D_013

データ統合サーバのトランザクション対応

Transaction Support of Data Federation Server

善家 直己† 杉嵜 百合子†
Naomi Zenge Yuriko Sugisaki

1. まえがき

情報統合では戦略的な利用を目的として，散在する多種多様なデータを単一のサーバ上に統合する．実現にあたっては，DB のレプリケーション機能等を利用してローカルに実データをコピーするアプローチのほか，異なるデータソース群を仮想化し，単一インタフェースでリアルタイムにアクセスするアプローチがあり，本稿で議論の対象とする「データ統合サーバ」は後者の機能を提供するものとする 1)．

データ統合サーバはリモートデータソースへのディスパッチャ的な役割を果たすだけでなく，リモートの統計情報や最適化アルゴリズムを駆使したデータベース操作や，トランザクション機能のサポートを行う．またこれに伴ったエラー処理，リカバリ処理も行う．

本稿ではデータ統合サーバの概要について触れた上で，サポートされるトランザクション機能の詳細と，その検証について議論する．

2. データ統合サーバの構成

データ統合サーバでは，RDBMS から提供されるクライアントライブラリを使用して，リモートのデータソースにアクセスする「ラッパー」または「コネクター」と，これを制御する「エンジン」，そして定義情報を記述した「カタログ」または「外部ファイル」から構成される．カタログには仮想テーブルと，リモートの実テーブルとの関連が記述されるが，エンジン内部のオプティマイザによって評価される実テーブル情報も記述される．図 1 に概略を示す．

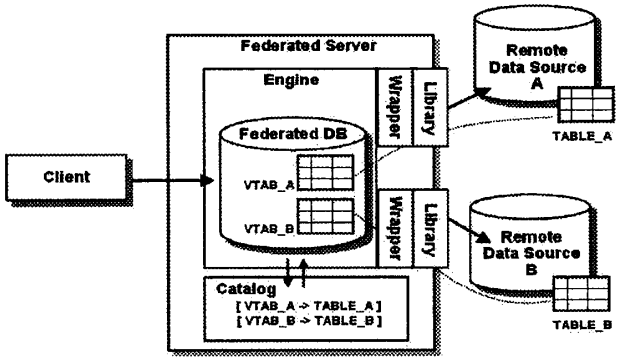

図 1 : データ統合サーバ概略図
Fig.1 Data Federated Server Overview

仮想テーブルに対して発行されたクライアントからの命令は，エンジンにて解釈され，ラッパー，クライアントライブラリを経由してデータソースに送信される．

このときラッパーでは SQL の方言の変換を行う．存在しない命令のシミュレーション，具体的には全件データを取得し，エンジン上で計算する等の操作はエンジンにて行われ，ラッパーは通信に専念する．

これ以外にラッパーはリモートデータソースの統計情報の取得，型変換，コードページ変換，エラー変換等の作業を行う．

ラッパーのロードにおいては，エンジンと同一プロセス内にロードする方法と，ロード専用の別プロセスにロードし，エンジンとはプロセス間通信する方法とがある．パフォーマンス的には前者が有利だが，後者にもいくつかの利点がある．すなわち，エラー伝播の局所化，クライアントスレッドの制限への対応，メモリ使用量の軽減などである．

複数のリモートデータソースが存在する場合は，最適化処理後，それぞれのデータソースに命令が送信され，受信した複数の結果をマージするなどして，クライアント側に返す．このときエンジン内部のオプティマイザは，カタログ内のリモート情報を参照しながら，ネットワークトラフィックを減らすべく，可能な限りリモートで処理する最適化を行う．ただしこのときも全件データを取得する場合がある．

3. トランザクション機能

データ統合サーバにおけるトランザクションは，内部に 2 フェーズコミット(以下，2PC)を持つ点において，通常の DBMS と異なる．

クライアントからは単独のデータベース上に，複数のテーブルがあるものとして操作するが，実際には異なるサーバ上の異なるテーブルを利用しているかもしれない．このとき必要となるのが 2PC である．必然的にデータ統合サーバはトランザクションマネージャの機能を果たすことになる．

3.1 実装

DBMS によっては固有のプロトコルを利用して 2PC をサポートするものもあるが，多種多様なデータソースを処理するデータ統合サーバでは，業界標準の X/Open XA のサポートが必須である．クライアントからの命令を受け取ったデータ統合サーバは，XA API を使用して各データソースにトランザクションの開始，PREPARE 処理，COMMIT 処理，ROLLBACK 処理，トランザクションの終了を行う．XA API は共通化されているが，xa_open のように DBMS ごとに異なるパラメータが必要なものもあるため，カタログを利用してこれを指定する 2)．

また PREPARE 後，トランザクション終了後に，それぞれログを出力し，エラーリカバリ処理に備える．このため 2PC を使用すると，DBMS 提供の専用プロトコルを使用した 1 フェーズコミット時に比べてパフォーマンスが劣る．

† 日本アイ・ビー・エム株式会社

2PC の利用を指定するオプションフラグを，カタログ内の情報で構成できるとよい．

3.2 エラー処理

トランザクションのエラー処理は，通常のトランザクションマネージャにおけるエラー処理と同様である．すなわち PREPARE フェーズで任意の DBMS がエラーを返せば，第 2 フェーズで全 DBMS に ROLLBACK が発行される．第 2 フェーズでのエラー，すなわち PREPARE フェーズ完了後，COMMIT 発行中，または ROLLBACK 発行中のエラーは未解決トランザクションとなり，自動的なリカバリ処理，または手動によるヒューリスティック処理の対象とする．

データ統合サーバはリカバリ処理の必要性を判断し，専用の外部プロセスを起動するか，または自身でリカバリ処理を行う．自動的なリカバリ処理まで待てない場合や，不可能な場合は，トランザクションの内容を意識しながら手動で，COMMIT，ROLLBACK，FORGET を呼び出す．

4. トランザクション機能の検証

データ統合サーバにおける検証では，様々な要素の組み合わせが必要となる．ここではトランザクション機能に絞って検証のパターンと準備，補助ツールを議論する．

4.1 検証パターン

検証のパターンとしては次のような要素をすべて組み合わせて確認する．特にこのトランザクション機能の検証では異常系の検証が一つのポイントとなる．

1) データソース．サポートする種類，バージョン．

2) クライアントからデータ統合サーバまでの接続プロトコル．1 フェーズコミットなのか，2PC か．2PC の場合，RDBMS 固有のものか，X/Open XA か．

3) 異常系検証としてのクラッシュ発生場所．リモートデータソースか，データ統合サーバ自身か．

4) 操作におけるクラッシュ発生箇所．xa_prepare 後，xa_commit 中，xa_rollback 中

5) 操作の種類．仮想テーブルに対する操作が照会か，更新かによりエンジン内部，リモートデータソースでのトランザクション処理は変化する．

また異常系の検証では，期待する未解決トランザクションが存在すること，それが自動リカバリ処理によって解消されることも確認する．

4.2 検証の準備

上で述べた検証パターンをすべて網羅するには様々な工夫が必要である．

まずはじめに検証パターンのすべてを文書化し，検証プログラムで変数化すべき要素と共通項目を洗い出す．またこのとき期待する結果も明確に記述する．ヒューリスティック処理では状態遷移図が必須である．検証は複雑であり，かつ長時間に及ぶためこの文書の完成度が効率を左右する．

続いて検証プログラムの作成に必要な言語の選定だが，Perl などのクロスプラットフォームなスクリプト言語を使用する．これには実際に検証を開始してからも多量の修正が現場レベルで発生すること，ある程度複雑な文字列処理，引数処理が必要なこと，OS コマンドの呼び出し，リダイレクト処理が必要なこと等の理由がある．特にヒューリスティック処理においては対話環境しか提供されない場合もあるため，リダイレクト処理が必須となる．

検証環境ではサポートするすべてのサーバー，OS，データソースを準備し，ネットワーク接続しておく．検証プログラムはコマンド行引数や環境変数で接続先を切り替えられるようにする．また rshell 等を利用して，リモートデータソースに直接コマンドを発行できるようにしておく．これは未解決トランザクションの確認，操作で必要になる．1 台のコンピュータに複数のデータソースを設定することは可能だが，1 回の検証で使用する場合は 1 台のコンピュータにつき 1 つのデータソースとなる．これはデータソースのクラッシュ等の作業が必要のためである．

検証プログラムはデータソース以外に，クライアント，データ統合サーバー間の接続プロトコル，クラッシュ箇所等を指定する引数を取る．ただし，外部からの操作だけで 2PC の検証は難しい．例えば第 2 フェーズの xa_rollback 処理を実現するには，トランザクション全体の COMMIT 処理に対して，第 1 フェーズの xa_prepare で任意のサーバがエラーを返さなければならない．そこで何らかのツールは必須である．

4.3 検証補助ツール

データ統合の対象となる DBMS 内部のコードを変更できる場合は，適切な箇所でクラッシュするような命令を挿入する．このとき外部ファイルにテストする異常値を記述できるとよい．クラッシュ後は，必要であれば未解決トランザクションの有無を確認後に，クラッシュしたサーバを再起動し，リカバリプログラムを自動実行する．

DBMS 内部を変更できない場合は，データ統合サーバで，クライアントライブラリからの正常な戻り値を，同様に外部ファイルから読み込んだ異常値で置換する．ただしこのテストはあくまでシミュレーションであり，DBMS とデータ統合サーバではステートが異なるため，以後の操作には注意する．すなわちテストケースごとにサーバの再起動を繰り返す．

5. まとめ

本稿ではデータ統合サーバの概要を紹介した上で，サポートされるトランザクション，特にユーザーの意識しない分散トランザクションについて焦点を当て，その構成と検証について述べた．検証項目は膨大であり，最初に作成する文書の完成度がそのまま検証の品質を左右する．また 2PC を網羅するにはツールの利用が不可欠である．

実際にはここでの議論以外に，DBMS 固有の 2PC プロトコルのサポートや外部の TM 傘下での稼動方法など検討すべき内容は多いが，いずれも 正確な文書化の後に検証を行えば，効率よく作業を進められる．

参考文献

1) http://www-06.ibm.com/jp/software/websphere/ii/v82/

2) Oracle Application Developer's Guide http://download-west.oracle.com/docs/cd/B19306_01/appdev.102/b14251/adfns_xa.htm