統計数理 (2001)
第 49 巻 第 2 号 317–331

特集「統計環境におけるインターネットの利用」
［原著論文］

インターデータベース
——DandD インスタンスのエージェント化——

横内 大介[1] ・柴田 里程[1]

（受付 2001 年 4 月 12 日；改訂 2001 年 10 月 5 日）

要　　旨

インターネットを活用して，必要なデータを取得したり，逆に自分が取得したデータを公開したり，さらには解析した結果やモデルの公開あるいはその蓄積を円滑に行うためには，それなりの仕掛けが必要である． 本論文では，そのような仕掛けの一つとして， DandD (Data and Description) ルールにもとづいて生成された DandD インスタンスをユーザとデータの間を仲介する一つのエージェントとして活用するインターデータベース (InterDatabase) のアイディアを紹介するとともに，その中核的な役割を果たす DandD サーバの構築について述べる．

キーワード：データ，インターネット，データサイエンス, DandD, XML.

1. インターネット上のデータ

インターネットは，当初の科学研究のためのコミュニケーションの手段から発展して，現在ではきわめてポピュラーな情報獲得の手段となった．このような流れを受けて，統計学，あるいはもうすこし広く言えばデータサイエンスの対象としての，データの流通，蓄積のインフラストラクチャーとしてのインターネットの利用を真剣に検討する必要がある．もちろん，Web に代表される，インターネット上の媒体で交換される情報にはありとあらゆる形態があり，データサイエンスもそのすべてを対象とするわけにはいかない．そこで，本稿では，特に「ある目的にそって組織化されたデータ」あるいは「組織化するための素材となるデータ」に限定して議論することにする．

1.1 代表的なデータ取得法とその問題点

現在，インターネット上で広く用いられているデータ取得法としては FTP (File Transfer Protocol) と HTTP (HyperText Transfer Protocol) の 2 つのプロトコルを利用した取得法がある．まず，FTP は古くからあるファイル転送用のプロトコルである．データが単一のファイルで構成されていれば，どんな形式であってもそのままの形で取得できるという点では汎用であるが，その内容に関しては透明性 (Transparency) が低い．つまり，あらかじめそのファイルに含まれるデータの形式を熟知している必要がある．一方，HTTP は本来 HTML (HyperText Markup Language) で記述された文書ファイルを交換するためのプロトコルである．HTML はタグによって文章のマークアップ（印付け)を行うマークアップ言語であり，Web ページ記述

[1]慶應義塾大学 理工学部数理科学科：〒223–8522 神奈川県横浜市港北区日吉 3–14–1

の基本言語となっている．しかし，HTTP は現在では HTML 文書だけでなくそこに含まれる JPEG (Joint Photographic Expert Group)，GIF (Graphics Interchange Format) のような形式の画像データ， MOV (MOVie file format)，AVI (Audio Visual Interleaved) のような形式の映像データなども取得できるプロトコルに進化しており，一つの汎用なデータ取得方法を提供している．しかし，HTML のマークアップは，もっぱら表示の体裁を整えるためのものであって，タグで囲まれた内容がどんなものかを反映したものではない．たとえば`<h1>02</h1>` というマークアップは，もっとも大きなフォントで 02 を表示せよという指定であるが，そこには数字と文字の区別はない． また，HTML では，独自のタグを新設することも許されないので，`<month>02</month>`で，このタグの内容の 02 が月の 2 であることを表現することもできない．このように HTML は単に表示の体裁を指定するための言語であり，構造や属性を持ったデータを表現することはできない．

また HTTP に，CGI (Common Gateway Interface) などの汎用な Web サーバと外部プログラムのインターフェイスと JDBC (Java DataBase Connectivity) や ODBC (Open DataBase Connectivity) のようなデータベース用の API (Application Programming Interface) を組み合わせれば，SQL (Structured Query Language) によるリレーショナルデータベースへのネットワークアクセスも可能となるが，先の例で示したようなデータの様々な属性や背景情報はデータベースとは別に保存しておく必要がある (菊田 (1999))．

1.2 インターデータベース

HTML の欠点は，HTML の後継に位置する XML (eXtensible Markup Language) を用いればかなり克服できる (XML/SGML サロン (1998))．XML では自由にタグを新設でき，属性を付与することもできる．XML もマークアップ言語であることには変りはないので，本格的な言語に較べれば，不自由ではあるが，後で述べるように，この XML を用いて DandD ルールを実装した経験からすれば，データサイエンスの対象となるようなデータを組織化し，記述するのに十分役立つ．ただし，インターネット上に散在したさまざまな形式のデータの活用を考えると，すべてを XML 形式に書き換えるのは，大変な作業であり，現実的ではない．

そこで登場したのが，インターデータベース (InterDatabase) のアイディアである（柴田 (2001)）．この名前は，(株)数理システム社長の山下浩氏との議論の中から生まれたものであるが，「インター」には，多くのデータベースをまたぐという意味と，インターネット上のデータベース利用という二つの意味が込められている．このアイディアは極めてシンプルである．XML 文書の一つである DandD インスタンスの中に FTP によるファイルの形式でのデータ取得や SQL によるリレーショナルデータベースからのデータ取得に必要な情報などを埋め込んでしまえば，インターネット上に散在した異なる形式のデータベースも自由に利用できるようになるだろうというアイディアである．この原型はすでに数年前の DandD ルール制定の当初から，外部データの記述という形で存在していたが，それをインターネット上に拡張し，さまざまなデータベースの違いも吸収してしまおうという考え方がインターデータベースの新しいところである． 図 1 はその概念図である．

このように，DandD インスタンスにデータ取得方法の記述まで埋め込んでしまえば，ユーザはデータがどこにどんな形式で存在するかを特に意識する必要はなく，はじめから DandD インスタンス中にデータが存在していると考えてもよいことになる．言い換えれば， DandD インスタンスがユーザとデータの間を仲介する一種のエージェント (agent) として働くことになる． それだけでなく， DandD ルールは様々な属性や背景情報を記述することを目的として出発したルールであるので，無味乾燥な数値だけのデータを取得することにはならない．

また，現在のデータ取得の現場では様々な形のデータベースを用いており，それを構築し

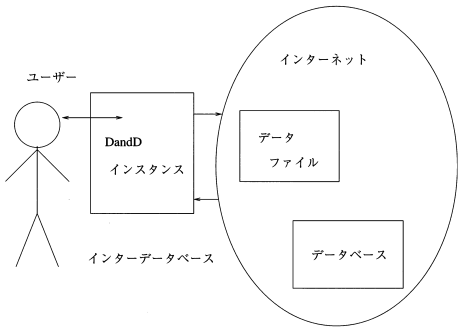

図 1. インターデータベース (InterDatabase).

なおすことはほぼ不可能であるが，インターデータベースの仕組みを用いれば基本的なデータベースに手を加える必要がない点も見逃せない．

2. DandD ルールとインターデータベース

インターデータベースのエージェントとして機能するのが，DandD インスタンスであり，それを規定するのが DandD ルールである．もちろん， DandD インスタンスは単なる XML 文書であり，それ自身がエージェントとして能動的に働くわけではない．エージェントが働くために必要な情報をすべて保持しているという意味で DandD インスタンスをエージェントと呼んでいる．まず DandD ルールの概略を説明しておこう．

2.1 DandD ルール

どんな複雑なデータも，基本的には，解きほぐせば単純な数値の並びであるベクトルの集まりとして表せる．そこで，DandD ルールでは，データはこのようなベクトルの集合であると規定している．その上で，それぞれのベクトルに属性を付与することで，データのさまざまな様相を表すことにしている．このようにさまざまな属性を付与したベクトルを，以降**データベクトル**と呼ぶことにする．したがって，カテゴリカルなデータもコーディング属性を付与することにして，数値の並びとして統一的に扱うことができる．さらに複数のデータベクトルを構造化することで複雑な関係を持つデータも扱う．一言でいえば，データの**実体 (entity)，属性 (attribute)，関係 (relation)** を分離して扱うことにより柔軟性と汎用性を確保するという原則を堅持していることになるが，その詳細は以下で与える具体例からお分かりになることと思う．

DandD ルールを実装するには，様々な手段が考えられるが， 数年前から始まった DandD プロジェクトでは現段階での実装手段として XML を用いている．人間にも計算機 にも可読であるだけでなく，XML が HTML の次世代言語として普及しつつあり，このような言語を利用することの利点を評価したからである．DandD ルール自体は，柴田 (2001) で解説されているようにどのような言語で実装するかにはよらず独立な存在であるので，どんな言語で実装するのが適当かは時代とともに変化するに違いない．実際 DandD ルールの前身である D&D ルール (Shibata and Sibuya (1991)) は，LISP に似た独自の記述言語で定義していた．

以下，DandD の構文定義ファイルである DandD.dtd に従って作成された XML 文書のことを **DandD インスタンス**と呼ぶことにする．

2.2 データベクトルとその実体

すでに述べたように DandD インスタンスのデータはすべてデータベクトルに分解されており，XML の言葉で言えば，要素 DataBody のいくつかの子要素 DataVector となっている．たとえば，もっとも簡単な例として

```
<DataVector Id="month" LongName="賞与月" Length="2" >6 12</DataVector>
```

を考えてみよう．ここで， Id は複数出現するデータベクトルを識別するための属性であり，LongName は説明的な正式な名前を与える属性，Length はデータの要素数を示す属性である．最後の属性 Length は抜け落ちている値がないかどうかのチェックのためと，サポートソフトウェアにとっては，あらかじめ要素数が分かっていたほうが処理しやすいという配慮からである．データベクトルには，これ以外にも数多くの属性を与えることができるが，それらについては必要になった時点で説明することにする．柴田 (2001) も参照されたい．

なお，この例ではタグで挟まれた 6 12 がこのデータベクトルの値の実体であり，一般的には空白を区切りとする自由欄形式の数字の並びである．XML では明示的に数字と文字を区別できないので，このような数値の並びも XML のインスタンスとしては，あくまでも文字列でしかない．しかし，DandD ルールではこの要素の値として「数値の指数表現と欠損値を表す NA 以外の並びは許さない」と規定してあるので，混乱する恐れはない．これは DandD ルールの XML による実装という観点からはセマンティックスに属する部分であり，XML では規定しきれない部分を補っている．もちろん，並びをさらにタグで区切って多くの子要素とする実装も可能ではあるが，NA 以外の文字を許さない以上，なんのメリットもない．

上の例は，データベクトルの値の実体が DandD インスタンス内に存在する場合であるが，ある程度規模の大きなデータや画像データなどを DandD インスタンス内に含めることは，煩雑さを増し，現実的ではないとの判断から，DandD ルールは当初から値が外部にあること，つまり「外部実体」も許している．これがまさにインターデータベースの基礎となるルールである． なお，もともと XML は画像のようなデータがインスタンス内部に存在することは許していない．それはリンクという形で外部実体として参照することになっている．しかし，このようなリンクという形式は，背景情報としては役立つが，他のデータと関連した形で扱う必要のある画像の場合には適当ではない．例えばスポーツデータのように映像から選手の手の軌道に関するデータを算出した場合，各数値に画像一枚一枚を対応させておくことが必要で，後のデータ解析に大いに役立つ．つまり，データベクトルの値として画像データも許すことが必要になる．

複数のデータベクトルがあるときは，その間の関係を記述する必要があり，それは要素 Data で構造化することになる．次の例は 2 つのデータベクトルを一つの関係形式として構造化し，関係づけた簡単な例である．

```
<Data>
<Relational Id="Height" LongName="身長の測定">
<Value RefId="name"/>
<Value RefId="height"/>
</Relational>
</Data>
<DataBody>
<DataVector Id="name" >Taro Jiro</DataVector>
<DataVector Id="height">173 166</DataVector>
```

```
</DataBody>
```

ここで，RefId 属性は DataVector の属性 Id への参照であり，2 つのデータベクトル，name と height を一つの関係形式 Height として構造化している．このほかにも配列形式や時系列，点過程などの特別な形式として構造化することもできるが，詳しくは柴田 (2001) を参照されたい．

このように DandD ルールでは，データベクトルの構造化をデータベクトルそれぞれがどのように定義されているかには依存しない形で行えるので，ネットワーク上の様々なところに分散し複雑に絡み合っている場合でも，そのことに注意を払う必要はない．つまり，構造に記述されるのはあくまでデータベクトルの参照であり，データベクトルの値の実体が異なるデータベース上やファイルとして存在してもよい．これが，DandD ルールを基礎にインターデータベースを構築するときの優位性である．

2.3 外部実体

データベクトルの値が空，つまりその実体がインスタンス内にはなく外部実体となっている場合，データ取得法の多様さを吸収し，あたかも，最初から DandD インスタンスの一部として存在したかのように扱う必要がある．そのためには，その処理に必要な情報を適切な属性として記述しておくことによって， DandD インスタンスをデータ収集のエージェントとして機能させればよい．

データベースに存在する場合

データベクトルの値が，外部のリレーショナルデータベースにあり，そのデータベースがネットワーク経由でアクセス可能な場合の例として

```
<DataVector Id="EYC1Y" LongName="一年満期の円建て債の金利"
Explanation="EY1YDef" Length="1606" Unit="%/年"
MissingType="not recorded" DatabaseNA="9999" DatabaseType="postgresql"
DatabaseServer="peanut.stat.math.keio.ac.jp"
SQLCommand="select eyc1y from kinri" DatabaseName="DandD"
DatabaseSecurity="UNSECURED" UserId="DandD"
AppletCodebase="http://www.stat.math.keio.ac.jp/DandD/server/download">
```

がある．DatabaseType，DatabaseServer， DatabaseName，DatabaseSecurity はデータベースを特定するための属性である．特に DatabaseSecurity はデータベースがユーザー ID やパスワードなどの認証を必要とする場合は SECURED，不必要な場合は UNSECURED を指定することでその違いを吸収している．データを取り出すための SQL 文は属性 SQLCommand に記述している．また， DatabaseNA はデータベースでの欠損値が 9999 で表されていることを示しており，外部実体の場合は欠損値の表現が一意的であることは期待できないので，これも重要な属性である．

なお，AppletCodebase は，これまでに DandD プロジェクトで開発された DandD ブラウザという DandD インスタンスを眺めるシステムを開発する段階で必要となった属性であり，データベースとの接続のための Java アプレットという JAVA プログラムを利用するためのものである．その意味では便宜的に導入された属性であり（小澤 他（2000)），後に紹介する DandD サーバが本格稼働すれば不要となる属性である．なお，DandD ブラウザについては DandD プロジェクトのホームページ (DandD Project (2001)) などを参照されたい．

ファイルとして存在する場合

XML では，何らかの処理を必要とする外部ファイルは「パース対象外外部エンティティ(entity)」として特殊扱いすることになっているが，この方法には柔軟性がなく，データベクトルの値としての利用は困難である．また，単純なテキストファイルであっても，そのままデータベクトルの値として埋め込むことはできない．記録の区切りが特定の文字であったり，ヘッダがついていればなおさらである．

そこでデータベースに存在する場合と同じように，その処理に必要な情報をデータベクトルの属性としてあらかじめ与えておき，それを反映した処理を行うよう記述する必要がある．まずファイルの所在は属性 `URL` で記述する．この属性にはよく知られた URL (Uniform Resource Locator) の形式で所在場所を記述するが，その先頭でその所在だけではなく，取得のための基本的な方法であるプロトコルも記述する．たとえば FTP なら `ftp://`，HTTP なら `http://` のようにはじめる．ただし， ローカルファイルを指定する場合，DandD ルールでは DandD インスタンスと同一のディレクトリにあるファイルあるいはディレクトリだけをその名前だけで指定することしか許さないことにしている．`//` 以降の記述がファイルシステムに依存するので，そのような曖昧さを避けるためである．またファイル名もファイルシステムに依存しないよう半角英数字のみで構成するといった注意も必要である．

これで，データファイルへの基本的なアクセス法は定まるとして，つぎにそのファイル中でどのようにデータが表現されているかが問題となる．理想的にはすべての可能性を尽くすことが望ましいが，これはある意味きりがない．そこで， DandD では音声，画像などのいわゆるメディアデータ以外のデータの場合には，データファイルとして，ヘッダのない自由欄形式の数値データだけを外部ファイルとして許すことにした．また各数値の区切り記号は属性 `Separator` で明示することにしている．もちろん，データベースにある場合と同様に，欠損値の表現の情報 `DatabaseNA` も重要な属性である．しかし， データファイルとしては基本的に DandD インスタンスの内部に保持する値と同等な数値の並びであることを仮定していることにはかわりがない．これ以外の場合は，一度リレーショナルデータベースに納めることによって，SQL を用いた取得法で取得することになる．もちろん，今後のプロジェクトの進行に伴って，ある程度サポートの範囲を広げることも考えられるが，圧縮ファイルの問題も含め，解決しなければならない課題は多い．

一方，音声データや画像データの場合は，データベクトルとして扱うとしても，他のデータベクトルとシンクロナイズした形で利用することになるので，その種類とフォーマットが分かれば十分である． したがって， DandD では，属性 `DataType` に `Sounds` を与えれば音声データ，`Slides` を与えれば動画像データであると判断し，属性 `Format` には `gif` や `jpeg` などでそのフォーマットを明示することにしている．一つの例が

```
<DataVector Id="Slides_of_swim0"
    LongName="水泳データの横から撮影した動画"
    DataType="Slides" URL="swim0" Format="gif" Length="57">
</DataVector>
```

である．この例は，水泳のフォームがスピードに与える影響を解析するためのデータで，手や足の位置やスピード，加速度といった観測値とシンクロナイズした形で，実際に上と横から撮影した動画像が重要な役割を果たす．このような動画像を背景情報として扱ったのでは，観測データとのシンクロナイズが切れてしまい，後の解析には役立たない．

このような，観測値とシンクロナイズした形での動画像は，いわば，データベクトルの要素それぞれに一枚の画像が対応しており，これまで述べてきたような数値の並びとしてのデータ

ベクトルとしては扱えない．そこで，DandD ルールでは，動画像を他のシンクロナイズする観測値それぞれに対応した静止画像つまりフレームの集まりとして扱い，それらの置かれた場所を，データベクトルの属性としてと同等な扱いができるようにしている．

より具体的には，画像データも DataBody 中の一つの DataVector として定義し，その DataType 属性に Slides を与えることによって，このようなシンクロナイズした動画像であることを示す．さらに属性 Format でその画像形式を示し，URL でフレームの集合の存在するディレクトリを示す．ただし，フレームの順序を明示するためには，フレームに順序の分かるような名前をつけておく必要がある．現在の実装では「番号. 拡張子」の形での名前をつけることにしている．拡張子は属性 Format に与えられたものと同じである．

ちなみに DandD ブラウザでは，このような形式で与えられたデータベクトルがあれば，他のデータと同期をとって表示するようになっている．具体的には関係形式の各記録(行)に番号をふり，各番号にカーソルを合わせると対応する画像に切り替わるような実装方法をとっている (横内 · 柴田 (2000)).

2.4 背景情報

DandD ルールはデータだけでなくその背景情報を要素 BackGround で記述でき，データと一体化するよう設計されている．とくに，データサイエンスの対象となるようなデータは文章だけではデータの意味を正確に伝えられない場合もあり，数式を使わなければならないことも多い．また，説明補助として図や表を使いたいこともあるだろう．しかし，残念ながら XML では数式，図や表のような文字型でないものを直接扱うことはできない．だからといって，図や表などのファイルを扱うために，その種類ごとに何らかのソフトウェアを用意しなければならないのではインターデータベースとしての価値が半減する．しかし，幸にしてデータベクトルの場合には柔軟性に欠けるとして採用しなかったパース対象外外部エンティティという XML の仕組みを利用することでフォーマットの異なるファイルも XML 文書の一部として扱うことができる．

パース対象外外部エンティティは，XML には文字としては扱えない画像などを XML の構文解析の際に解析の対象外として扱い，適当な外部システムに処理を依頼する仕組みで，たとえば，TEX のように XML 以外の特定の言語に従って記述された部分が memo1 にあれば

```
<!ENTITY memo1 SYSTEM "ファイルの URL" NDATA TEX>
<!NOTATION TEX SYSTEM "TeX を処理するプログラムの URL">
```

のように宣言した上で，

```
<Unparsed file= "memo1"/>
```

のように引用することによって，ENTITY に記述されているファイルを NOTATION に記述された外部のシステムが適当な処理をして，Unparsed と置き換えてくれることを期待している．もちろんこのような機能は，アプリケーションが自前で提供しなければならない機能である．ちなみに DandD ブラウザでは Java RMI を利用して実装している (横内 · 柴田 (2000)).

2.5 外部実体をもつ DandD インスタンスの例

ここで外部実体を利用した DandD インスタンスの一つを紹介しておこう．具体的に外部実体として各データベクトルはデータベースサーバである PostgreSQL に存在している．また，背景情報を記述する BackGround 要素の子要素である Introduction にはパース対象外外部エンティティである TEX ファイルが埋め込まれている．Introduction は背景情報を自由文形

式で記述するための要素だがこのファイルの内容には表が含まれており，XML では扱えないのでパース対象外外部エンティティを導入している．また，紙面の都合上，インスタンスの DataVector の一部を省略しているが，記述されているものとほぼ同じ内容である．

金融先物価格データ

```
<?xml version ="1.0" encoding="Shift_JIS"?>
<?xml-stylesheet type="text/xsl" href="DandD.xsl" ?>
<!DOCTYPE DandD  SYSTEM "DandD.dtd"[
<!ENTITY TeX SYSTEM
"http://www.stat.math.keio.ac.jp/DandD/GoldFutures.tex" NDATA TEX>
<!NOTATION TEX SYSTEM "http://www.stat.math.keio.ac.jp/DandD/TeX.class">
]>
<DandD>
<Title>金商品先物価格データ</Title>
<BackGround>
<Introduction><Unparsed file="TeX"/></Introduction>
</BackGround>
<Data>
<Relational Id = "Futures" LongName="金商品先物価格"
    Explanation="explanation">
<Value RefId="DealingTime"/>
<Value RefId="S_price"/>
<Value RefId="H_price"/>
<Value RefId="L_price"/>
<Value RefId="E_price"/>
<Value RefId="Amount"/>
<Value RefId="M_Amount"/>
<Value RefId="Delivery"/>
</Relational>
<Time Id = "DealingTime" LongName="立会日">
<Year  RefId="Dealing_Year"/>
<Month RefId="Dealing_Month"/>
<Day   RefId="Dealing_Day"/>
</Time>
</Data>
<DataBody>
<DataVector Id="Dealing_Year" LongName="立会日の年" Length="23069"
 DatabaseType="postgresql" DatabaseSecurity="UNSECURED"
 DatabaseServer="peanut.stat.math.keio.ac.jp" DatabaseName="showroom"
 SQLCommand="select DealingYear from gold" UserId="DandD"
 AppletCodebase="http://www.stat.math.keio.ac.jp/DandD/server/download">
</DataVector>
......
```

```
<DataVector Id="Delivery" LongName="当該限月" Length="23069"
 DatabaseType="postgresql" DatabaseSecurity="UNSECURED"
 DatabaseServer="peanut.stat.math.keio.ac.jp" DatabaseName="showroom"
 SQLCommand="select DeliveryMonth from gold" UserId="DandD"
 AppletCodebase="http://www.stat.math.keio.ac.jp/DandD/server/download">
</DataVector>
</DataBody>
</DandD>
```

3. DandD サーバ

先に述べたように，DandD インスタンスはそれだけでは単なる一つの XML 文書にすぎず，何らの働きもしない．DandD インスタンスがインターデータベースのエージェントとして実際に働くためには，それなりのソフトウェアが必要である． また DandD インスタンスを作成したり編集したり，さらには解析ソフトウェアに取り込むためにもさまざまなソフトウェアが必要となる．これまで DandD プロジェクトでは，そのようなソフトウェアの一部として DandD インスタンスを眺めるための DandD ブラウザと DandD インスタンスから解析ソフトウェアへのエキスポートシステムを開発してきた（伊藤 · 高際（1999), 宮澤 · 柴田（2000））．しかし，これらのソフトウェアは当初 DandD インスタンスが外部実体を許すといったインターデータベースとしての機能を想定して設計されたものではなく，研究を進めていくうちにその必要性に気づき，機能を追加したものである．したがって，これ自身十分な機能を持っているものの，いくつかの問題点も生じている．その問題点を解決して，開発を続けていくことも一つの選択肢ではあったが，プロジェクトが進行するにつれ，核となるサポートソフトウェアである DandD サーバを中心にサポートソフトウェアを再構築しなおした方がよいことになった．そこで，以下ではまずこれまでの開発の経緯を明らかにした上で，DandD サーバの構築の必要性とその役割について述べることにする．

3.1 これまでの DandD サポートソフトウェア開発

DandD ブラウザは XSL (eXtensible Stylesheet Language) と呼ばれる XML の書式を指定するスタイルシート言語を利用して開発された (W3C (2000a))．具体的には XSL の機能の一部である XSLT (XSL Transform) を利用して，XML 文書である DandD インスタンスを HTML 文書に変換し，既存の Web ブラウザで表示させている (W3C (1999))．そして，外部にあるデータベクトルや背景情報で必要なファイルを処理，取得する機能を Java アプレットと呼ばれる Web ブラウザ上で実行可能なプログラムと Java RMI (Remote Method Invocation)，さらに Java プログラムからリレーショナルデータベースにアクセスするための一つのインタフェース群である JDBC を利用することで，パース対象外外部エンティティの処理やリレーショナルデータベースサーバへのデータアクセスを実現している（横内 · 柴田（2000), 小澤 他（2000), SunMicroSystems（2000））．このようなソフトウェア構築法は，既存のソフトウェアを利用できるという点で，各サポートソフトウェアを基本から開発する場合に比べて時間と労力を節約でき，必ずしも悪いとはいえない．ただ，予想に反して現在でも，この XSL に完全に対応しているブラウザは存在せず，Microsoft 社の InternetExplorer5.0 (IE5) 以降のバージョンのものが XSLT に近い機能である MS-XSL を実装しているだけである．しかも現在の DandD ブラウザは MS-XSL が独自実装した機能を利用しており，事実上，IE5 上でしか動かない．また，Java アプレットが「Java アプレットは，自分自身がアップロードされていたサーバとしかアクセス

ができない」というセキュリティのための制限がかけられているために，単なるデータベースの接続に RMI という分散アプリケーション技術をわざわざ導入して実装した経緯がある．

DandD インスタンスから解析ソフトウェアへのエキスポートシステムでは Yacc/Lex を利用して独自の構文解析プログラムを実装し，インスタンスを解析してインスタンスの木構造を解析ソフトウェアに合わせた形で移植している．また，外部実体の扱いについては上述の RMI と JDBC を用いて解析ソフトウェア側から参照できるプログラムを作成し，その処理を実現している（宮澤 · 柴田（2000））．このようにエキスポートシステムはブラウザとは独立しており，パーサーもブラウザのそれとは別に実装している．エキスポートシステムはデータ解析をサポートするシステムであることはいうまでもないが，データ解析はまずデータをブラウズすることが重要であり，ブラウザと独立していることはマイナス要素である．

これらの主な原因は，1998 年に DandD プロジェクトを開始した時点では XML を扱う環境が整っていなかったため，既存の Web ブラウザを利用するしかなかったからである．しかし，現在では，XML を扱う環境，特に W3C での XML 関連の技術仕様が固まり，次々と正式な勧告として発表されており，様々なベンダによるプログラミング言語レベルでのサポート，特に Java による XML パーサーや DOM (Document Object Model) の実装が活発に行われている．幸い，これまでの開発で DandD ルールの XML による実装と，それに基づく DandD インスタンスの有効性は，これらのサポートソフトウェアによって十分検証できたので，このような環境の変化を受けて，すべてのソフトウェアを再構築することにした．

3.2 DandD サーバ開発のもたらすもの

サポートソフトウェアの再構築に際しては，さまざまなサポートソフトウェアを個々別々に開発する愚を避けるため，各ソフトウェアで共通な基本的な機能を洗い出して DandD サーバとして実現することにした．このことによって，様々なサポートソフトウェアを統一的に，また柔軟に開発することができる．図 2 がその概念図であり， DandD サーバが DandD インスタンスや外部データなどへのアクセスを集中的にサポートし，クライエントプログラムである

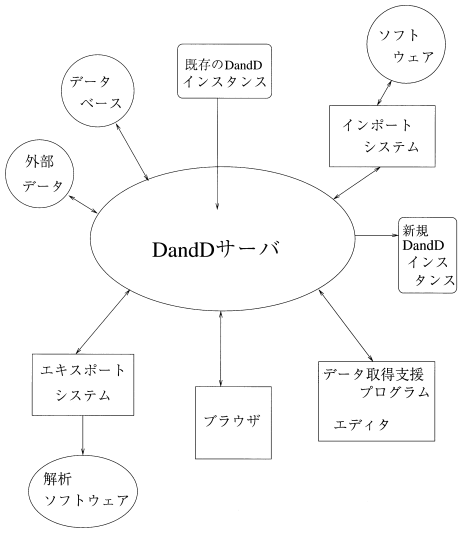

図 2. DandD サポートソフトウェア群における DandD サーバの位置づけ．

さまざまなサポートソフトウェアは，必要に応じてこのサーバへの通信を行って，必要な部分だけを取得すればよい．これは，特に大規模なデータが含まれる場合には極めて有利である．

DandD サーバはソケット接続によるスレッド間通信を利用した一種のデーモンプログラムであり，後で説明する DOM (Document Object Model) と呼ばれる文書を操作するためのモデルを利用したクライエントプログラムの要求に応じて必要な処理を行い，結果を返すサービスを提供する．ソケットは TCP/IP を利用するためのネットワーク用の API であり，ポートと呼ばれる TCP で用いるデータ送受信用のアドレスを利用して通信を行う．

まず，図 2 のブラウザを例にして DandD サーバの動きを簡単に説明しておこう．まず，ブラウザは DandD サーバへブラウジングの対象である DandD インスタンスのロードを要求する．それを受けて DandD サーバはそのインスタンスをロードし，その旨をブラウザに通知する．そして，ブラウザからの次の要求，例えばそのインスタンスに記述されているデータや補助情報の取得，データベースなどにある外部実体の取得などの要求を受けて，サーバはロードしたインスタンスの内容を参照し適当な処理を行いその結果をクライエントに返す．このような一連の動作を繰り返すことによって，クライエントであるブラウザは必要な情報を取得し結果を表示する．また，データ取得支援システムやエディタの場合でも，DOM は文書の内容を操作するためのモデルであるので上述のような DandD サーバ との通信を繰り返すことで，DandD インスタンスの新規作成や既存のインスタンスの改訂ができることは保証されている．

このようなデーモン方式の利点は，基本的な処理はサーバがすべて行うので，クライエントプログラムはソケットによる通信機能さえ準備すればよいということである．また，もし各ソフトウェアに共通な機能をライブラリとして実装した場合，どうしても言語やプラットフォームに依存してしまうが，デーモン方式ではクライエントプログラム側のプラットフォームや言語にも制限を受けないという利点がある．また，サーバにアクセスできさえすれば，携帯端末での利用も可能である．そのサーバもいくつかの拠点に存在しさえすればよい．

このように DandD サーバの開発には大きなメリットがあり，もちろん，既存のアプリケーションを利用しないことやソケットのプログラミングを理解する必要があるなど各サポートソフトウェアの開発に時間や労力がかかることは否めないが，それを差し引いても十分開発する価値がある．特に，インターデータベースの中核として DandD サーバは重要な位置を占めるものである．

3.3 DandD サーバの基本機能

DandD インスタンスは XML 文書でもあるので，基本的にはまず XML 文書の各要素に対する操作がサーバを介して自由に行えればよい．このためには，DOM (Document Object Model) (W3C (1998), (2000b)) と呼ばれるモデルが利用できる．ここで，DOM が「モデル」と呼ばれているのは，DOM が特定の言語に依存しない形で，文書の内容，構造，スタイルにアクセスできる操作を定義しているからである．実際，DOM は文書を DOM ツリーと呼ばれる一つの木構造でモデル化し，その各**ノード**（節）に対しての基本的な操作であるメソッド（method，操作)を幾つか定義している．

DOM の機能を実装することで， DandD サーバの DandD インスタンスを操作するための基本的機能はほとんど実装できることになるが，それでも DandD サーバの機能としては不十分な点がありメソッドの補足が必要となる．たとえば，DOM ではあくまでも XML 文書がメモリ上にロードされた状態での操作のみを定義しているが，文書自身をロードするメソッドは用意されていない．また，パース対象外外部エンティティを扱う `ENTITY` や `NOTATION` 要素の操作は読み取り専用のメソッドしか用意されていない．そこで DandD サーバには次のメソッドを用意することにした．

loadDocument：DandD インスタンスをロードするメソッド
flushDocument：DandD インスタンスをクライエントプログラムに渡すメソッド
appendEntity：DOCTYPE 要素に ENTITY 要素を追加するメソッド
removeEntity：DOCTYPE 要素の子要素から ENTITY 要素を削除するメソッド
appendNotation：DOCTYPE 要素に NOTATION 要素を追加するメソッド
removeNotation：DOCTYPE 要素の子要素から NOTATION を削除するメソッド

3.4 外部実体の処理

外部実体を扱うには，本来の DOM の機能を拡張し DandD に特有な処理を行えるようにする必要がある．具体的には，データベクトルの実体が DandD インスタンスの外部のファイル，あるいはデータベースに存在する場合や，パース対象外外部エンティティを扱うために本来の DOM で用意されたメソッドを一部拡張する必要がある．

データベクトルの外部実体のための拡張

クライエントプログラムが DataVector の子要素の取得を getChildNodes() メソッドなどで DandD サーバに依頼したとき，DataVector 要素が外部に実体を持っていて空である場合は子要素が存在しないので処理ができない．そこで，DandD サーバは自動的に DataVector の属性を参照して, データベクトルの実体を取得できるよう拡張する．その際必要ならば，外部ファイルの各数値の欄分割記号を表す属性 Separator を参照して空白に置き換える，DatabaseNA で定義された欠損値を表す値を NA に置き換えるなどの処理を行う．

パース対象外外部エンティティのための拡張

クライエントプログラムが getNodeValue() メソッドなどでパース対象外外部エンティティである Unparsed 要素の取得を DandD サーバに依頼した場合は，代わりに，その実体を取得してくる必要がある．具体的には DandD サーバは Unparsed の属性 file の値を手がかりに DandD インスタンス内の要素である ENTITY や NOTATION を自動的に参照し，実体を取得できるよう拡張する．

3.5 DandD サーバの実装

ネットワークとの親和性や実装の容易性から，DOM の API の実装は Java 言語上でもっとも盛んである．しかし，C や C++ 言語でも XML 用の API の開発がすすんでおり，たとえば，libxml は Gnome と呼ばれるウィンドウマネージャーでのアプリケーション開発用のライブラリであったが，バージョン 2 から Gnome だけに限らず，利用できるようになってきている (xmlsoft (2001))．したがって， DandD サーバをどの言語を基本として実装するかについては多くの選択肢がある．サーバはネットワーク上のいくつかの拠点で稼働すればよいので，移植の容易さよりも，その処理効率やメインテナンスの効率を重視する必要があり，C を基本とするサーバ開発の方が望ましい．しかし，プロトタイプの開発には Java 言語の方が開発効率がよいので，DandD プロジェクトとしては，まず Java 言語にもとづく DandD のプロトタイプから着手することにした．

Java 言語による DOM の実装として有名なものの一つとして XML Parser for Java があるが，これは IBM 東京基礎研究所で開発され，ソースコードも公開されている (alphaWorks (2001))．他にも JAXP (Java API for XML Parsing) のような実装もあるが，まだ完成の域には達していない．2001 年 3 月 の段階では，XML Parser for Java の最新版は xerces という名前に変わり，開発も HTTP のデーモンの開発で有名な Apache に移っているが，xerces は完全な日本語対応でないこともあり，あえて旧来の XML Parser for Java を利用して DandD サーバの中

心部分を構成することにしている．複数のクライエントプログラムからの要求を同時に処理する機能は java.Thread パッケージを利用することで実装することにし，DandD サーバとの接続ポートの番号は 11009 と定めた．外部実体の処理に関しては外部ファイルなら基本的には前述の java.net パッケージを利用して，データベースならば RMI のような分散処理システムではなく，java.net と JDBC の組み合わせによるソケット通信で直接必要なデータを取得できる．

4. むすび

以上， DandD インスタンスをエージェントとして利用したインターネット上のデータを活用する仕組みであるインターデータベースのアイディアとその実装について述べて来たが，DandD サーバを中心にした新しいサポートシステムはさまざまな可能性を秘めている．PDA (Personal Digital Assistance) などの携帯端末の発展は目覚しいものがあり，データ取得の現場で，携帯端末を使ってデータとその背景情報まで DandD インスタンスとして一体化してしまうことも考えられる．その場合，DandD サーバがネットワークを介したサービスを提供することによって，携帯端末の機動性をフルに生かすことができる．これは，コンサルティングの現場でも同様である．

データとその記述をデータ取得の段階から一体化しておけば，データの流れにそって発生する様々な問題をスムーズに解決できるという発想から始まった DandD プロジェクトも，インターネットの急速な進歩にともない， DandD を一種のメタデータとして位置づけ，データの実体はかならずしも抱え込まずに必要になった時点でネットワークを介して取得すればよいという考え方が現実的になってきた段階で，さらに，積極的にインターデータベースのエージェントとして活用するという方向に新たな展開を見せて来た．インターネットは分散化の一途であるが，本稿で紹介したインターデータベースは，データ自身が分散していたとしても，DandD インスタンスが DandD サーバのエージェントとして統一のとれた形で必要なデータを収集してくれるという,「分散の中の集積」を実現する新たな仕掛けである．

今や，データの扱いをコンピュータの専門家にまかせておけばよい時代ではない． 情報の洪水に飲みこまれないためにはどのようにデータを扱ったらよいかという，社会のだれもが関心をもっている問題に対して，データの専門家である統計学研究者がどれだけ積極的に関与していくのか，またできるのかその真価が問われているといってよい．本論文の背景には，そんな危機意識がある．

参 考 文 献

alphaWorks (2001). alphaWorksHomePage, http://www.jp.ibm.com/alphaWorks/

DandD Project (2001). DandD プロジェクト, http://www.stat.math.keio.ac.jp/DandD/index.ja.html

伊藤和子，高際 睦 (1999). DandD サポートシステム, 第 67 回日本統計学会講演報告集，151–152.

菊田英明 (1999).『実践 JDBC Java データベースプログラミング術』, オーム社，東京.

宮澤美紀，柴田里程 (2000). DandD から解析ソフトウェアへのエキスポートシステム, 第 68 回日本統計学会講演報告集，150–151.

小澤和弘，林 剛正，神保雅一 (2000). DandD におけるデータベースへのアクセス, 第 68 回日本統計学会講演報告集，112–113.

柴田里程 (2001).『データリテラシー』, 共立出版，東京.

Shibata, R. and Sibuya, M. (1991). User interface provided by D&D (Data and Description), *Bulletin*

of the International Statistical Institute, **1**, 1–25.
渋谷政昭，柴田里程 (1992). 『S によるデータ解析』, 共立出版，東京.
SunMicroSystems (2000). Java2 document, http://java.sun.com/products/jdk/1.2/ja/download-ja-docs.html
W3C (1998). DOMLevel1, http://www.w3c.org/TR/REC-DOM-Level-1/
W3C (1999). XSLT 1.0, http://www.w3c.org/TR/xslt/
W3C (2000a). XSL 1.0, http://www.w3c.org/TR/xsl/
W3C (2000b). DOMLevel2, http://www.w3c.org/TR/REC-DOM-Level-2-core/
XML/SGML サロン (1998). 『標準 XML 完全解説』, 技術評論社，東京.
xmlsoft (2001). The XML C library for Gnome, http://xmlsoft.org/
横内大介，柴田里程 (2000). DandD におけるデータファイルへのアクセス, 第 68 回日本統計学会講演報告集，148–149.

InterDatabase
—DandD Instance as an Agent on the Internet—

Daisuke Yokouchi

(Department of Mathematics, Keio University)

Ritei Shibata

(Department of Mathematics, Keio University)

We propose InterDatabase, a mechanism to make full use of data scattered over the Internet. InterDatabase also works as an interface to different types of relational databases on the Internet. We describe an implementation of InterDatabase through DandD (Data and Description) instance, in which all necessary information to access data is kept together with enough descriptions of the data itself. A key software is the DandD server, which activates DandD instance as an agent.

Key words: Data, Internet, DandD, InterDatabase, XML.