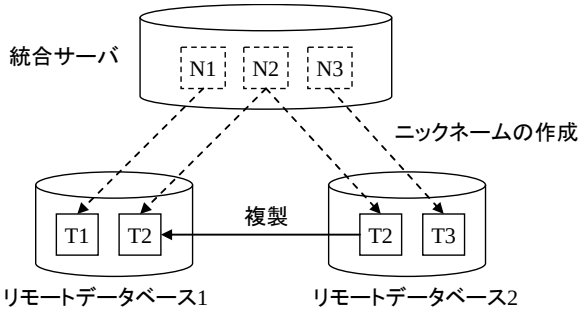

図 5. フェデレーション技術とレプリケーション技術の組み合わせ例 1

T1, T2, T3 のニックネームをそれぞれ N1, N2, N3 とすると，T1 と T2 のジョインは N1 と N2 を使用して行い，T2 と T3 のジョインは N2 と N3 を使用して行う．N2 は 2 つの T2 に関連付けられており，SQL 文が効率的に処理される方の T2 を使用する．一般的には，同じデータベース内でジョインした方がパフォーマンスが良いので，そのように仮定すると，N1 と N2 のジョインが要求された際にはリモートデータベース 1 の T1 と T2 が，N2 と N3 のジョインが要求された際にはリモートデータベース 2 の T2 と T3 が使用され，効率的にジョインの演算が行われる．結果として，SQL 文に応じてジョインの演算が行われるデータベースが分かれるといった負荷分散が行われ，かつ，複製も T2 の 1 つのみで済んでいる．

さらに T1,T2,T3 のジョインを行いたい場合，T2 を複製する代わりに，T1 をリモートデータベース 2 に複製すると，すべてのジョインがリモートデータベース 2 で行われる．このケースでも 1 つの複製で，効率的にジョインの演算が行われる．しかしながら，SQL 文の処理の負荷がすべてリモートデータベース 2 にかかってしまう．そこで，図 6 のように，さらに T2 をリモートデータベース 1 に複製すると，T1 と T2 のジョインはリモートデータベース 1 で行うことも可能となり，リモートデータベース 2 は T2 と T3 のジョインや T1,T2,T3 のジョインを行い，リモートデータベース 1 は T1 と T2 のジョインを行うといった負荷分散が可能となる．これは，それぞれの SQL 文において，どのリモートデータベースを使用するかといったポリシーを持たせることで実現できる．

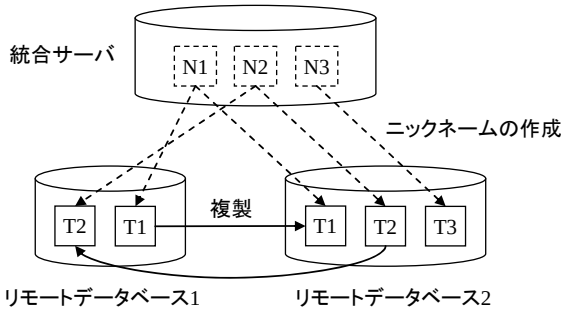

図 6. フェデレーション技術とレプリケーション技術の組み合わせ例 2

さらに T3 をリモートデータベース 1 に複製すると，すべてのジョインで負荷分散を図ることができるが，複製には負荷がかかることや，ディスクスペースを要することなどから，発行される SQL 文に応じた最適な構成を行うことが望まれる．

OLAP などのデータ分析処理では，SQL 文で集約関数などを使用して集約結果を得ており，その処理には時間がかかることから，事前に集計結果をキャッシュした表を用いることがよく行われている．この表はマテリアライズド照会表（MQT）またはマテリアライズドビューなどと呼ばれる．

図 7 のように，統合サーバに MQT を作成することで，MQT にヒットする SQL 文は MQT を使用して統合サーバのみで処理され，ヒットしない SQL 文はリモートデータベース 1 やリモートデータベース 2 を使用して SQL 文が処理されるといった構成を取ることができる．この構成により，統合サーバの資源も有効利用でき，効率的に SQL 文が処理できる．

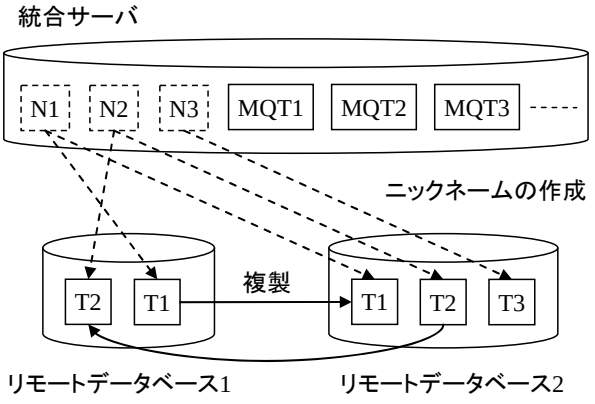

図 7. フェデレーション技術とレプリケーション技術と MQT の構成例

MQT は SELECT 文の演算結果を保持している表なので，その SELECT 文が参照している表（ベース表）が更新されたら，それに合わせて MQT も更新（リフレッシュ）する必要がある．ベース表がリモートデータベースにある MQT のリフレッシュ方法として，さまざまな手法が提案[10][11][12]されているが，いずれにせよ，ベース表の情報やその更新情報を用いて演算する必要があり，すべてのベース表が同じリモートデータベースに存在した方が効率的に処理できる．

MQT が保持するデータと発行される SQL 文の結果が一致する必要はなく，粒度が大きなものでも，それを基に演算する方が効率的と見積られれば，MQT が利用される．粒度が大きいと，そのリフレッシュが容易になるが，SQL 文を処理する際に演算処理が必要となるので，どういった MQT を作成するかも，ハードウェア資源を有効利用する際のポイントとなる．

その他の利用構成としては，レプリケーション技術は負荷分散や可用性の向上に利用することができるので，フェデレーション技術と組み合わせると図 8 のような構成が実現できる．

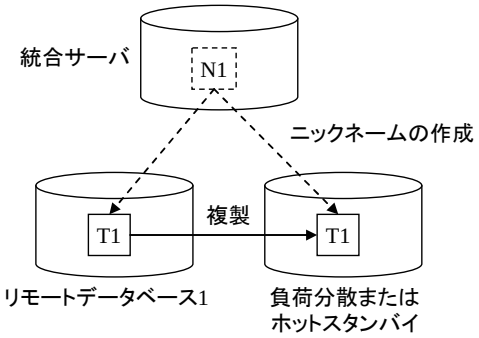

図 8. 負荷分散や高可用性に関する構成例

この構成で，N1 に対する SQL 文が発行されると，統合サーバでどちらの T1 にアクセスしたら効率的に処理できるかを判断し，最適な T1 にアクセスする．また，いずれかの T1 に障害が発生した場合，正常に動作している T1 にアクセスすることで可用性を向上させることができる．

4. フェデレーション技術の拡張

3 章の解決方法を実現するためにフェデレーション技術を拡張し，仮想的な表と現実の表の 1 対多のマッピングを可能とする．

まず仮想ニックネームを導入し複数のニックネームをこれに関連付ける．統合サーバに対する SQL 文の発行は仮想ニックネームに対して行う．

統合サーバはこれを解釈し，適当なニックネームに

置き換えてアクセスプランをコンパイルする．複数の候補がある場合,複数のアクセスプランが生成される．このうち,最適なアクセスプランを用いて SQL 文を処理する．見積りコストの値が近い複数のアクセスプランがある場合，これらをキャッシュし，ラウンドロビンで使用するなど負荷分散に利用する．

また，リモートデータベースを監視するなどにより，障害が検出された場合は，該当するアクセスプランを使用不可にし，正常なリモートデータベースを使用することで，可用性を向上させる．

以下，詳細を述べる．

4.1. 仮想ニックネーム登録

複数個のニックネームを仮想的に同一視するための仮想ニックネームの概念を導入する．

WFS におけるニックネーム登録の DDL 文を例にすると，以下のように拡張することが考えられる．

既存のニックネーム登録：
CREATE NICKNAME [ニックネーム名] FOR [リモートサーバ名]. [リモートスキーマ名]. [リモート表名]

⇩

仮想ニックネームと関連付けたニックネーム登録：
CREATE NICKNAME [実ニックネーム名] **AS [仮想ニックネーム名]** FOR [リモートサーバ名]. [リモートスキーマ名]. [リモート表名]

図 9. DDL の拡張例

後者の DDL 文ではニックネーム（実ニックネーム）が作成されると同時に仮想ニックネームとも関連付けられる．

例えば,以下の 2 つの DDL 文で 2 つのニックネーム N1, N2 が作成され，それらが仮想ニックネーム VN に関連付けられる．

```
CREATE NICKNAME N1 AS VN FOR RDB1.SYS.T1
CREATE NICKNAME N2 AS VN FOR RDB2.SYS.T2
```

この構成で VN を参照する SQL 文が発行されると，RDB1.SYS.T1 表と RDB2.SYS.T2 表のうち，最適な表にアクセスし，SQL 文が処理される．

4.2. 統合サーバコンポーネントの拡張

従来行っている SQL 文の処理について述べた後，仮想ニックネームと実ニックネームの 1 対多の関係の実現や，それを利用した効率性・可用性向上のための拡張について述べる．

4.2.1. 従来の SQL 処理

SQL 文がクライアントから発行されるとアクセスプランマネージャがすでにコンパイルされた SQL 文の実行形式（アクセスプラン）がキャッシュされていないかを検索する．キャッシュされていればそれを利用する．そうでない場合はコンパイラを呼び出し，SQL 文を実行形式にコンパイルし，キャッシュするとともに，それを利用する．

アクセスプランマネージャは利用するアクセスプランをランタイムインタプリタに渡し，ランタイムインタプリタはそれを実行する．

4.2.2. 機能の拡張

図 10 に統合サーバの主要コンポーネントと各種処理の流れを示す．

仮想ニックネームと実ニックネームの 1 対多の関係の実現や，それを利用した効率性・可用性向上のために，アクセスプランマネージャ，システム・カタログ・サービス，コンパイラ，ランタイムインタプリタの各コンポーネントを拡張する．

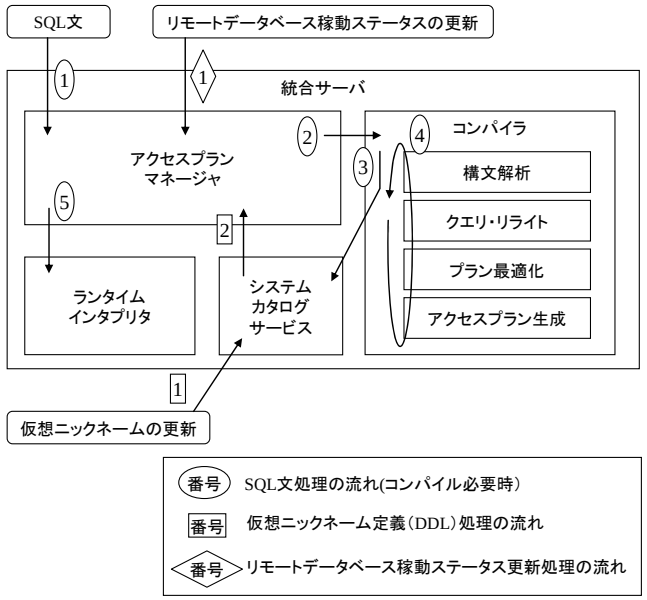

図 10. 統合サーバコンポーネントと各種処理の流れ

アクセスプランマネージャは以下の機能拡張を行う．

- SQL 文に対応する複数のアクセスプランをキャッシュする機能. それぞれのアクセスプランは，その見積りコスト，使用するリモートデータベース，参照している仮想ニックネーム，使用する実ニックネームなどを保持する．更に別途，

リモートデータベースの状態を保持し，アクセスプランを選択する際，該当するリモートデータベースが正常稼動していない場合，そのアクセスプランを選択しないようにする．また，SQL 文のアクセスプランがキャッシュされている時に新たに関連するニックネームが追加された場合，その情報もアクセスプランに保持し，アクセスプラン選択時にコンパイルし，キャッシュしているアクセスプランの見直しを行う．

- 複数のアクセスプランと上記で述べたような関連情報をコンパイラから得る機能
- 外部プロセスからリモートデータベースの状態を変更できる機能．
- 生成されたアクセスプランからポリシーに基づいて，例えばコストベースで効率的な 1 つまたは複数のアクセスプランをキャッシュし，最適な 1 つをランタイムインタプリタに渡す機能．
- ランタイムインタプリタに実行要求した結果，リモートデータベースの障害によるエラーとなった場合，そのリモートデータベースの状態をエラーとし，次の候補のアクセスプランをランタイムインタプリタに渡す機能．

システム・カタログ・サービスは以下の機能拡張を行う．

- 仮想ニックネームの登録情報を保持できる機能
- 仮想ニックネーム名に関連付けられた実ニックネーム名のリストを返す機能
- 仮想ニックネーム名登録時，既に該当するアクセスプランがキャッシュされていたら，そのアクセスプランに登録情報を付加する機能
- 仮想ニックネーム削除時，該当するアクセスプランがキャッシュされていたらそれを除去する機能．

コンパイラは以下の機能拡張を行う．

- コンパイラの構文解析機能により SQL 文で使用しているニックネームがわかるので，それを用いて，仮想ニックネームと実ニックネームの置き換え候補を作成し，順にコンパイルする．
- クエリ・リライト機能は SQL 文を効率的に処理できるように書き換える機能で，これを拡張し，仮想ニックネーム名を実ニックネームに置き換える．

ランタイムインタプリタは以下の機能拡張を行う．

- リモートデータベースの障害により SQL 文がエラーとなった場合，その情報を返す機能．

4.3. リモートデータベースの障害検知

別プロセスでリモートデータベースの監視を行うことで，統合サーバに対して SQL 文が発行される前にリモートデータベースの障害を検出することができる．

監視機能を持たない場合，統合サーバに対して SQL 文が発行された時点で，リモートデータベースに対する SQL 文のエラーコードからその障害を判断することができる．リモートデータベースの復旧は，復旧した後に，その状態フラグを正常に戻すことで統合サーバから使用可能となる．したがって，それを実現するためのコマンドまたは GUI を用意する．

5. プロトタイプによる検証

プロトタイプを開発して簡単な構成で動作検証とパフォーマンスの測定を行った．

プロトタイプは WFS を拡張した．仮想ニックネームの登録は，本論文で述べられている新たな構文は開発せず，既存のニックネーム登録構文のオプション情報を指定する機能を拡張することで実装した．また，見積りコストの値に対して適当な閾値を設け，その閾値に入る最適なアクセスプランをキャッシュし，ラウンドロビンで選択するようにした．さらに，動的に実ニックネームを追加・削除できるようにした．

これらの機能の正常動作を確認するとともに，MQT を登録した際は，MQT が正しく使用されることも確認した．

パフォーマンスの測定は，TPCH[13]のベンチマークで規定されている SQL 文（図 11）の Q12 を使用した．

```
select l_shipmode, sum(case when o_orderpriority='1-URGE
   NT' or o_orderpriority='2-HIGH' then 1 else 0 end) as hi
   gh_line_count, sum (case when o_orderpriority<>'1-URGE
   NT' and o_orderpriority<>'2-HIGH' then 1 else 0 end) as
    low_line_count
from orders, lineitem
where o_orderkey=l_orderkey and l_shipmode in ('MAIL', 'S
   HIP') and l_commitdate<l_receiptdate and l_shipdate<l_co
   mmitdate and l_receiptdate>='1994-01-01' and l_receiptdat
   e < CAST('1994-01-01' AS DATE) + 1 YEAR
group by l_shipmode order by l_shipmode
```

図 11. TPCH の SQL 文（Q12）

パフォーマンスを測定した構成を図 12 に示す．2 つのリモートデータベースにそれぞれ LINEITEM 表と ORDERS 表を保持する．この状態と，ORDERS 表をもう一方のリモートデータベースに複製した状態とでパフォーマンスの比較を行った．

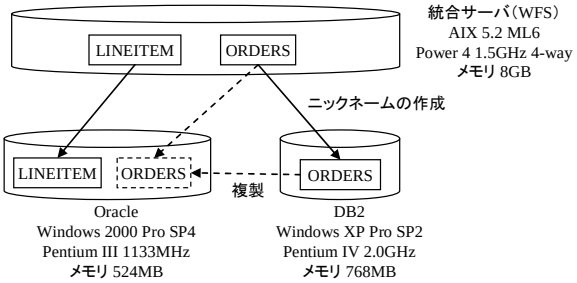

図 12. パフォーマンス測定の構成

比較結果は表 1 の通りで，後者の方がパフォーマンスが大幅に向上しており，予想通りの結果となった．

表 1. パフォーマンス比較

仮想ニックネームの使用	Q12 の処理時間
未使用	8 分 03 秒
使用	1 分 46 秒

6. むすび

本論文で述べられた手法は，ハードウェア資源を有効利用し，効率的に SQL 文を処理するために利用できる．また，この機能により，統合された情報を利用するアプリケーションの SQL 処理が効率的に行われるだけでなく，ピア・ツー・ピア型の複製と組み合わせて使用することで，1 つのデータベースを利用するアプリケーションの SQL 処理にも利用でき，負荷分散や可用性の向上を図ることができる．

本論文では見積りコストの値が最適な複数のアクセスプランをキャッシュして使用するとしているが，その閾値のデフォルトをどのくらいにするか，どのようなチューニング指針が提供されるべきかなどは今後の検討課題である．さらに，図 6 のように SQL 文に応じてアクセスするリモートデータベースを選択するようなポリシーのほかに，ログインユーザごとに選択するリモートデータベースを変えたり，時間帯によって変えたりするなどのポリシーも考えられる．

また，業務アプリケーションのハードウェア資源の利用状況，統合サーバに発行される SQL 文やデータ分析で使用する OLAP の構成，複製処理の負荷などから，どの表をどこに複製し，どのような MQT を作成したら，業務アプリケーションへの影響が少なく，ハードウェア資源を有効利用でき，統合サーバに発行される SQL 文が効率的に処理できるかを推奨し，適用するメカニズムが望まれる(図 13).

業務アプリケーションは一般的にその処理負荷が時間とともに変動する．情報分析アプリケーションが業務アプリケーションになるべく影響を与えないように，ハードウェア資源の利用状況に応じて，最適なアクセスプランを選択することが望まれる．この実現には，ハードウェア資源の利用状況を監視し，既存のコンパイラのメカニズムで使用している CPU レシオなどのパラメータを動的に変更して再コンパイルすることで可能だが，再コンパイルに時間を要する．そこで，コンパイル結果の見積りコストを値ではなく，CPU レシオなどのパラメータを変数とした式で保持し，動的に最適なアクセスプランを効率的に選択することも考えられる．

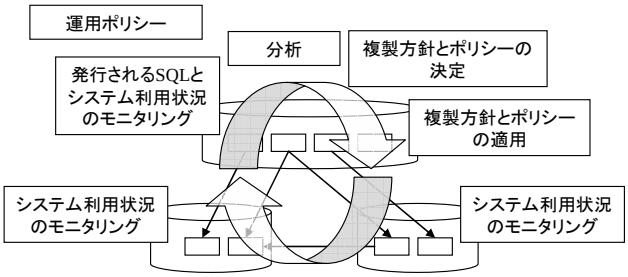

図 13. 複製方針とポリシーの決定および適用

文　献

[1] WebSphere Federation Server, http://www-06.ibm.com/jp/software/websphere/ii/fs/

[2] WebSphere Replication Server, http://www-06.ibm.com/jp/software/websphere/ii/rs/

[3] DB2 Version 9, http://www-06.ibm.com/jp/software/data/db2/v9/

[4] Administration Guide for Federated Systems, WebSphere Information Integration Version 9, SC19-1020-00

[5] A. Betawadkar-Norwood, E. Lin and I. Ursu, “Using data federation technology in IBM WebSphere Information Integrator: Data federation design and configuration”, IBM developerWorks, 2005, http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0506lin/

[6] L. Haas, E. Lin, “Data integration through database federation”, IBM System Journal, Vol. 41, No. 4, 2002

[7] L. Haas, E. Lin, “IBM Federated Database Technology”, IBM developerWorks, 2002, http://www-128.ibm.com/developerworks/db2/library/techarticle/0203haas/0203haas.html

[8] V. Josifovski, P. Schwarz, L. Haas and E. Lin, “Garlic: a new flavor of federated query processing for DB2”, SIGMOD 2002

[9] Performance Guide, DB2 Version 9 for Linux, Unix, and Windows, pp.149-pp.220

[10] D. Agrawal, A. El Abbadi, A. Singh, and T. Yurek, “Efficient Vew Maintenance at Data Warehouse”, SIGMOD, 1997

[11] K. Salem, K. Beyer, B. Lindsay, “How To Roll a Join: Asynchronous Incremental View Maintenance”, SIGMOD, 2000

[12] 松澤裕史，大川昌弘，福田剛志，”分散データベース環境におけるマテリアライズドビューの同期手法の実装とその評価”，DEWS，2007

[13] TPC-H, http://www.tpc.org/tpch/