オープンな協調型データベースのためのアーキテクチャ

八木 哲[†] 高橋直久[††]

本稿では，インタネット上に散在するデータベース群をコモディティと見なし，これらを自在に協調動作させて1つのデータベースシステムとして機能させる，オープンな協調型データベースのためのアーキテクチャであるMRC（Make Resources Cooperate）について述べる．MRCの目的は，より高度に情報を利用するためのインフラの実現である．MRCは，データ駆動方式に基づいてデータベース群を協調動作させる実行制御系と，メタなクエリ表現形式を提供する利用者支援系と，協調動作させるデータベース群のリソース情報を管理し，実行制御系と利用者支援系，および利用者に対して必要な情報を提供するリソース管理系から構成される．MRCの各系は自律型モジュールの集合体であり，データベース群とともにある種のマクロなデータ駆動型計算機を構成する．本稿では，まず，MRCの実行制御系，利用者支援系，リソース管理系，およびMRCに基づいたシステムの運用形態について，枠組みを述べる．次に，MRCの中核である実行制御系について，実現法と実験用システムを用いた適用実験の内容を述べる．最後に，MRCの適性を考察する．

An Architecture for a Cooperative Open Database System

Satoru Yagi[†] and Naohisa Takahashi[††]

This paper presents the MRC (Make Resources Cooperate) architecture. MRC regards heterogeneous databases connected to the Internet as commodities and makes them cooperate to provide advanced information sharing facilities. MRC has three components. The execution management component makes databases cooperate based on the data driven method. The user support component absorbs heterogeneousness among the query languages of databases. The resource management component manages information about databases and provides it to the execution management and user support components and to the users. Each component is a group of autonomous elements. Along with databases, they form a macro data-flow computer. This paper describes the model of the three components of MRC, the design of the execution management component and experiments in which MRC was applied to simple applications. It also discusses the potential of MRC architecture.

1. はじめに

WWWベースのシステムの普及やネットワークの高速化とあいまって，ネットワーク接続された様々な規模や種類のデータベースが運用されている．これらのデータベース群を相互利用し，データベースの持つ構造化されたデータをそのまま共有することで，データベースを単体で利用する場合と比べて，より高度に情報を利用できる．たとえば，a) eマーケットプレイス[1),2)]における在庫データベースなどの相互利用や，複数のeマーケットプレイスの連携，b) カスタマサービスやサプライチェーンのために，個別運用されていたデータベースを連携させる，c) 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サービスの連携利用，d) 複数箇所に蔵書されたテキスト，画像，音，地図などの多様なデータを連動させる電子図書館，e) モバイルデータベースや組み込みデータベースとバックエンドのデータベースとの連携利用，f) 比較ショッピングサイト[3)]と提携を結んでいる電子モール群のログ解析や，負荷分散装置により単一の電子モールとして機能する広域分散したサーバ，キャッシュ，ファイアウォールなどのログの解析，g) 地理分散した拠点で観測された，気象，交通，ネットワークトラフィックなどの観測データや，研究データ[4)]の相互利用である．

このためには，データベース群を相互利用し，データベースの持つ構造化されたデータをそのまま共有しているコミュニティが複数存在したときに，地理分散している多数のデータベースの運用主体が，各々に運用している多様なデータベースをそのまま利用して，各コミュニティに自由に参加，脱退できるインフ

† NTT未来ねっと研究所
NTT Network Innovation Laboratories
†† 名古屋工業大学
Nagoya Institute of Technology

ラが求められる．すなわち，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見なし，これらを自在に協調動作させて 1 つのデータベースシステムとして機能させる，オープンな協調型データベースである．1) 規模と広域性への対応，2) データベース群を相互利用している各コミュニティへの参加と脱退が容易，3) 各データベースの独自仕様が利用可能，4) 各コミュニティを構成する利用者とデータベースの構成変更に対する対応が容易なアクセス権の管理方式，5) 統一的なインタフェースの提供が課題となる．

これらの課題に対して，我々はオープンな協調型データベースのためのアーキテクチャである MRC（Make Resources Cooperate）[5)~7)]の研究を進めている．MRC は，課題 1)，2) のために，分散並列型実行制御と自律型モジュールの集合体による構成を特徴とする実行制御系により，データベース群を協調動作させる．本実行制御系は，課題 3) のために，各データベースのクエリ言語を直接記述できるクエリ表現形式を入力として採用する．課題 4) のために，データベース群を相互利用する目的ごとにアクセス権を設定でき，各目的に参加する利用者とデータベースを変更するための作業範囲を局所化できる，3 階層型のアカウントモデルを採用する．また課題 5) のために，課題 3) の項で採用したクエリ表現形式に対するメタなクエリ表現形式を提供する，利用者支援系を採用する．加えて，協調動作させるデータベース群のリソース情報を管理し，実行制御系と利用者支援系，および利用者に対して必要な情報を提供する，分散協調型のリソース管理系を採用する．

本稿では，まず，関連研究について示し（2 章），MRC の構成要素である，実行制御系，利用者支援系，リソース管理系，および MRC に基づいたシステムの運用形態について，枠組みを示す（3 章）．次に，MRC の中核である実行制御系について，実現方法を示すとともに（4 章），実験用システムを複数の会社が協調して在庫管理を行う問題に適用し，適用性を検討する（5 章）．また，オープンな協調型データベースの実現に対する MRC の適性を考察する（6 章）．最後に，本稿の内容をまとめ，今後の課題を示す（7 章）．

2. 関 連 研 究

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[8)~11)]がある．また，情報収集に重点を置いたメディエータ[12)~14)]がある．これらのアプローチでは，何らかの水準において，スキーマを利用してデータベースを統括する集中型のマスタ機能があり，データベースがスキーマを介して強く結び付いている．このような，a) 集中型のマスタ機能による統括と，b) スキーマを介した結合を特徴とする方式では，次の問題点がある．まず，b) の特徴に起因して，統括下にあるデータベース群の構成や内容に変更があった場合には，すべてのデータベースの間でデータの整合を取りなおす必要があり，この作業の影響範囲が広い．このため，1 章で示した課題 2) を満たすことが困難である．また，単一のクエリ表現形式が用いられるために，各データベースの機能の積をとった部分のみが利用可能であり，1 章で示した課題 3) を満たすことが困難である．DIOM [15)]では，集中型のマスタ機能とデータベースとのインタフェースの構築を容易にするために，インタフェース定義言語を用意し，定義したインタフェースをライブラリ化している．次に，DIOM も含め，a) の特徴に起因して，データベース間のデータ通信が集中型のマスタ機能を仲介して行われる．この結果，集中型のマスタ機能の通信能力，および集中型のマスタ機能が接続しているネットワークの帯域が隘路になる．また，データベース間で直接データ通信する場合と比較して，通信経路が長くなるために通信効率が低くなる．このため，1 章で示した課題 1) を満たすことが困難である．MOCHA [16)]では，各データベースに閉じて行える処理について，JAVA のコードをデータベース側に転送，実行することにより，データ通信量を削減している．しかし，データ通信は集中型のマスタ機能を仲介しており，根本的な隘路の解消にはならない．エージェント方式に基づいた Bee-gent [17)]では，既存リソースにラッパをかぶせることによりエージェント化し，これらを仲介するモバイルエージェントを用いて既存リソースを協調動作させる．データ通信は集中型のマスタ機能を仲介しないが，基本的に開発ツールキットであり，クエリ言語の水準とは乖離がある．また，多数のデータベースを効率良く統括するための枠組みを提供していない．

3. 枠 組 み

3.1 概 要

MRC を構成する 3 つの系の関係を**図 1** にまとめる．実行制御系は，マルチデータベースシステムのクラス[18)]の中で，データベースの自律性が最も高い相互運用システムを基礎として，これにデータ駆動方式に基づく実行制御機構を付加したものである．入力は，DFG（Data Flow Graph）クエリと呼ぶデータフローグラフ形式のクエリである．DFG クエリを各

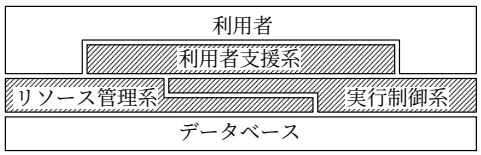

図 1 MRC の構造
Fig. 1 A construction of MRC.

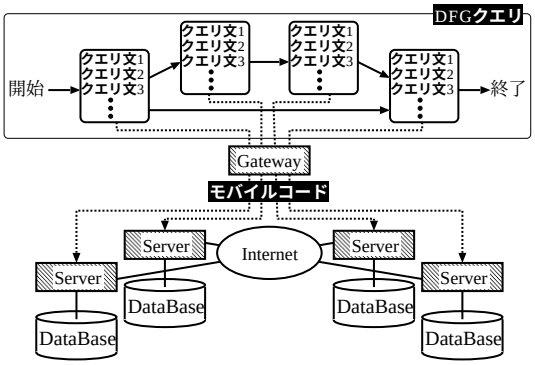

図 2 実行制御系
Fig. 2 The execution management system.

データベースと組をなす実行制御系の構成要素に分配し，データ駆動方式に基づいて実行制御することにより，データベース群を協調動作させる．利用者支援系は，利用者がより容易にクエリを記述できるように，DFG クエリよりも高水準なメタなクエリ表現形式を提供する．リソース管理系は，協調動作させるデータベース群のリソース情報を管理し，実行制御系と利用者支援系，および利用者に対して必要な情報を提供する．次節以降では，MRC の各系と MRC に基づいたシステムの運用形態について，枠組みを示す．

3.2 実行制御系

データベース群を協調動作させる実行制御系[5),6)]の枠組みを**図 2** に示し，構成要素を説明する．実行制御系の特徴は，分散並列型実行制御と自律型モジュールの集合体による構成である．

実行制御系の入力は，DFG クエリと呼ぶデータフローグラフ形式で表現されたクエリである．DFG クエリのノードは，各々のデータベースで実行される各々のデータベースのクエリ言語で記述された，クエリ文の組である．アークは，あるノードを構成するクエリ文を実行するために必要なデータが，他のノードを構成するクエリ文の実行により生成されるという，ノード間の依存関係である．トークンは，クエリ文の実行結果である．DFG クエリの記述クラスは，クエリ文の依存関係が表現可能という意味で，従来のクエリ文の記述クラスと基本的に同等である．

実行制御系の入力インタフェースである Gateway は，DFG クエリを受け取ると，DFG クエリのノードを，データベースと組で実行を受け持つ Server に対応付ける．対応付けの終わった DFG クエリを，モバイルコードと呼ぶことにする．データベースと Server の組の一覧など，対応付けに必要な情報はリソース管理系から取得する．さらに Gateway は，ネットワークを経由して，モバイルコードのノードを，対応付けた Server に分配する．Server は，発火規則[19)]に従うデータ駆動方式に基づいて，分配されたノードを自律的に実行制御する．発火規則とは，ある処理について，処理を実行するために必要なデータがすべて揃ったという事象を，処理の実行開始の契機とする実行制御方式である．実行を制御するための専用の信号を使用しない，簡明な方式である．すなわち，Server は，ノードを構成するクエリ文の実行に必要なデータがアークに従いトークンとして送信されてきた場合，トークンを組をなすデータベースのデータモデルに対応付けて一時保存する．必要なデータをすべて受信すると，ノードを構成するクエリ文を，組をなすデータベースに対して発行する．クエリ文の実行結果として絞り込まれたデータをトークンのデータモデルに対応付け，アークに従いトークンとして他の Server に直接送信する．

本実行制御系は，データベースと Server の組を単位処理装置として，モバイルコードを発火規則に従うデータ駆動方式に基づいて分散並列実行することにより，多数のデータベースを協調動作させる．これにともない，データベース間のデータ通信は，クエリの実行結果として絞り込まれたデータを対象として，データベースと組をなす Server の間で直接行う．この結果，通信負荷と通信トラフィックは，それぞれ各 Server と各 Server が接続しているネットワークに分散する．通信経路は，マスタ機能を仲介して通信を行う場合と比較して短縮される．また Server は，実行時に分配されるモバイルコードを発火規則に従うデータ駆動方式に基づいて実行制御する，自律型モジュールである．加えて，協調動作させるデータベース群の関係はモバイルコードにのみ記述され，Server には埋め込まれていない．この結果，協調動作させるデータベース群の構成変更に必要な作業は，モバイルコードの元となる DFG クエリの記述変更のみである．データベース間のデータの整合性については，3.5 節で述べる．

データベース群を相互利用する目的ごとにアクセス権を設定でき，各目的に参加する利用者とデータベースを変更するための作業範囲を局所化できる，3 階層

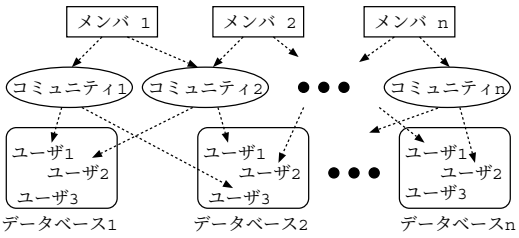

図 3 アカウントモデル
Fig. 3 The account model.

型のアカウントモデルを**図 3** に示す．本モデルでは，利用者に割り当てるアカウント“メンバ”とデータベースに作成するアカウント“ユーザ”を，データベース群を相互利用する目的ごとに作成するアカウント“コミュニティ”を介して対応付ける．

たとえば，図 3 のメンバ 1 のアカウントを持つ利用者は，コミュニティ1 とコミュニティ2 のアカウントが代表する 2 種類の目的でデータベースを利用できる．コミュニティ1 が代表する目的で利用可能なデータベースの範囲は，データベース 1 に対してユーザ 1 のアカウントでアクセスできる範囲と，データベース 2 に対してユーザ 3 のアカウントでアクセスできる範囲である．この場合，利用者は，メンバ 1 のアカウントを用いて Gateway にあるコミュニティ1 のアカウントにログインし，DFG クエリを入力する．Gateway は，データベース 1 と組をなす Server およびデータベース 2 と組をなす Server のコミュニティ1 のアカウントにログインし，モバイルコードを分配する．各 Server は，それぞれデータベース 1 とデータベース 2 に対して，ユーザ 1 とユーザ 3 のアカウントでログインし，クエリ文を発行する．“コミュニティ”の概念は，ある利用目的を代表するという意味において，UNIX におけるグループと基本的に同一である．しかしその有効範囲は，ネットワークを越えて Gateway，Server にまたがるために，セキュリティの観点から“コミュニティ”をアカウントとした．

本方式では，“メンバ”と“ユーザ”を“コミュニティ”を介して対応付けているために，利用者の構成に変更があった場合に修正が必要な情報は，利用者が属する Gateway が保有する，“コミュニティ”に属する“メンバ”の情報である．データベースの構成に変更があった場合に修正が必要な情報は，データベースが属する Server が保有する，‘コミュニティ”に属する“ユーザ”の情報である．したがって，アカウントの変更操作は，利用者が属する Gateway，あるいはデータベースが属する Server に閉じて行える．これは，各組織に関する変更が各組織に閉じて行えることを意味する．加えて，データベースのログイン方式に依存する“ユーザ”の認証情報の差異が，データベースと組をなす Server に隠蔽できる．

3.3 利用者支援系

DFG クエリでは，各データベースのクエリ言語をそのまま利用してノードにクエリ文を記述するために，各データベースに対する操作を詳細に記述できる反面，各データベースのクエリ言語の知識を必要とする．利用者支援系は，利用者がより容易にクエリを記述できるように，DFG クエリよりも高水準なメタなクエリ表現形式を提供する．メタなクエリ表現は，最終的には DFG クエリに展開される．次のようなアプローチが考えられる．マクロとシンタックスシュガの機能は，Server に持たせることもできる．

- マクロ：クエリ文の組，あるいは DFG クエリのノードの組を定型処理を行う部品として登録し，マクロ記述から展開する機能を提供する．Server において，特定のマクロのみを実行可能とすることにより，データベースへのアクセスを限定する手段としても利用できる．
- シンタックスシュガ：あるデータモデルに対するクエリ言語に標準がある一方で，製品ごとに差異がある場合に，標準のクエリ言語を用いた記述を，個々の製品のクエリ言語を用いた記述に変換する機能を提供する．DFG クエリのノードに記述するクエリ言語の種類を，データモデルの数に集約する．
- マルチデータベース言語システム[18]：トークンのデータモデルを，データベースのデータモデルを統合するメタデータモデルとして用いるクエリ言語を定義し，DFG クエリへの変換機能を提供する．多様なデータモデルがある場合にも統一的なクエリ表現形式を提供できるが，利用可能な機能は各データベースの機能の積をとった部分に限定される．

これらのメタなクエリ表現形式は，DFG クエリのノードに記述するクエリ文，あるいはノードとして，各データベースのクエリ言語とともに DFG クエリに併記可能である．必要に応じて，記述の詳細さと記述の容易さとを選択することができる．

3.4 リソース管理系

リソース管理系[7]は，各データベースに関して，次の項目からなるリソース情報を管理，提供する．これらのリソース情報を，実行制御系は，DFG クエリからモバイルコードを作成するときに利用する．利用者支援系は，利用可能なメタなクエリ表現形式の種別を

特定するときに利用する．利用者は，データベースの内容を理解するときに利用する．

(1) データベースのアドレス．URL 形式などによるデータベースの名前．
(2) Server のアドレス．データベースと組で運用される Server のアドレス．実行制御系（Gateway）が，DFG クエリからモバイルコードを作成するために利用する．
(3) データベースのカタログ情報とカタログ情報を補完する解説文．利用者が，データベースの内容を理解し，クエリを作成するために利用する．また，実行制御系（Gateway）が DFG クエリからモバイルコードを作成するときに，データの有無や文法の確認などのために利用する．
(4) ユーティリティ．Server が提供している，マクロやシンタックスシュガの一覧とその内容．利用者支援系が，使用可能なメタなクエリ表現形式を特定するために利用する．
(5) データベースと Server の処理能力，容量，特定処理に対するラウンド・トリップ・タイムなどの，負荷状態の測定値．実行制御系（Gateway）が，DFG クエリからモバイルコードを作成するときに，負荷状況の確認やミラーリングされたデータベースを選択するために利用する．

これらのリソース情報の特徴は，a) 変化の頻度が多いことと，b) 多数の情報源と利用先が広域分散していることである．リソース情報の検索負荷を分散させるために単純に複製を用いると，複製の同期負荷が問題になる．またキャッシュを用いると[20),21)]，キャッシュの有効期限と相反して，リソース情報の信頼性が問題になる．本リソース管理系では，複数の ResourceManager をデータベース，Server，Gateway と組を作るように分散配置して，リソース情報を管理，提供する．ResourceManager の間では，リソース情報のうち (1)，(2) の情報，および (3)，(4)，(5) の概要情報のみを複製し，(3)，(4)，(5) の情報は個々の ResourceManager で保持する．ここで概要情報とは，リソース情報から情報量や精度を落とした情報である．値の変化が抑制されるために，複製の同期をとる頻度を削減できる．リソース情報を利用する場合は，まず，組をなす ResourceManager から概要情報を取得し，これを利用する．概要情報では必要とする情報の量や精度が不十分な場合は，概要情報を索引として利用して必要なリソース情報を絞り込み，そのリソース情報を保持する ResourceManager に直接問い合わせる．本リソース管理系は，複製の同期負荷が比較的小さい概要情報を利用して，ResourceManager に対するアクセスを分散させる．

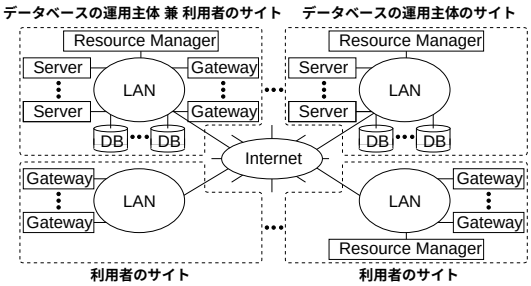

図 4 MRC システム
Fig. 4 A MRC system.

3.5 運 用 形 態

データベース群の相互利用は，コミュニティにおいて定めた合意事項に従って行う．各コミュニティでは，利用目的ごとに合意事項を作成する．各合意事項は，まず，利用形態として利用目的，利用者，使用するデータベースを規定する．次に，利用形態に従って Server と Gateway に設定するアカウント情報と，ResourceManager に登録する 3.4 節で示したリソース情報とを規定する．合意事項が規定する内容の例を，5.2 節で示す．MRC に基づくシステムの構成例を**図 4** に示す．各構成要素について，利用者は，利用頻度に応じた数の Gateway と ResourceManager を立ち上げる．利用頻度が少ない場合は，既存の Gateway や ResourceManager を共同利用することも可能である．データベースの運用主体は，データベースと，データベースと組をなす Server と，ResourceManager を立ち上げ，データベースのリソース情報を ResourceManager に登録する．また Gateway と Server には，アカウント情報を登録する．

データベース群の相互利用のためには，相互利用するデータの構成，データの要素の表記方法や表記内容などについて，コミュニティの中で整合をとる必要がある．このようなデータの整合性は，リソース情報の (3) データベースのカタログ情報とカタログ情報を補完する解説文として合意事項において規定され，基本的にデータベースの運用主体が責任を持つ．しかし，1 つのデータベースが複数のコミュニティに属することや，コミュニティに属するデータベースが随時入れ替わることなどを考えれば，つねに整合のとれた状態を保つことは困難である．このため，動的にデータの整合をとる方法を採用する．すなわち，DFG クエリのノードにデータクレンジング処理を記述し，モバイルコードの実行にともなってデータベースの間でデー

表 1 DFG クエリ
Table 1 DFG query.

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 “.” データ名 [“.” 要素名] “;”
出力	=	“OUTPUT” データ形式 ノード名 “.” データ名 “=” 変数名 “;”
クエリ	=	[“BEGIN{” クエリ文の組 “}”] [“ERROR_BEGIN{” クエリ文の組 “}”] [“BODY{” クエリ文の組 “}”] [“ERROR_BODY{” クエリ文の組 “}”] [“END{” クエリ文の組 “}”] [“ERROR_END{” クエリ文の組 “}”] [“ERROR{” クエリ文の組 “}”]

タが転送されるときにデータクレンジング処理を行い，データの整合をとる．DFG クエリのノードにデータクレンジング処理を記述する方法として，次のアプローチがある．

(i) クエリ文を用いてデータクレンジング処理をノードに記述する方法．受信したトークンを，トークンを格納する変数の型宣言に従って，型変換を行ってから格納するという，実行制御系の対応付け機能（4.3 節参照）と，クエリ言語の各種演算子を併用してデータクレンジング処理を記述する．SQL であれば，算術演算子，文字列演算子，外部結合演算子と対応表の組を利用して，データの要素に対する変換，結合，分解，置換操作を記述できる．たとえば，単位の変換，名前の連結，コード番号の置換などである．

(ii) データベースの代わりにデータクレンジングサーバを DFG クエリのノードに割り当て，クエリ文の代わりにデータ変換言語を用いてデータクレンジング処理をノードに記述する方法．たとえば，sed や XSLT などのデータ変換言語を解釈可能なサーバを用いる．

これらの方法によりノードにデータクレンジング処理を記述し，動的にデータの整合をとる場合は，データベースの運用主体があらかじめデータクレンジング処理をマクロとして作成し，ResourceManager に登録する．DFG クエリを作成するときは，このマクロを利用する．ResourceManager に登録するマクロは，リソース情報の (4) ユーティリティとして合意事項において規定する．利用者が，ResourceManager に登録されたリソース情報の (3) データベースのカタログ情報とカタログ情報を補完する解説文を利用して，独自に作成，利用する場合も考えられる．このような動的にデータの整合をとる方法では，データの整合をとるためのコミュニティ内の調整作業が，各合意事項に閉じて必要な部分に限定したうえで行える．前記 (i) の方法を用いたデータクレンジング処理用のマクロの例を，5.2 節で示す．

4. 実行制御系の実現方法

4.1 概　　要

実行制御系の実現方法を示すために，まず，実行制御系で用いるデータとして，利用者からの入力である DFG クエリの仕様と，実行コードであるモバイルコードの仕様と，データベース間のデータの受渡しに用いるトークンのデータモデルを示す．また，データを受け渡すときに必要となる，トークンのデータモデルとデータベースのデータモデルとの対応付け機能についても示す．次に，実行制御系の動作シーケンスを示す．最後に，3 階層型のアカウントモデルを実現するためのアカウント情報を示す．

4.2 クエリの表現

DFG クエリの仕様を拡張 BNF [22] 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表現．ノードとなるクエリ（クエリ文の組）に，アークのディスティネーションである入力と，アークのソースである出力の情報を加えた形式である．属性として，ノードの ID であるノード名，実行回数などのノードの実行形態を表す実行形式がある．
- 入力：アークのディスティネーション．属性として，入力データ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，入力データを受信するときのフォーマットを示すデータ形式，入力データの格納先を示す変数名，送信元ノードを示すノード名，入力データの名前，入力データのうちの格納する部分を示す要素名がある．
- 出力：アークのソース．属性として，出力データを送信するときのフォーマットを示すデータ形式，送信先を示すノード名，出力データの名前，出力データの格納元を示す変数名がある．
- クエリ：1) 各データベースのクエリ言語，2) Server で展開されるマクロ，3) Server のシンタックスシュガで変換されるクエリ言語を用いて記述されたクエリ文の組．

表 2 データ受渡しの指定方法の例
Table 2 An example of a data handling.

入力の項の変数名	=	表の名前 / “ITEM(“ 行指定 ”,“ パラメータ指定 ”)”
出力の項の変数名	=	“ITEM(“ 行指定 ”)”

クエリの項では，個々のデータベースが提供するトランザクションに関するクエリ文を利用して分散トランザクションを実現するために，クエリ文を 7 つのブロックに分割して記述する．

- BEGIN ブロック：初期化処理，トランザクションの開始宣言，ロック操作を記述する．また，DFG クエリの入力の項の変数名に記述した変数に対する型宣言を記述する．受信したトークンについて，この型宣言の記述がある場合には，宣言されたデータ型に変換した後に変数に格納する．型宣言の記述がない場合には，送信元で格納されていたときのデータ型を踏襲して変数に格納する．4.3 節を参照．すべてのノードの BEGIN ブロックの実行が正常終了した場合のみ，BODY ブロックを実行する．
- BODY ブロック：データベース処理の本体を記述する．DFG クエリの入力の項と出力の項は，BODY ブロックにのみ作用する．すべてのノードの BODY ブロックの実行が正常終了した場合のみ，END ブロックを実行する．
- END ブロック：一時データの削除やコミット操作を記述する．すべてのノードの END ブロックの実行が正常終了した場合のみ，DFG クエリの実行を正常終了したと判断する．
- ERROR_BEGIN ブロック：BEGIN ブロックの実行でエラーが発生した場合に実行する．ロールバック操作を記述する．このブロックの記述がない場合は，BEGIN ブロックの実行にともなって発生したエラーは無視する．
- ERROR_BODY ブロック：BODY ブロックの実行でエラーが発生した場合に実行する．一時データの削除やロールバック操作を記述する．このブロックの記述がない場合は，BODY ブロックの実行にともなって発生したエラーは無視する．
- ERROR_END ブロック：END ブロックの実行でエラーが発生した場合に実行する．一時データの削除やロールバック操作を記述する．このブロックの記述がない場合は，END ブロックの実行にともなって発生したエラーは無視する．
- ERROR ブロック：他のノードの実行でエラーが発生した場合に実行する．一時データの削除やロールバック操作を記述する．このブロックの記述がない場合は，他のノードの実行にともなって発生したエラーは無視する．

BEIGN ブロック，BODY ブロック，END ブロックを単位として，ノードの間で実行の同期をとることにより，2 相ロックと 2 相コミットを実現する．すなわち，すべてのノードの BEIGN ブロックを実行することでトランザクションの開始処理を行ってから，各ノードにおいて処理の本体である BODY ブロックを実行する．すべてのノードの BODY ブロックの実行が正常終了してから，各ノードの END ブロックを実行することでコミットの処理を行う．あるブロックの実行でエラーが発生した場合は，エラーが発生したノードではそのブロックに対応するエラーブロックを実行し，その他のノードでは ERROR ブロックを実行することで，全体をロールバックする．ロック期間を局所化したい場合は，DFG クエリを分割して順次実行する方法がある．

DFG クエリにおける，入力の項，クエリの項，出力の項の間でのデータ受渡しを指定する方法について，たとえばクエリ言語として SQL を用いる場合には，入力の項の変数名として表の名前か CLI 形式[23]で記述された SQL 文のパラメータを指定する関数を用いて，入力データをクエリ文に渡す．また，出力の項の変数名として SQL 文を指定する関数を用いて，クエリ文の実行結果を出力データとして取得する．変数名の指定方法の例を，拡張 BNF を用いて**表 2** に示す．「行指定」には，BODY ブロックにおける行番号やラベルを記述する．「パラメータ指定」には，CLI 形式で記述されたクエリ文内でのパラメータの出現順番を記述する．

次に，モバイルコードの仕様を拡張 BNF を用いて**表 3** に示し，各要素を説明する．

- ゲートウェイ：モバイルコードを分配した Gateway のアドレス．たとえば，IP アドレス，ホスト名，ポート番号により表す．
- データベース指定：DFG クエリの部分グラフと，部分グラフに含まれるノードを構成するクエリ文が発行されるデータベースとの対応関係．たとえば，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により指定する．

表 3 モバイルコード
Table 3 Mobile code.

モバイルコード	=	ゲートウェイ ∗データベース指定 ∗サーバ指定 DFG クエリ
ゲートウェイ	=	“GATEWAYE” Gateway のアドレス “;”
データベース指定	=	“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” Server のアドレス “=” DFG クエリの部分グラフの指定 “;”

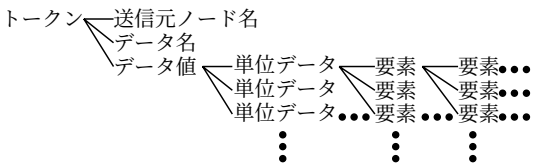

図 5 トークンのフォーマット
Fig. 5 A token format.

- 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ノードの実行制御を行う Server との対応関係．たとえば，ノード名のリストと，IP アドレス，ホスト名，ポート番号により表された Server のアドレスの組により指定する．

4.3 トークンのデータモデルと対応付け機能

トークンのデータモデルの表現能力は，トークンを用いてデータの受渡しが可能なデータベースのデータモデルを規定するため，表現力の高い XML [14] を用いる．トークンのフォーマットを**図 5** に示す．図 5 の記述要素と DFG クエリの記述要素とを対応付ければ，図 5 の「送信元ノード名」と「データ名」は，トークン送信側のノードにおけるノード名と出力の項のデータ名に対応する．この 2 つが，トークン受信側のノードにおける入力の項のノード名とデータ名に一致した場合に，入力の項の変数名で指定された変数に図 5 の「データ値」が格納される．「データ値」はデータの本体であり，「単位データ」は同じデータ構造が繰り返している場合の繰返しの単位である．「要素」はアトミックな値であり，名前や型を属性として持つ．

トークンの送受信時に必要となる，トークンのデータモデルとデータベースのデータモデルとの対応付け処理について，トークンと出力の項の変数名で指定された変数，およびトークンと入力の項の変数名で指定された変数との対応付けルールを，出力の項のデータ形式，および入力の項のデータ形式に記述するキーワードにより指定する．トークンを送信する場合は，対応付けルールに従って変数をトークンに変換し送信する．トークンを受信する場合は，対応付けルールに従ってトークンを変数に変換しデータベースに格納する．関係データベースを対象とする対応付けルールについて，キーワードと変換処理の内容を例示する．

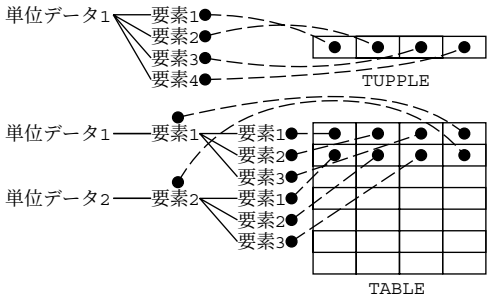

図 6 対応付けルールの例
Fig. 6 An example of mapping rule.

- CTL：あるクエリ文の実行が終了したことを示す．「データ値」は持たない．
- TUPPLE：トークンとタプルを対応付ける．**図 6** 上参照．「単位データ」は 1 つである．タプルからトークンへ対応付ける場合は，「要素」の階層は 1 段であり，タプルの各項目を「要素」に対応付ける．トークンからタプルへ対応付ける場合は，「要素」に階層があればこれを展開し，「要素」をタプルの各項目を対応付ける．
- TABLE：トークンと表と対応付ける．図 6 下参照．1 つの「単位データ」が 1 つのタプルに対応する．「単位データ」とタプルの対応関係は，上記の TUPPLE と同様である．

トークンを受信したときに，DFG クエリのノードの BEGIN ブロックにおいて，トークンを格納する変数に対する型宣言の記述がある場合は，まず，要素の名前を用いてトークンの要素とトークンを格納する変数の要素との対応関係をとる．次に，格納すべきトークンの要素を選別し，型宣言に従って要素のデータ型を変換する．最後に，前記の対応付けルールに従って，トークンを変数に格納する．恣意的な対応付けルールが必要な場合には，要素の名前を用いてトークンの要素とトークンを格納する変数の要素との対応関係を陽に指定し，DFG クエリとともに配布する方法が考えられる．また，トークンと他のデータモデルとの対応付けルールについて，重要度が増している XML や階層型データモデルに対しては，同形のまま対応付けが可能であると考えられる．また，ネットワーク型データモデルに対しては，XML のリンク機能 [14] の応用が

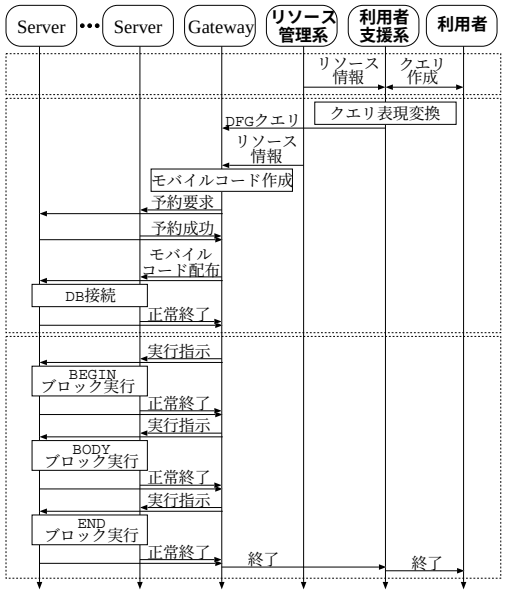

図 7 正常動作
Fig. 7 A normal sequence.

考えられる．トークンのデータモデルの制約により，データベースの間で完全なデータの受渡しが不可能な場合には，データベースのデータにインデックスとなる項を作成し，これをポインタとして利用する方法が考えられる．

4.4 動作シーケンス

MRC システムの正常時の動作を，**図 7** のシーケンス図に示す．3 つの段階がある．最初の段階はクエリの作成である．利用者は，リソース管理系が提供するデータベースのカタログ情報や解説文を利用して，利用者支援系と対話的にクエリを作成する．また必要に応じて，トークンとトークンを格納する変数との対応付けルールを作成する．次の段階は資源予約である．利用者支援系は，DFG クエリのうちメタなクエリ表現形式を用いて記述された部分を展開し Gateway に送る．Gateway は，リソース管理系が提供する Server の情報，データベースのカタログ情報，負荷状態の情報を利用して，DFG クエリからモバイルコードを作成する．さらにモバイルコードの実行を担当する Server を予約し，モバイルコードを分配する．各 Server は，モバイルコードを受け取ると，モバイルコードで指定されたデータベースに接続する．最後の段階はクエリの実行である．Gateway は，DFG クエリの各ブロックを単位に，各 Server の実行の同期をとる．各 Server は，各ブロックごとにクエリ文をデータベースに発行する．各段階においてエラーが発生した場合のシーケンスの概要は次のとおりである．

表 4 Gateway のアカウント情報
Table 4 An account information of the Gateway.

項目	内容
メンバ	利用者に割り当てられたアカウント名
コミュニティ	メンバが属するコミュニティのアカウント名

表 5 Server のアカウント情報
Table 5 An account information of the Server.

項目	内容
ユーザ	データベースに作られたアカウント名
データベース	ユーザのアカウントがあるデータベースの名前
コミュニティ	ユーザが属するコミュニティのアカウント名

(1) エラーが発生した Server は，エラー処理を行うとともにエラーの発生を Gateway に通知する．
(2) Gateway は他の Server にエラー処理の実行指示を出す．
(3) 各 Server はエラー処理を行い，結果を Gateway に返す．
(4) DFG クエリの実行を終了する．

4.5 アカウントモデル

3 階層型のアカウントモデルを実現するために，Gateway と Server は，**表 4**，**表 5** に示す項目を持つ表形式のアカウント情報を用いる．表中の各アカウント名の項は，認証に必要な情報を持つ表とリレーションシップをとる．ここでは 3.2 節の図 3 に従って，アカウント “メンバ 1” を持つ利用者が，アカウント “コミュニティ1” で利用可能なデータベースにおいて DFG クエリを実行する場合を例にとり，アカウント情報を説明する．利用者は，Gateway のアカウント情報の表に “メンバ 1” と “コミュニティ1” のアカウント名の組があり，“メンバ 1” に対する認証が成功したときのみ，Gateway にログインして DFG クエリを入力できる．Gateway は DFG クエリをモバイルコードに変換すると，データベース 1 と組をなす Server およびデータベース 2 と組をなす Server のアカウント “コミュニティ1” に対して，自身の持つアカウント “コミュニティ1” の認証情報を用いてログインし，モバイルコードを分配する．データベース 1 と組をなす Server では，この Server のアカウント情報の表にある，“コミュニティ1” およびモバイルコードで指定された “データベース 1” と組をなすアカウント “ユーザ 1” が選択される．データベース 2 と組をなす Server では，同様にしてアカウント “ユーザ 3” が選択される．各 Server は，それぞれモバイルコードで指定されたデータベース 1 とデータベース 2 に対して，アカウント “ユーザ 1” とアカウント “ユーザ 3” の認証情報を用いてログインし，クエリ文を発行する．

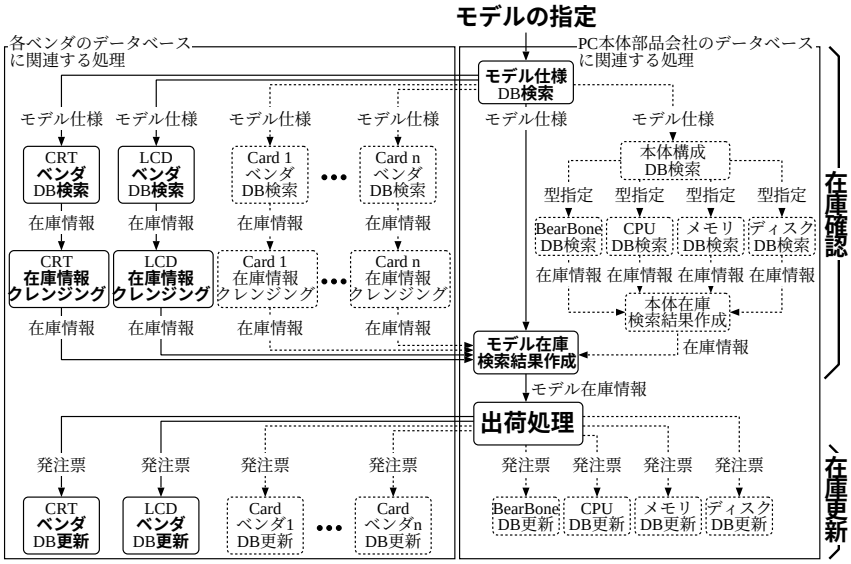

図 8 在庫管理システム
Fig. 8 A stock management system.

5. 適 用 実 験

5.1 実験用システム

実験用システムの目的は，MRC の中核である実行制御系の動作と MRC の適用領域の検証である．実行制御系の構成要素である Gateway と Server は，可搬性のために JAVA で記述した．Gateway について，ResourceManager がないために入力はモバイルコードであり，ファイルから直接読み込む．Server について，組をなすデータベースとして，関係データベースと XML データベースとを対象に実装を進めているが，現時点で組をなせるのは関係データベースのみである．関係データベースとの API は JDBC である．トークンのデータモデルとデータベースのデータモデルとの対応付け機能は，4.3 節で示した CTL，TUPPLE，TABLE をサポートする．XML 形式のトークンとの API は SAX であるが，作業用の DOM を介してトークンの単位データとタプルとを対応付ける．また Server は，複数のノードを同時に扱えるようにマルチスレッドを用いて実装した．各スレッドのトークン受信用ポートは 1 つであり，複数の接続要求があった場合はブロックされる．すなわち，Server は静的データ駆動方式で動作する．Gateway と Server の間，および Server と Server の間の通信は，信頼性のためにストリームソケットを用いた．Gateway と Server のアカウント情報は CSV 形式のファイルであり，認証情報は文字列によるアカウント名とパスワードの組である．また Gateway は，モバイルコードを Server に分配するときに乱数を用いて認証コードを生成し，モバイルコードとともに分配する．この認証コードは，モバイルコードの実行段階において，Gateway と Server の間，および Server と Server の間の通信の認証に用いられ，実行終了とともに破棄される．

5.2 実 験 内 容

例題として在庫管理問題を取り上げ，DFG クエリによる記述を試みることにより，MRC の適用性と MRC に基づくシステムを情報処理システムに組み込む場合の形態を検討する．この例では，PC 本体部品会社が主催するコミュニティにおいて，PC 本体部品会社とモニタ（CRT，LCD）やカード（Video，Modem，SCSI，etc.）を扱う複数のベンダとが，協調して完成モデルの出荷処理を行う．各ベンダは，他の取引先が主催するコミュニティにも参加している．処理内容をデータフロー図を用いて**図 8** に示し，内容を説明する．まず在庫確認を行う．「モデルの指定」に基づいて PC 本体部品会社の「モデル仕様 DB 検索」を実行して，「モデル仕様」を抽出する．各ベンダでは，「XX ベンダ DB 検索」を実行して，「モデル仕様」に合致した「在庫情報」を抽出する．この「在庫情報」に対して，「XX 在庫情報クレンジング」を実行して，このコミュニティに対してデータの整合がとれた「在庫情報」を作成する．PC 本体部品会社では，「本体構成 DB 検索」を実行して，「モデル仕様」に合致した構成部品の「型指定」を抽出する．さらに，各構成部品の

在庫データベースを検索することで各構成部品の「在庫情報」を作成し，「本体在庫検索結果作成」を実行することで本体の「在庫情報」を作成する．各ベンダと PC 本体部品会社からの「在庫情報」を,「モデル在庫検索結果作成」を実行することで結合し，「モデル在庫情報」を作成する．次に出荷処理を行う．「出荷処理」を実行することで，在庫を確認するとともに出荷量を決め，「発注票」を作成する．最後に在庫更新を行う．「発注票」に従って，各ベンダと PC 本体部品会社のデータベースの在庫数を更新する．

在庫管理を行うコミュニティにおける，図 8 に示した出荷処理のための合意事項は，以下の内容を規定する．データの整合性について，「XX 在庫情報クレンジング」は，コミュニティの主催者である PC 本体部品会社のデータベースをデータの整合性の基準とする，ベンダのデータベースから抽出されたデータのためのデータクレンジング処理である．ベンダがマクロとして作成する．ベンダは主催者のデータベースをデータの整合性の基準として，他の合意事項に関するマクロも作成する．

- 利用形態
  - (1) 利用目的として図 8 に示す出荷処理.
  - (2) 利用者として PC 本体部品会社の担当者.
  - (3) 利用するデータベースとして図 8 に表れるデータベース.
- Gateway と Server に設定するアカウント情報
  - (1) 図 8 の出荷処理を代表するコミュニティのアカウント.
  - (2) 図 8 の処理を実行する PC 本体部品会社の担当者に与えるメンバのアカウント.
  - (3) 図 8 に表れるデータベースに作成する，在庫情報にアクセス可能なユーザのアカウント.
- ResourceManager に登録するリソース情報（3.4 節参照）の内容
  - (1) 図 8 に表れるデータベースのアドレス.
  - (2) 図 8 に表れるデータベースと組をなす Server のアドレス.
  - (3) 図 8 の処理に用いるデータのカタログ情報と，これを補完する解説文．各データベースの運用主体が用意する.
  - (4) 図 8 の「XX 在庫情報クレンジング」に対応する，データクレンジング処理を行うマクロ．各ベンダが用意する.
  - (5) 図 8 に表れるデータベースに対する負荷測定項目.

表 6 動作環境
Table 6 An execution environment.

処理種別	データベース種別
PC 本体部品会社での処理	MySQL 3.22.32
CRT ベンダでの処理	PostgreSQL 6.5.3
LCD ベンダでの処理	PostgreSQL 6.5.3
出荷処理のスタブ	mSQL 2.0.11

図 8 のドキュメントは，基本的に同形のまま DFG クエリに変換できる．DFG クエリの作成は，PC 本体部品会社とベンダが，各々のデータベースに割り当てられるノードごとに分担することもできる．加えて，作成したノードをマクロとして登録し，マクロの展開を Server 側で行うことにすれば，各々のデータベースに対する操作の詳細を非公開にすることができる．この方法はセキュリティ面にメリットがある．図 8 の実線部分に相当する DFG クエリの記述例と，実験用システムで DFG クエリを実行したときの中間結果と最終結果を，付録の**図 9** に示す．また使用したデータベースの一覧を**表 6** に示す.

DFG クエリは，まず，全ノードの BEGIN ブロックが実行される．「CRT 在庫情報クレンジング」と「LCD 在庫情報クレンジング」のノードでは，CRT と LCD の在庫情報を型変換するための型宣言が行われる．「CRT ベンダ DB 更新」と「LCD ベンダ DB 更新」のノードでは，CRT と LCD の在庫情報がロックされる．次に，全ノードの BODY ブロックが実行され，図 8 の処理が行われる．「CRT 在庫情報クレンジング」では，型変換と単位変換によるデータクレンジング処理を行い，中間結果 2 から中間結果 3 を生成する．最終結果として，MODEL1 シリーズの PC に使われる CRT17 の在庫が 200 から 190 に更新される．最後に，全ノードの END ブロックが実行される．「CRT ベンダ DB 更新」と「LCD ベンダ DB 更新」のノードでは，CRT と LCD の在庫情報がコミットされる．現在の実験システムでは，DFG クエリのノードにはデータベースのみが割当て可能なために，「出荷処理」はデータベースを利用したスタブとした.

6. 考　　察

オープンな協調型データベースを実現するため課題である，1) 規模と広域性への対応，2) データベース群を相互利用している各コミュニティへの参加と脱退が容易，3) 各データベースの独自仕様が利用可能，4) 各コミュニティを構成する利用者とデータベースの構成変更に対する対応が容易なアクセス権の管理方式，5) 統一的なインタフェースの提供について，MRC と，

2 章で示した，a) 集中型のマスタ機能による統括と，b) スキーマを介した結合を特徴とする方式，およびエージェント方式とを比較する．

課題 1) について，MRC の実行制御系は，Gateway を入力インタフェース，データベースと Server の組を単位処理装置として，モバイルコードを発火規則に従うデータ駆動方式に基づいて分散並列実行する．加えて，利用者は Gateway と組でコミュニティへ参加し，データベースは Server と組でコミュニティへ参加するために，規模の拡大に合わせて処理能力が必然的に拡張される．またこの特長が，並列性の利用が容易な反面，実行時に多量のリソースを必要とする発火規則に従うデータ駆動方式の採用を可能にしている．これらの結果，意図的に処理能力を拡張する必要がある a) の特徴を持つ方式やエージェント方式と比較して，規模の拡大が容易である．5.2 節，図 8 の例において，ベンダ数が増加し在庫集計処理「モデル在庫検索結果作成」の負荷が増加する場合を想定する．a) の特徴を持つ方式では，検索処理「XX ベンダ DB 検索」を要素データベースが担当し，在庫集計処理「モデル在庫検索結果作成」をマスタ機能が担当するため，マスタ機能の再構成が必要になる．エージェント方式では，検索処理「XX ベンダ DB 検索」の結果を収集するモバイルエージェントの再構成が必要になる．MRC では，PC 本体部品を対象とした中間在庫集計処理「本体在庫検索結果作成」と同様に，部品種別ごとにベンダのグループを作り，中間在庫集計用のノードを設けてグループの適当な単位処理装置に割り当てるだけでよい．

また MRC の実行制御系は，データベース間のデータ通信を，クエリの実行結果として絞り込まれたデータを対象に，データベースと組をなす Server の間で直接行う．この結果，通信負荷と通信トラフィックは，それぞれ各 Server と各 Server が接続しているネットワークに分散するため，マスタ機能とマスタ機能が接続しているネットワークに通信負荷と通信トラフィックが集中する a) の特徴を持つ方式と比較して，隘路が解消される．通信経路は，マスタ機能を仲介して通信を行う a) の特徴を持つ方式と比較して短縮されるため，通信効率が改善される．5.2 節，図 8 の例にあてはめると，a) の特徴を持つ方式では，在庫集計処理を行うマスタ機能を中心に，検索処理および更新処理を行う要素データベースとの間で，データがスター型に流れる．MRC では，各ノードに割り当てられた単位処理装置の間で，図 8 のアークに従ってデータが流れる．このため，通信負荷と通信トラフィックは，マスタ機能とマスタ機能が接続しているネットワークから，ノードが割り当てられた単位処理装置と単位処理装置の間のネットワークに分散する．通信経路は，三角形の二辺分（データベース → マスタ機能 → データベース）から，一辺分（データベース → データベース）に短縮される．

課題 2) について，Server は，実行時に分配されるモバイルコードを発火規則に従うデータ駆動方式に基づいて実行制御する，自律型モジュールである．加えて，協調動作するデータベース群の関係はモバイルコードにのみ記述され，Server には埋め込まれていない．またデータの整合性について，DFG クエリのノードにデータクレンジング処理を記述し，モバイルコードの実行にともなってデータベースの間でデータを転送するときにデータクレンジング処理を行い，データの整合をとるという，動的にデータの整合をとる方法を採用している．これらの結果，協調動作させるデータベース群の構成変更に必要な作業は，モバイルコードの元となる DFG クエリの記述変更のみである．また，データの整合をとるためのコミュニティ内の調整作業は，各合意事項に閉じて必要な部分に限定したうえで行える．すべてのデータベースの間でデータの整合をとりなおす必要がある b) の特徴を持つ方式と比較して，変更作業の範囲が局所的であり，コミュニティへの参加と脱退が容易である．5.2 節，図 8 の例において，新規にベンダが加わる場合を想定する．MRC では，新規のベンダは合意事項に従って Server にアカウント情報を登録し，ResourceManager にリソース情報を登録する．この後，このベンダがコミュニティ向けに作成したデータクレンジング用のマクロを用いて DFG クエリの記述を変更すればよい．この一連の作業は，既存のベンダには無関係であり，コミュニティの他の合意事項で規定された処理の実行にも影響を及ぼさない．

課題 3) について，各データベースで行う処理は，DFG クエリのノードに各データベースのクエリ言語をそのまま用いて記述可能であり，各データベースの機能の和をとった部分が利用できる．この結果，各データベースの機能の積をとった部分のみが利用可能な単一のクエリ言語を使用する b) の特徴を持つ方式と比較して，利用可能な各データベースの機能の範囲が広い．

課題 4) について，利用者に割り当てるアカウント “メンバ” とデータベースに作成するアカウント “ユーザ” を，データベース群を相互利用する目的ごとに作成するアカウント “コミュニティ” を介して対応付け

ている．この結果，利用者の構成に変更があった場合には，利用者が属する Gateway のアカウント情報を修正するだけでよい．また，データベースの構成に変更があった場合には，データベースが属する Server のアカウント情報を修正するだけでよい．単純に “メンバ” と “ユーザ” を直接対応付ける方法では，次の問題が生じる．対応関係を各 Gateway で管理すれば，データベースの構成を変更するために，関連するすべての Gateway で変更が必要になる．また，対応関係を各 Server で管理すれば，利用者の構成を変更するために，関連するすべての Server で変更が必要になる．直接対応付ける方法と比較して，本方式ではアカウントの変更に必要な作業の範囲を局所化できる．すなわち，各組織に関する変更が各組織に閉じて行えるために，変更のためのコストが低い．5.2 節，図 8 の例にあてはめると，新規にベンダが加わる場合には，このベンダの Server にアカウント情報を追加するだけでよい．また利用者が変わる場合には，PC 本体部品会社の Gateway のアカウント情報を変更するだけでよい．

課題 5) について，課題 3) と相反するこの課題を解決するために，リソース管理系によりデータベースのカタログ情報や解説文を利用者に提供し，データベースの内容理解を支援する．加えて，利用者支援系により DFG クエリよりも高水準なメタなクエリ表現形式を提供する．メタなクエリ表現形式は，DFG クエリのノードに記述するクエリ文，あるいはノードとして，各データベースのクエリ言語とともに DFG クエリに併記可能である．記述の詳細さのための各データベースのクエリ言語を用いた記述と，記述の容易さのためのメタなクエリ表現形式を用いた記述とを，必要に応じて選択できる．

MRC の適用性について，5.2 節の実験では「出荷処理」にデータベースを利用したスタブを用いたが，データベースと同様にサーバを DFG クエリのノードに割当て可能にすれば，データベースとサーバを連携させたトランザクション処理が可能なことが分かる．たとえば，モバイルコードの「サーバ指定」の項の拡張と，サーバ向けのイベント駆動型 API の規定により，実現できる見通しがある．この拡張は，データクレンジングサーバをノードに割り当てる場合にも利用できる．利用者の観点から，データフロー図を用いたドキュメントから DFG クエリへの変換と，DFG クエリからモバイルコードへの変換が同形のまま行われるために，DFG クエリの可読性が高く動作の理解が容易という特長がある．

7. おわりに

本稿では，インタネット上に散在するデータベース群をコモディティと見なし，これらを自在に協調動作させて 1 つのデータベースシステムとして機能させる，オープンな協調型データベースのためのアーキテクチャである MRC について示した．MRC の各系は自律型モジュールの集合体であり，データベース群とともにある種のマクロなデータ駆動型計算機を構成する．また，実験用システムを用いた適用実験により，MRC の適用性と MRC に基づくシステムを情報処理システムに組み込む場合の形態を検討した．

今後の課題として，実行制御系に関しては，DFG クエリのノードに割当て可能な，データベースとサーバの種別を拡張する．データベースのために，データベースのデータモデルとトークンのデータモデルとの対応付け機能の拡張と，データクレンジングサーバを利用したデータクレンジング機能の検討を行う．サーバのために，モバイルコードの拡張とサーバ向けの API の検討を行う．また実行性能の定量的な評価を行うとともに，並列処理型の Server やデータクレンジングサーバなどの直接的な方法と，中間処理結果の再利用や複数の DFG クエリの連動を可能にする実行スケジューリング機能などの間接的な方法の両面から，性能向上手法を検討する．加えて，認証や通信暗号化などのセキュリティ面の検討がある．利用者支援系に関しては，DFG クエリに対するメタなクエリ表現形式，およびクエリ作成環境の具体化がある．リソース管理系に関しては，動的に変化するリソース情報を，広域に提供し利用する手法の具体化がある．また，コミュニティの合意事項の形成シーケンスや，合意事項に従った DFG クエリの作成シーケンスなど，システムの運用面の検討がある．

参考文献

1) Japan Automotive Network eXchange. http://www.jnx.ne.jp/
2) インタネット取引所ディレクトリ．http://nis.nikkeibp.co.jp/nis/ex/
3) ISIZE SMART SHOPPING. http://www.isize.com/s-shopping/
4) 高木，金久：ゲノムネットのデータベース利用法［第 2 版］，2 章，pp.11–42，共立出版 (1998).
5) 八木，高橋：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，2000-DPS-96-3, pp.13–18 (2000).
6) 八木，高橋：オープンな協調型データベースの構築法，2000-DBS-122-32, pp.245–252 (2000).

7) 八木，高橋：メタ・コンピューティングのためのリソース管理フレームワーク，第 61 回情報処理学会全国大会，第 3 分冊，pp.519–520 (2000).
8) Sheth, A.P. and Larson, J.A.: Federated Database System for Managing Distributed, Heterogeneous, and Autonomous Databases, *ACM Computing Surveys*, Vol.22, No.3, pp.183–236 (1990).
9) Litwin, W., Mark, L. and Roussopoulos, N.: Interoperability of Multiple Autonomous Databases, *ACM Computing Surveys*, Vol.22, No.3, pp.267–293 (1990).
10) 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, pp.2217–2230 (1998).
11) Lakshmanan, L.V.S., Sadri, F. and Subramanian, I.N.: SchemaSQL—A Language for Interoperability in Relational Multi-database Systems, *Proc. 22nd International Conference on Very Large Data Bases*, pp.239–250 (1996).
12) Wiederhold, G.: Mediators in the Architecture of Future Information Systems, *IEEE Computer*, 25, pp.38–49 (1992).
13) Morishima, A. and Kitagawa, H.: Info Weaver: Dynamic and Tailor-Made Integration of Structured Documents, Web, and Databases, *Proc. ACM Digital Libraries*, pp.235–236 (1999).
14) Goldfarb, C.F. and Prescod, P.: *The XML Handbook*, Section 7, 9, 29, 34, Prentice Hall (1998). 安藤慶一（訳）：XML 技術大全，プレンティスホール出版 (1999).
15) Liu, L. and Pu, C.: The Distributed Interoperable Object Model and Its Application to Large-scale Interoperable Database systems, *CIKM '95*, pp.105–112, ACM (1995).
16) Rodriguez, M. and Roussonpoulos, N.: MOCHA: A Self-Extensible Database Middleware System for Distributed Data Sources, *MOD2000*, pp.213–224, ACM (2000).
17) 川村，田原，長谷川，大須賀，本位田：Bee-gent 移動型仲介エージェントによる既存システムの柔軟な活用を目的としたマルチエージェントフレームワーク，信学会論文誌 D-1, Vol.J82-D-1, No.9, pp.1165–1180 (1999).
18) Bright, M.W., Hurson, A.R. and Pakzad, S.H.: A Taxonomy and Current Issues in Multidatabase Systems, *IEEE Computer*, Vol.25, No.3, pp.50–59 (1992).
19) Sharp, J.A.: *Data Flow Computing*, Ellis Horwood (1985). 富田眞治（訳）：データ・フロー・コンピューティング，サイエンス社 (1987).
20) Fitzgerald, S., Foster, I., Kesselman, C., Laszewski, G.V., Smith, W. and Tuecke, S.: A Directory Service for Configuring High-Performance Distributed Computations, *Proc. 6th IEEE Symp. on Hight-Performance Distributed Computing*, pp.365–375 (1997).
21) Wolski, R., Swany, M. and Fitzgerald, S.: White Paper: Developing a Dynamic Performance Information Infrastructure for Grid Systems. http://www-didc.lbl.gov/GridPerf/ (2000).
22) Crocker, D.H.: Standard for the format of ARPA internet text messages, RFC822 (1982).
23) Date, C.J. and Darwen, H.: *A Guide to THE SQL STANDARD*, Forth Edition, Addison-Wesley (1997). QUIPU LLC（訳）：標準 SQL ガイド改訂第 4 版，アスキー (1999).

付録　DFG クエリの記述例と実行例

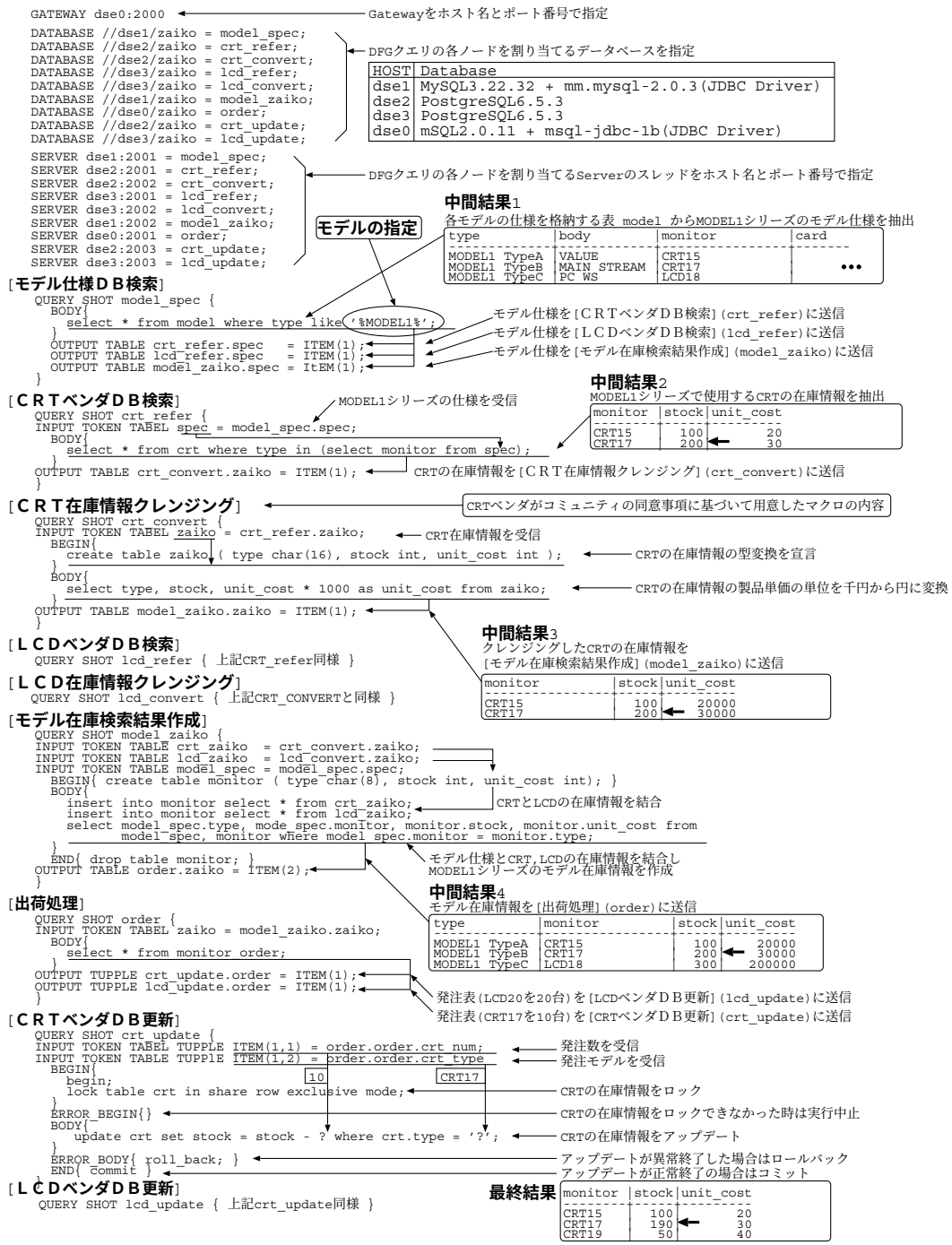

図 9　出荷処理
Fig. 9　A shipping operation.

(平成 12 年 12 月 20 日受付)
(平成 13 年 3 月 30 日採録)

(担当編集委員 宝珍 輝尚)

八木 哲(正会員)

昭和 41 年生．平成 4 年大阪大学大学院工学研究科電子工学専攻博士課程前期修了．同年日本電信電話株式会社入社．ソフトウェア工学，交換機シミュレータ，ネットワーク・システム，広域データベース等の研究開発に従事．現在，NTT 未来ねっと研究所所属．

高橋 直久(正会員)

昭和 26 年生．昭和 49 年電気通信大学応用電子工学科卒業．昭和 51 年同大学大学院修士課程修了．同年日本電信電話公社武蔵野電気通信研究所入所．以来，機能分散型並列計算機，データフロー型計算システム，ソフトウェア工学，コンピュータ・ネットワーク等の研究に従事．平成 13 年より名古屋工業大学電気情報工学科教授．工学博士(東工大)．電子情報通信学会，日本ソフトウェア科学会，ACM 各会員．