マルチメディア通信と分散処理 96－3
（2000．2．9）

モバイルコードを用いた協調型オープンデータベースシステムの構想

八木 哲　高橋直久

NTT 未来ねっと研究所

本稿では，インタネット上に散在するデータベースをコモディティとみなし，協調動作させるシステムの構想について述べる．提案するシステムでは，データフローグラフ形式のクエリをもとに，データベースを協調動作させる．即ち，各データベースで実行するクエリ文の組みをノード，クエリ文の組の依存関係をアークとして，クエリを表現する．このクエリに実行のための情報を付加したモバイルコードを，各データベースに対応するサーバに分配し，発火規則に基づいて実行制御する．本稿では，システム概要，実現法とともに，システムの適用実験についても示す．

Cooperative open database system based on mobile code

Satoru Yagi and Naohisa Takahashi

NTT Network Innovation Laboratories

This paper presents the concept of a system for making heterogeneous databases cooperate. The query is input to the system as a data-flow diagram. In the query, nodes express the set of queries executed by the individual databases. Arcs express dependences among nodes. The query and supplementary are translated into a mobile code. The mobile code is distributed to servers paired with the databases. Servers execute the mobile code based on the data-driven method. This paper describes the design of the system and an experiment in which the system was applied to a simple application.

1　はじめに

WWW ベースのシステムの普及ともあいまって，ネットワーク接続された様々な規模／種類のデータベースシステムが運用されている．これらを関連付けて共同利用することは極めて有意である．例えば，アプリケーション・サービス・プロバイダが提供する複数のデータベースサービスを連係して利用，モバイルデータベースや組み込みデータベースのバックグラウンド・データベースエンジン，企業内の異なる部門のデータベースの連係，複数の EC サイト (WWW サーバ) で構成する電子モール[1] のログ解析，地理分散した観測点からのデータ集計や解析，画像や音を扱う多様なデータベースを連動させる電子図書館などである．即ち，ネットワーク上に散在するデータベース群をコモディティとみなして協調させる，データベース指向のメタ・コンピューティングである．

このためには，WWW サーバと同様に，データベースの運用主体がサーバを立ち上げ登録するだけで，協調動作するデータベース群のコミュニティに参加できる形態が良い．1) 広域性と規模への対応，2) コミュニティへの参加と脱退が容易 (データベースの運用主体側で可能)，3) 各データベースの独自仕様の使用が可能，4) 柔軟なデータベースのアクセス権の設定 (データベースの運用主体側で可能)，5) 統一的なインタフェースの提供が課題となる．異種データベースを統括するアプローチに，マルチデータベース[2)3)4)] がある．また，

情報収集に重点を置いたメディエータ[5)6)7)]がある．これらのアプローチは，情報共有に焦点があり，スキーマ水準でデータベースの動作や検索結果の統括しており，スキーマを参照しながら全体を統括するマスタ機能を必要とする．この結果，マスタ機能への制御の集中のために上記1)2)の条件，統括するデータベースの構成に変更を加える場合にはスキーマ水準での変更を要するために上記2)3)の条件を満たすことが難しい．

これに対し，通信用プリミティブを付加したクエリ言語の水準で，データベースの動作を統括する，データベースを協調動作させるシステム (Make Resource Cooperate) を提案する．MRCでは，個々のデータベースで実行される，各データベースの仕様に従ったクエリ文の組みをノード，ノードの依存関係をアークとする，データフローグラフ形式でクエリを表現し，これを基盤にデータベースを協調動作させる．このクエリ表現を，DFGクエリと呼ぶことにする．DFGクエリの実行は，各データベースに対応するサーバ群により行われる．DFGクエリは，サーバ群へのマッピング情報が付加され，モバイルコードとして，サーバ群に分配される．サーバは，これを発火規則[8)]に基づいて解釈し，ノードであるクエリ文の実行に必要なデータが揃った時に，クエリ文をデータベースに発行し，アークに従ってその結果を転送する．MRCでは，実行と制御をサーバ群に分散させることで，上記1)2)の課題に対応し，データベースの独自仕様をDFGクエリのノードと対応するサーバに局所化することで，2)3)の課題に対応する．上記4)の課題については，実行と制御が分散されることを利用し，データベースの運用主体が，データベースに対応するサーバにおいて，アクセス権を設定することで対応する．上記5)の課題については，独自仕様が局所化されることを利用し，各データベースの運用主体が，アクセス可能な情報を，ノード形式の部品で提供することや，シンタックス変換系を用意することで対応する．

本稿では，まず提案するシステムの概要を示し(2章)，実現方法として，MRCの各構成要素について述べる(3章)．次に，実験用MRCを用いて，簡単なアプリケーションを題材に，MRCの適用実験を行う(4章)．最後に，本稿の内容をまとめ，今後の課題を示す(5章)．

2 システム概要

MRCの構成を図1に示す．DFGクエリは，各データベースで実行される，各データベースの仕様に従ったクエリ文の組みをノード，ノードの依存関係をアークとする，データフローグラフ形式のクエリ表現である．Gatewayは，DFGクエリに実行に必要な情報を付加し，モバイルコードを作成する．付加情報としては，リソース保護のための認証情報と，実行を担当するServerとデータベースへの配置情報が必要である．認証情報と配置情報は，リソース情報として管理する．さらにGatewayは，配置情報をもとにモバイルコードを部分グラフに分解し，実行を担当するServerに分配する．上記のマッピング過程は，自身のリソースを参照しながらServer群が談合し，決定する方法も考えられる．Serverは，Gatewayから分配されたモバイルコードの部分グラフを，発火規則に基づいて解釈し，対応するデータベースにクエリ文を発行する．クエリの実行結果は，DFGクエリのアークに従い，ストリームとして他のServerに転送する．実行結果は，アプリケーションがネットワークを介して，必要な結果が置かれたデータベースを直接アクセスする．

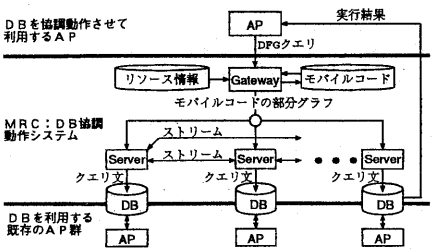


図 1: システムの構成

以降では，実現法として，DFGクエリとモバイルコードの仕様，リソース情報としての認証情報と配置情報，システムの動作概要を示す．

表 1: DFG クエリ

DFG クエリ	=	* (“QUERY” 実行形式 ノード名 “{” *入力 *クエリ *出力 “}”)
入力	=	“INPUT” 入力形式 データ形式 変数名 “=” ノード名 “.” ストリーム名 “;”
クエリ	=	各データベースのクエリ言語で記述されたクエリ文
出力	=	“OUTPUT” データ形式 ノード名 “.” ストリーム名 “=” 変数名 “;”

表 2: モバイルコード

モバイルコード	=	認証情報 * DB 指定 *サーバ指定 DFG クエリ
認証情報	=	“AUTH GATEWAY =” Gateway のアドレス “, ID = ” モバイルコードの ID “;” # (認証箇所 ユーザ名 パスワード)“;”
DB 指定	=	“DATABASE” データベース名 “=” DFG クエリの部分グラフの指定 “;”
サーバ指定	=	“SERVER” Gateway のアドレス “=” DFG クエリの部分グラフの指定 “;”

3 実現法

3.1 DFG クエリ

モバイルコードの構成を，拡張 BNF[9] を用いて表 1 に示し，各要素を説明する．

- DFG クエリ：データフローグラフ形式のクエリ．ノードとなるクエリ文の組に，アークとなる入力と出力の情報を付加した形式で記述する．ノード自体の属性として，ノード名と実行形式がある．実行形式は，停止性のための実行回数やトランザクションの扱いを示す．
- 入力: 属性として，入力ストリームをトークンとして消費するか，初期値として再利用するかなどを示す入力形式，データフォーマットを示すデータ形式，データの格納先を指定する変数名，転送元ノードを示すノード名，ストリーム名がある．
- クエリ: 各データベースのクエリ言語で記述されたクエリ文．
- 出力: 属性として，出力ストリームのデータフォーマットを示すデータ形式，転送先を示すノード名，ストリーム名，転送するデータの格納元を示す変数名がある．

入力，クエリ，出力の間のデータの引渡しについて，例えばクエリ言語として SQL を用いる場合，入力の属性の変数名として，表の名前か，CLI 形式[10] で記述された SQL 文のパラメータを指定する関数名を用いて，データをクエリに渡す．出力の属性の変数名としては，SQL 文を指定する関数名を用いて，クエリからデータを取得し，ストリームとして転送する．

3.2 モバイルコード

モバイルコードの構成を，拡張 BNF を用いて表 2 に示し，各要素を説明する．

- 認証情報：モバイルコードを識別するために Gateway のアドレス (URL 等) と，Gateway がローカルに付ける ID の組を用いる．また，認証のために，認証を行う箇所，ユーザ名，パスワードの組のリストを用いる．
- DB 指定：DFG クエリの部分グラフと，部分グラフに含まれるクエリ文が発行されるデータベースの対応関係を示す．例えば，ノード名のリストと URL 形式で指定されたデータベース名の組を用いて示す．
- サーバ指定：DFG クエリの部分グラフと，部分グラフに含まれるクエリの実行制御を行う Server の対応関係を示す．例えば，ノード名のリストと Server のアドレス (URL 等) を用いて示す．

3.3 認証情報

本システムでは，5 つの段階で考えられるリソース保護のための認証情報を，各リソース運用主体が設定することで，システム運用に柔軟性をもたせる．1 は Gateway の運用主体が管理するレベルであり，ユーザに割り振るアカウント情報を利用する．2,3 は，データベースを共同利用するコミュニティが管理するレベルであり，ユーザ

のアカウント情報とコミュニティのアカウント情報の対応をとり，利用する．4,5はデータベースと対応するServerの運用主体が管理するレベルであり，コミュニティのアカウント情報とデータベースのアカウント情報との対応をとり，利用する．

1. APがGatewayにアクセスする時．
2. Gatewayが Serverにアクセスする時．
3. Server間でストリームを転送する時．
4. Serverが データベースにログインする時．
5. データベースのデータにアクセスする時．

3.4 配置情報

配置情報は，DFGクエリの，データベースと対応するServerへの割り当て情報である．DFGクエリとデータベースの対応は，データベースのカタログ情報から求める．もしくは，明示的に指定する．通常，Serverとデータベースは組で運用されるため，DFGクエリとServerの対応は，対応付けられたデータベースから求められる．これらの，データベースとServerの情報は，データベースとServerの運用主体が登録する．

3.5 システムの動作概要

図1の構成にそって，システムの動作概要を示す．実行は，3つのフェーズからなる．

- Step1 モバイルコードの配布フェーズ．
  1. Gateway：モバイルコードを作成する．
  2. Gateway：Serverを予約する．
  3. Server　：データベースに接続し予約する．
  4. Gateway：全て予約できた場合は，モバイルコードをServerに分配し，実行指示を出す．
  5. Gateway：予約できないServerやデータベースがあった場合は，実行を中止する．
- Step2 モバイルコードの実行フェーズ．
  1. Server：ストリームを受信した場合，DFGクエリの入力の記述に従って変数に格納する．
  2. Server：DFGクエリのクエリ文の実行に必要な変数が揃った場合，DB指定に従ってデータベースにクエリを発行する．
  3. Server：DFGクエリの出力の記述に従って，送信可能な変数が生成された場合，サーバ指定に従ってストリームとして送信する．
  4. Server：全てのクエリ文の実行が終了すればGatewayに通知する．
  5. Server：異常が発生した場合はGatewayに通知する．
- Step3 モバイルコードの実行終了フェーズ．
  1. Gateway：全てのServerから正常終了が通知された場合は，正常終了とする．
  2. Gateway：異常終了の通知があった場合は，全Serverにアボートの指示を出す．

4 適用実験

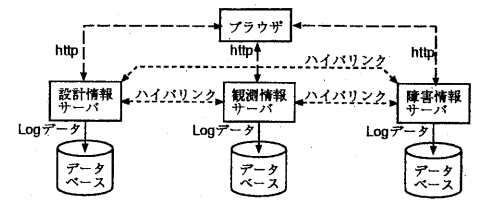
4.1 実験用MRC

実験用MRCの目的は，DFGクエリの動作と適用範囲の検証にある．GatewayとServerは，可搬性のためにJAVAで記述している．DFGクエリには，一般的なSQLが記述できる．リソース情報の扱いは簡略化しており，認証情報として，データベースにログインするのためのユーザ名とパワスワードの指定と，配置情報として，データベースとServerの明示的な指定が必要である．

4.2 実験内容

MRCの適用領域の1つとして，大規模電子モール向きの構成として考えられる，専用検索エンジン等で統括され，相互にリンクを張り合うElectronic Commerce用のサーバ群の，ログ解析がある．この問題では，電子モールの顧客動向を分析するために，各出店者(EC用サーバの運用主体)が運用する，複数のデータベースを協調させ，アクセス・ログを解析する．この問題と同種でかつ単純なものとして，3台のWWWサーバ[11]からなるネットワーク監視システム(図2)に対して，特定ユーザのアクセスシーケンスを調べ，障害対応のTipsを調べるという問題を取り上げる．

各サーバのログ(表3)は，バッチ的にデータベース[12]に格納される．ハイパリンクのジャンプ先をログに落すために，JAVA Scriptを用いて，専



設計情報サーバ：自ネット内にあり監視対象ネットの設計情報を保持する
観測情報サーバ：監視対象ネットにあり種々の観測を行なう
障害情報サーバ：外部からアクセス可能なネットにあり障害関係情報を保持する

図 2: ネットワーク監視システム

表 3: ログの項目

ip	リモートホスト
name	認証で得られたユーザ名
date	日付
time	時刻
http_from	Http ヘッダの Referer フィールド
http_here	リクエストされた URL パス
http_to	ハイパリンクのジャンプ先

用ページがアクセスされた時に，指定リンク先にジャンプさせる．同時に，mod_asis モジュールを用いて，Http ヘッダの Location フィールドにジャンプ先の URL を埋め込み，これをログに落す．

ログを保存する 3 台のデータベースを，実験用 MRC により協調動作させ，付録 A に示すモバイルコードを実行することで，付録 B に示す WWW サーバのアクセスシーケンスが得られる．サーバを渡り歩きながら，情報検索している様子が分かる．この結果は，障害対応の Tips を調べる基礎データの 1 つとして利用できる．

5 おわりに

本稿では，ネットワーク上に散在するデータベース群をコモディティとみなして協調させるために，データフローグラフ形式のクエリ表現をもとに，データベースの動作を統括する MRC について示した．MRC では，実行と制御を Server 群に分散させ，データベースの独自仕様を DFG クエリのノードと対応 Server に局所化する．このことにより，1) 広域性と規模への対応，2) データベースを協調動作させるコミュニティへの参加と脱退が容易，3) データベースの独自仕様の使用が可能，4) 柔軟なデータベースへのアクセス権の設定，といった課題に対処する．データベースの運用主体は，サーバを立ち上げ登録することで，データベース群を協調動作させるコミュニティに参加でき，データセンタのような利用ができる．

今後の課題として，DFG クエリの記述性，リソース情報，デバック機能も含む広域動作に耐えうる異常系，トランザクションの扱いを検討する．また，本システムの API として，独自仕様がノード内に局所化されることを利用した，統一的なインタフェースの提供の検討も行う．

謝 辞

日頃御指導いただく皆様方に深謝いたします．

参 考 文 献

1) *NETde* 通販,http://www.iijnet.or.jp/netde/index2.html

2) A.P.Sheth, J.A.Larson : Federated database System for Managing Distributed, Heterogeneous, and Autonomous Databases, ACM Computing Surveys, Vol.22, No.3, pp.183-236 (1990)

3) W.Litwib, L.Mark, N.Roussopoulos : Interoperability of Multiple Autonomous Databases, ACM Computing Surveys, Vol.22, No.3, pp.267-293 (1990)

4) 細川, 清木 : 関数型計算によるマルチデータベースシステムの問い合わせ処理方式, 情報処理学会論文誌,Vol 39 No.7,pp2217-pp2230 (1998)

5) G.Wiederhold : Mediators in the Architecture of Future Information System, IEEE Computer,25. pp.38-49 (1992)

6) R.Yerneni, C. Li, H.Garcia-Molina, J.Ullman : Computing Capabilities of Mediators, SIGMOD'99

7) 根本, 森嶋, 北川 : 異種情報源統合利用環境におけるメディエータの設計と開発, 信学技法,DE98-6,pp.39-46 (1998-05)

8) J.A.Sharp : データ・フロー・コンピューティング, サイエンス社,1987

9) D.H.Crocker : *Standard for the format of ARPA internet text messages*, RFC822 (1982).

10) C.J.Date,H.Darwen : 標準 SQL ガイド改訂第 4 版, アスキー,1999

11) *Apache*, http://www.apache.org/

12) *PostgreSQL*, http://www.postgresql.org/

A モバイルコード

```
AUTH GATEWAY = dse1:2000, ID = 1, USER = xxxx, PASSWD = xxxx;
            <-- ホスト名とポート番号で Gateway のアドレスを指定. アカウントは共通.
DATABASE //dse1/logs = triger;          <-- URL 形式でデータベースのアドレスを指定し,
DATABASE //dse1/logs = log_dse1;            ノードと対応付ける.
    :
DATABASE //dse1/logs = result;

SERVER dse1:2001 = triger;              <-- ホスト名とポート番号で Server のアドレス
SERVER dse1:2001 = log_dse1;                を指定しノードと対応付ける.
    :
SERVER dse1:2002 = result;

QUERY SHOT triger                               <-- ノード triger は SHOT 形式 (1 回だけ
{                                                   実行) で実行.
  create table seed (ip char(16), name char(32));
  insert into  seed values ('xx.xx.xx.xx', 'xxxx');
  select ip   from seed;
  select name from seed;                        <-- 検索キーを生成.
OUTPUT ATOM log_dse1.ip   = ITEM(3);            <-- 3,4 番目のクエリ文の結果を log_dse1,
    :                                               2,3 にデータ形式 ATOM, ストリーム名
OUTPUT ATOM log_dse1.name = ITEM(4);                ip,name で送信.
    :
}

QUERY SHOT log_dse1                             <-- ノード log_dse1 は SHOT 形式 (1 回だけ
{                                                   実行) で実行する.
INPUT TOKEN ATOM ITEM(1,1) = triger.ip;         <-- 1 番目のクエリ文の 1,2 番目のパラメータ
INPUT TOKEN ATOM ITEM(1,2) = triger.name;           にノード triger からのデータを代入.
  select date, time, http_here, http_to from log where ip=? and name=?;
                                                <-- ip と name でテーブルを検索
OUTPUT TABLE result.seq = ITEM(1);              <-- 1 番目のクエリの検索結果を, ノード
}                                                   result に, データ形式 TABLE, スト
                                                    リーム名 seq で送信.
QUERY SHOT result                               <-- ノード triger は SHOT 形式 (1 回だけ
{                                                   実行) で実行する
INPUT TOKEN TABLE tmp_r1 = log_dse1.seq;        <-- ノード log_dse1,2,3 からのデータを,
    :                                               データ名 tmp_r1,2,3 で保存.
  create table result ( date int, time int, http_here char(64), http_to char(64) );
  insert into  result ( select * from tmp_r1 union all select * from tmp_r2 union all
               select * from tmp_r3 );          <-- データの和をテーブル result に格納.
}
```

B 実行結果

```
logs=> select http_here, http_to from result order by date, time;
http_here                                  |http_to
-------------------------------------------+-----------------------------------------------
/htdocs/section_observe/url_od_js.asis     |http://dse2/htdocs/section_observe/index.html
/htdocs/section_observe/od_info.shtml      |<---- 観測情報サーバ DSE2 にジャンプ
/htdocs/ping/ping_info.shtml              <---- Ping によるポーリング結果を確認
/htdocs/snmp/snmp_info.shtml              <---- SNMP によるポーリング結果を確認
/htdocs/snmp/status_snmp_trap.shtml       <---- SNMP トラップメッセージを確認
/htdocs/syslog/router_1.shtml             <---- ルータ 1 の syslog メッセージを確認
/htdocs/console/router_1.shtml            <---- ルータ 1 のコンソール出力を確認
/htdocs/section_conf/url_conf_js.asis      |http://dse1/htdocs/section_conf/index.html
/htdocs/section_conf/nc_info.shtml         |<---- 設計情報サーバ DSE1 にジャンプ
/htdocs/config/router_1.shtml             <---- ルータ 1 の設定を確認
/htdocs/trouble/trouble_info.shtml         |-
/htdocs/section_fs/url_fs_js.asis          |http://dse3/htdocs/section_fs/index.html
/CGI/problem_list.cgi                      |<---- 障害情報サーバ DSE3 にジャンプ
/htdocs/section_observe/url_od_js.asis     |http://dse2/htdocs/section_observe/index.html
/htdocs/section_observe/od_info.shtml      |<---- 観測情報サーバ DSE2 にジャンプ
/htdocs/ping/ping_info.shtml              <---- Ping によるポーリング結果を確認
```