マルチメディア通信と分散処理 64−10
（１９９４．３．３）

ＤＢ流通基本システム（DB-STREAM）とその適用

塩原 寿子　池田 哲夫　村田 達彦　井戸 正幸

ＮＴＴ情報通信網研究所

データベースの相互利用形態の一つとしてＤＢ流通システムの利用がある。筆者らは、ＤＢ流通システムを簡易に構築するためのプラットフォームソフトウェアとして、DB-STREAM を提案する。

DB-STREAM は、流通の機能をソフトウェア部品として実現した処理メソッドの提供と、処理メソッドの組合せを非手続き的に記述することによって流通処理の流れを定義するシナリオ方式とを特徴とする。

DB-STREAM を用いることにより、ＤＢ流通システムの初期構築と保守が容易になる。

A Data Delivery Platform System

Shiohara Hisako　Ikeda Tetsuo　Murata Tatsuhiko　Ido Masayuki

NTT Network Information Systems Laboratories

1 - 2356, Take, Yokosuka-shi, Kanagawa, 238-03 Japan

As a practical solution of integrated use of multiple databases, a data delivery system approach is available. Data delivery system is placed between different databases and transforms and delivers data of one database to another.

This paper describes a platform software system, called "DB-STREAM", which facilitates construction of data delivery systems.

DB-STREAM provides a set of ready-made software modules whose behavior is essential function of data delivery process. In addition, DB-STREAM provides a dialect of functional language which makes it possible to describe data delivery process non-procedurally.

1 はじめに

これまで多くのＤＢシステムが構築されてきた。しかし、これらのＤＢシステムのほとんどは個別に開発されたため、ＤＢは単独システムでの利用しかされていなかった。近年、これらの既存ＤＢを複数のシステムで有効利用したいという要求が高まってきている。そこでまず考えられたのが、複数の既存ＤＢを１つの統合されたＤＢとして利用できるようにすることであった。しかしこれらの既存ＤＢの多くは分散していて、互いに異種であり、また自律的な運用を前提として設計されているため、統合化には次のような難しい問題がある [1]。

- それぞれのＤＢで採用されている多様なデータモデル（RDB、CDB、…）やデータの持つ意味を全て表現可能な共通データモデルをつくること。
- それぞれ異なった意味を持つ複数のＤＢのスキーマを、統一的に表現することのできる全域的スキーマを生成すること。
- 独自の同時実行制御方式を採用している、複数の自律的なＤＢシステムにまたがったグローバルトランザクション管理。

特に２番目のスキーマ統合の問題は、それぞれのＤＢのデータの意味や制約条件の差異、さらには将来の拡張等も考慮すると設計が難しい。

そのため、スキーマ統合あるいはグローバルトランザクション管理を必要としない有効活用の方法も検討され、いくつかの形態の相互運用システムが提案されている [2]。これらは、既存ＤＢシステムのデータを有効利用する現実的な解として期待されている。

このような相互運用システムを実現する上で問題となるのは、異なるＤＢのデータ間の**更新従属性** [4] が保証されないということである。更新従属性とは、複数のデータの間に、一方が更新されると他方もそれに合わせて更新しなければならないという更新従属関係が存在することである。異なるＤＢのデータ間に更新従属関係がある場合、相互運用システムでは一方のＤＢだけを自律的に更新することができるので、ＤＢ間でデータの矛盾が生じる可能性がある。

このようなデータの矛盾状態はそれらのＤＢを利用する業務に深刻な影響を与える。実際、企業では業務ＤＢシステム間で更新従属性のあるデータを持っていることが多いが、今まではこのようなデータの更新を各ＤＢシステム毎に個別に行なっていたため、ＤＢ間の矛盾が生じることがあり、大きな問題となっていた。

この問題に対処するためには、ＤＢの更新を自動的に他のＤＢに反映するような機構が必要である。この機構は、既存ＤＢに対する大きな改造は困難なことが多いので、ＤＢシステムの間に別システムとして構築されることが望ましい。

そこで、更新従属性の保証機能を持つシステムとして、更新されたＤＢから更新データの内容を受けとり、それを更新従属の関係にあるＤＢへ流通するようなシステムを考える。このシステムをＤＢ流通システムと呼び、その概念を図１に示す。

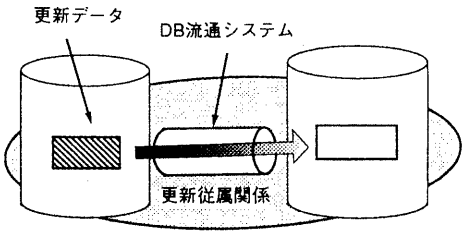

図 1: ＤＢ流通システムによる更新従属性の保証

本稿では、このようなＤＢ流通システムを簡易に構築するためのプラットフォームシステムとして提案した [8][9]、ＤＢ流通基本システム DB-STREAM のアーキテクチャと主要機能について述べる。

以下、2章では DB-STREAM のアーキテクチャについて、3章ではＤＢ流通における異種性の解決法と DB-STREAM での実現法について、4章では DB-STREAM の言語設計についてそれぞれ述べ、最後に DB-STREAM を実際の社内システムに適用した結果について評価する。

2 DB-STREAM のアーキテクチャ

ＤＢ流通システムを汎用プログラムで実現しようとすると、それぞれのＤＢシステムの組ごとにアプリケーションプログラムを開発しなければならないため、構築に時間と工数がかかる上にその後の変更や拡張が困難であるという問題がある。

この問題に対処するため、DB-STREAM では次のことを目標とした。

- 簡易かつ短期間でＤＢ流通システムを構築できること
- ＤＢの変更や増加などの際に、流通システムの変更や拡張が容易に行なえること。つまり、保守性が良いこと。

この目標へのアプローチとして、非手続き的な記述を用いて、簡易に流通システムを構築できるような方式を検討した。

まず、ＤＢ流通に必要な機能を分析しいくつかの原子的な機能に分解して、この原子的な機能の組合せでＤＢ流通を実現できるようにした。これをソフトウェアとして部品化したものを処理メソッドと呼ぶ。

また、この処理メソッドの組合せを簡易に記述できるようにするため、専用言語を開発した。処理メソッド

の組合せの記述をシナリオと呼び、シナリオを記述するために開発した言語をシナリオ定義言語と呼ぶ。このシナリオを DB-STREAM の実行制御部で解釈することにより、流通処理が実行されるようにした。

全体のアーキテクチャを図2に示す。

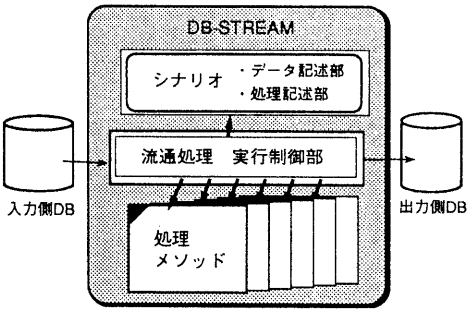

図 2: DB-STREAM のアーキテクチャ

これらの方式により、手続き的なプログラムを書くことなく流通システムの構築やカスタマイズを簡易に行なうことが可能となった。その概要について以下に述べる。

2.1 シナリオ

シナリオは、個々の流通システム毎に記述され、以下の内容を含む。

- データ記述：データ構造の記述
- 処理記述：処理メソッドの組合せとして記述

各処理に関する細かい機能の指定が必要な場合は、処理メソッドのパラメータとして記述するようにした。

シナリオ定義言語については4章に述べる。

2.2 処理メソッド

流通処理に必要な機能を分析して整理し、原子的な機能をソフトウェアの部品としたものを処理メソッドという。

DB-STREAM では、次のような機能を処理メソッドとして提供する。

集信メソッド 入力側のＤＢシステムから流通されたデータを受信する集信処理機能

変換メソッド ＤＢシステム間のさまざまな異種性を吸収するための変換処理機能

振り分けメソッド 変換したデータを必要な出力側ＤＢシステムに振り分ける振り分け処理機能

配信メソッド それぞれのデータを必要な出力側ＤＢシステムに送信する配信処理機能

これらの処理メソッドを自由に組み合わせることにより、さまざまなＤＢシステム間のデータ流通システムを構築できる。

3 異種性への対処

流通処理の中でもっとも重要な機能はＤＢ間のさまざまな異種性の吸収である。

ＤＢの異種性とは、それぞれのデータが異なる表現形式で管理されていることをいう。この章では、ＤＢ間のさまざまな異種性 [6] を、ＤＢ流通における変換という観点から次のように整理した。

- 名称レベル
- 値レベル
- 構造レベル
- 制約レベル

それぞれのレベル毎に、どのような異種性吸収機能が必要であるかについて述べる。

3.1 名称の異種性

それぞれのデータにどのような名前を付与しているかはＤＢシステムによって異なるため、同じデータに対し違う名前が付けられていたり、同じ名前の示すデータが全く異なっていたりする。

ＤＢ流通をするためには、どのデータがどれに対応するかが明示されている必要がある。DB-STREAM では、データ間の対応をシナリオ定義言語で記述する。

3.2 値の異種性

データ間の対応が取れていてもそのままの形式で流通することはできない。なぜなら、単位やコード体系、型など、データ値の表現形式がＤＢによって異なっているからである。

したがって、流通システムではデータ値の変換機能が必要である。

ＤＢ流通基本システムは汎用システムを目指しているので、その構成要素となるデータ変換の実現方式も汎用的である必要がある。そこでデータ変換処理の部品化を行ない、この組合せによりデータ変換処理を実現するアプローチをとった。

データ変換処理の部品としては、

- 機能的に互いに独立した
- 共用性の高い

ものを選択する必要がある。そこで、まず、変換処理に必要な部品をリストアップし、複合的な変換部品をより原子的な変換部品に分解した。そして原子的な部品

を整理・統合し、共用性の低い特殊な変換を除いたものをデータ変換基本部品として選択した。データ変換基本部品による変換機能としては、型・精度変換、文字列演算、数値演算のほか、ロジックで記述できない変換を行なうためのデータ値対応表を用いた変換等を提供する。このデータ変換基本部品を組合わせることにより、多様な変換に対応する。なお、特殊なデータ変換機能については、個別に作成し、組み込むことができるようにした。抽出した約３０個のデータ変換基本部品のうち、主なものを表1に示す。

表 1: データ変換基本部品 (抜粋)

種類	機能	関数名†
型・精度変換	整数 → 文字列 変換 文字列 → 文字列 変換 文字列 → 整数 変換	altic() altcc() altci()
数値演算	加、減、乗、除 剰余	+, −, ×, ÷ mod()
文字列演算	連結 部分列切り出し 右詰め	couple() substr() rpack()
対応表変換	データ値 → データ値	altcode()
無変換	代入	move()

†4章参照

3.3 構造の異種性

構造の異種性とは、データを表現している構造がＤＢによって異なることを言う。例えば、図3では、装置ＡとＢのデータを持つ２つのＤＢがあるが一方のＤＢではＡとＢの間のインターフェーサをＡの方に含め、他方のＤＢではＢの方に含めている。このように実体の管理単位が異なるなどの構造の異種性が考えられる。

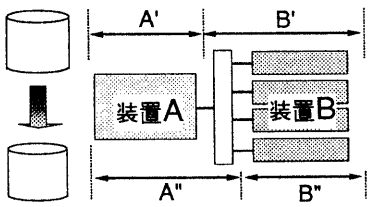

図 3: ＤＢによる実体の捉え方の違い

このような構造の異種性に対処するために必要な機能を以下の節で述べる。

3.3.1 流通ユニット

一般にＤＢの更新はあるまとまり毎に行なわれる。これは、それぞれのＤＢの整合性を保つためで、更新されるデータそのものだけでなく、制約チェックのための参照データが必要な場合もある。更新従属性の保証を行なう場合は、このまとまりを１つの単位として流通させなければならない。したがって、流通機構ではこの単位を扱う機能が必要であり、流通ユニットという概念を導入した。流通ユニットとは、ＤＢシステムにとっての意味あるまとまり、言いかえれば更新に必要なまとまり、を表現する単位である。

ＤＢの構造と整合性制約はＤＢシステム毎に異なっているので流通ユニットもＤＢシステム毎に異なる。１つのデータが複数のＤＢのデータと更新従属の関係にある場合には、入力された更新データのまとまりをそれぞれのＤＢシステムに合わせた流通ユニットに変換しなければならない。ユニット変換の例として、図3では、A″はA′とB′を変換して、B″はB′を変換して、流通される。

このようにＤＢ流通システムでは流通ユニットの組み替えが必要である。組み替えは、流通ユニットの分解と結合の組合せで実現できる。分解の場合は入力されたユニットの中から必要なデータを抽出する機能があれば良い。結合の場合は、流通機構特有の問題がある。

3.3.2 流通ユニットの結合

流通ユニットを結合するには、まず、流通されるユニットの中から結合すべき相手を選択しなければならない。この機能は、選択条件をユニットが持つデータ項目の値で指定することによって実現することができる。

しかし、この条件を満たすユニットがいくつあるかは流通システムにはわからない。なぜなら流通システムは、入力側ＤＢから流通されるデータを流通ユニットとして受けとるだけであり、入力側ＤＢの持つデータ全体を見ることができないからである。したがって流通システムはその結合処理をいつまで続ければよいか解らず、結合すべきユニットをずっと待ち続けることになる。

これを解決する一つの方法として範囲指定法を提案する [10]。これは結合すべきユニットの範囲を入力側ＤＢに提示してもらう方法である。流通システムはこの範囲でのみ結合を行なう。

DB-STREAM では、結合すべきユニットを１つのファイルとして流通することにより範囲指定法を実現している。

3.4 制約の異種性

ＤＢは独自の整合性制約を持っているので、ＤＢ更新時に整合性を保証するには値の制約チェックが必要である。

しかし、出力側ＤＢ更新時のチェック機能を流通システムで提供しようとした場合、次のような問題がある。

- 整合性制約の中には、出力側ＤＢを検索しないとチェックできないものが多い。(unique 制約や存在参照制約など)

- ＤＢシステムで既にチェック機構を持っていることが多く、冗長になる。

したがってDB-STREAMでは、制約チェック機能は提供しないこととした。

4　シナリオ定義言語設計

シナリオ定義言語を設計にあたり、シナリオの作成・保守の容易性を実現するため、できる限り非手続き的記述のみでシナリオ定義が行なえることを目標とした。

この章では、シナリオ定義言語のうち、異種性を解決するための変換に関する部分について設計方針および言語仕様概要を説明する。シナリオの中で、変換に関する部分は以下の２種である。

- 流通ユニット定義
- 流通ユニット変換定義

4.1　流通ユニット定義

まず、流通データの単位である流通ユニットを記述する流通ユニット定義について述べる。

流通ユニット定義言語に関しては、以下のことが実現できることが望ましい。

1. 業務ＤＢの主流であるRDBのデータ構造を容易に記述できること。
2. データのまとまりや他の主要データモデルの構造など、より複雑なデータ構造を記述できること。

上記を満たすため、フラットテーブル構造の簡易な記述をベース（図4の（ａ）参照）に、構造体、配列を記述可能（図4の（ｂ）参照）とした。

```
UNIT 流通ユニット名
  {  データ項目１   型;            ……（ａ）
     データ項目２   型;
     :
     構造体１
     {     データ項目３   型;     ……（ｂ）
           :
           :
     };
  };
```

図 4: 流通ユニット定義例

4.2　流通ユニット変換定義

次に、流通ユニット間の変換を記述する流通ユニット変換定義について述べる。

流通ユニット変換定義は、以下の２つの部分からなる。

- 変換元の流通ユニットの選択方法を記述する流通ユニット選択定義
- 選択された流通ユニットのデータ項目の変換方法を記述するデータ項目変換定義

4.2.1　流通ユニット選択部

１つ以上の変換元流通ユニットを選択あるいは、結合して、１つの変換先流通ユニットを生成する処理は、RDBにおける選択／結合処理に類似している。そこで、流通ユニット選択部では、RDB言語として広く普及しているSQLに準じたスタイルを採用した言語を考案した。図5に言語仕様を示す。

```
流通ユニット選択定義

  MAP     変換先流通ユニット名
  FROM    変換元流通ユニット名  [, ... ]
  WHERE   選択条件

    選択条件 ::= 式 比較オペレータ 式
    式       ::= 項目参照 | 定数 | 関数
    関数     ::= 関数名( 式 [, ...])
```

図 5: 流通ユニット変換定義言語仕様

4.2.2　データ項目変換部

データ項目変換は、もっとも記述量の多いこと、および記述難度の高いことから、記述の簡易性をもっとも強く必要とする。

変換を簡易に記述できるためには以下が要求される。

1. 変換の入出力関係を明確化するだけで、記述できること
2. 基本変換部品（3.2節参照）を容易に記述できること

1.に対処するため、関数型言語スタイルの言語を考案し（図6参照）、入出力関係を明確化するだけで変換の記述を可能とした。

2.については、3.2節で選択した基本変換部品を標準の関数として指定できるようにし、容易な記述を可能にした。（表1に各変換部品対応の関数名を示す。）

なお、基本変換部品の組合せでは表現し切れない変換を記述するための手段として、以下を提供することとした。

- 分岐構文（IF～THEN～ELSE）
- 利用者が個別に開発した関数を標準関数と同様に使用可能とするインターフェース

図6に言語仕様を示す。

```
データ項目変換定義

  { 変換先データ項目 ：＝  変換指定
    ：
    ：                                    } ;

    変換指定 ::= 関数
    関数 ::= 関数名( 式 [, ...])
    式   ::= 項目参照 ¦ 定数 ¦ 関数¦ ( 式 )
```

図 6: 流通ユニット変換定義言語仕様

5 適用性評価

電気通信分野のＤＢシステム間でのＤＢ流通を行なうプロトタイプを DB-STREAM を用いて作成しその有効性を評価した。

DB-STREAM では、簡易にＤＢ流通システムを構築できることが目標であった。その効果を確認するため、ＤＢ流通システムを汎用プログラミング言語で構築する場合と稼働を比較した。

１ポイントデータの結果ではあるが、DB-STREAM を利用した場合、約２／３の稼働が削減されることがわかった。

6 おわりに

複数のＤＢシステム間でのＤＢ流通システムを簡易に構築するためのプラットフォームソフトウェアとして、データ流通基本システム（DB-STREAM）を提案した。

DB-STREAM は、流通の機能をソフトウェア部品として実現した処理メソッドの提供と、処理メソッドの組合せを非手続き的に記述することによって流通処理の流れを定義するシナリオ方式とを特徴とする。

処理メソッドの検討においては、値の異種性の解決策として、約３０の基本的な変換部品を抽出した。また、シナリオの検討においては、変換処理を簡易に記述できる関数型の言語スタイルを考案した。

ＤＢ流通システムを汎用プログラム言語で構築する場合と、DB-STREAM を用いて構築する場合とを比較し、DB-STREAM の有効性を示した。

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, pp. 183-236, 1990.

[2] M.W.Bright, A.R.Hurson, and Simin H.Pakzad "A Taxonomy and Current Issues in Multidatabase Systems ", IEEE COMPUTER, pp.50-59, Dec. 1992.

[3] P.Drew, R.King, D.McLeod, M.Rusinkiewicz, A. Silberschatz, "Report of the Workshop on Semantic Heterogeneity and Interoperation in Multidatabase System", SIGMOD RECORD, Vol.22, Mo.3, Sep. 1993.

[4] Witold Litwin, Leo Mark, Nick Roussopoulos, "Interoperability of Multiple Autonomous Databases ", ACM Computing Surveys, Vol.22, No.3, pp.267-293, 1990.

[5] Marek Rusinkiewicz, Amit Sheth, George Karabatis, " Specifying Interdatabase Dependencies in a Multidatabase Environment ", IEEE COMPUTER, pp.46-53, Dec. 1991.

[6] Won Kim, Jungyun Seo, "Classifying Schematic and Data Heterogeneity in Multidatabase Systems", IEEE COMPUTER, pp.12-18, Dec. 1991.

[7] Sudha Ram, "Heterogeneous Distributed Database Systems ", IEEE COMPUTER, pp.7-10, Dec. 1991.

[8] 池田 哲夫 他, "ＤＢ流通の基本方式について" 情報処理学会第４６回全国大会, 1993.

[9] 星野 隆 他 "ＤＢ流通におけるデータ変換方式について" 情報処理学会第４６回全国大会,1993.

[10] 塩原 寿子 他 "ＤＢ流通における整合性の保証法" 情報処理学会第４８回全国大会,1994 （予定）.