Vol.2009-DBS-149 No.12
2009/11/20

計算機と人との協調によるデータ管理のためのデータベース言語の提案

安 西 則 晃[†1]　森 嶋 厚 行[†1]

近年，エンドユーザによる計算機ネットワーク利用とデータ作成・発信の一般化により，様々な種類のデータがエンドユーザによって生成され，計算機に格納されることが日常的に行われている．これらのデータの管理は伝統的な DBMS を用いたアプローチでは難しい．本論文では新たなアプローチとして，データに係る人々も情報源とみなし，計算機と人との協調によりデータ管理やデータ集約型処理を行うためのデータベース言語を提案する．本論文の貢献は，提案する言語により，計算機と人の協調によるデータ管理を議論するためのアブストラクションを提供することである．
キーワード：ソーシャルデータマネジメント，データベース言語

A Database Language for Collaborative Data Management by Computers and People

Noriaki ANZAI[†1] and Atsuyuki MORISHIMA[†1]

Today, it is common that end users publish their data through computer networks, and that various kinds of user-generated data are stored in computers. Traditional DBMSs, however, do not have functions to effectively manage such user-generated data. This paper proposes a novel database language designed for collaborative data management by computers and people. The language consider people as information sources and is appropriate for the management of user-generated data and data-intensive applications. The main contribution of this paper is to provide an abstraction, which can be used as a language in which we can discuss problems in the management of user-generated data and the data-intensive applications.
Keywords：Social Data Management，Database Languages

1. はじめに

近年，エンドユーザによる計算機ネットワーク利用とデータ作成・発信の一般化により，様々な種類のデータがエンドユーザによって生成され，計算機に格納されるようになってきた．近年は，Web 上などで，これらの活動から価値を生み出すサービスが数多く提供されている[1)2)3)]．また，この様な Web 上でのサービスに限らず，エンドユーザが生成したデータを計算機に格納する事が日常的に行われている．例えば，デジタルカメラで撮られた大量の写真や，Excel ファイルとして置かれた名簿などをファイルサーバなどに置くことが日常となっている．したがって，これらのような，ユーザによって作成されたデータの管理は今日における重要な問題である．

しかし，それらのデータの管理を行うことは難しい．例えば，Excel ファイルに入った名簿などの各種データは最新の情報と異なることが多く，ファイルサーバに大量に格納される画像コンテンツはメタデータが付いていないということがしばしばある．Web 上での各種サービスなどでは，登録ユーザの相互評価などの仕組みをアプリケーション毎に作りこむことにより，データの品質などの管理の努力を個別に行わなければならない．

上に例を挙げたような，ユーザ生成データの管理の問題は，既存の DBMS が提供するデータ管理機能を用いるだけでは解決しない．したがって，新たなアプローチが必要とされる．

本論文では，このような，伝統的な DBMS が想定していないデータ管理やデータ集約型処理を支援するためのアプローチとして，閉世界仮説[4)] を前提としない DBMS とそのためのデータベース言語 Mixed-World Datalog (以下 MW-Datalog) を提案する．本言語は Datalog[5)] を拡張したものである．次は，アドレスリストを作成する MW-Datalog の問合せの一例である．

```
AddressList(name, address, phone) <- LabMember(name)
```

通常の Datalog と異なり，MW-Datalog は，右辺に現れない変数（`address`, `phone`）が左辺に現れることを許可する．そのような変数が現れると，システムはデータベース内のデータを探すのではなく，後述するように，適切と考えられる人に問合せを転送し，受け取った答えを利用して問合せ結果を生成する．

本論文の貢献と扱う問題．本論文の第一の貢献は，伝統的な DBMS がこれまで対象として

†1 筑波大学大学院 図書館情報メディア研究科
Grad. Sch. of Library, Information and Media Studies, Univ. of Tsukuba

こなかったデータ管理やデータ集約型処理の側面に焦点を当てた，新たな枠組みを提案することである．具体的には，ディスク上のデータには必ずしも完全なデータが格納されているという前提をおかず，その代わり，そのデータに係わる人々が情報をもっている可能性を考える．我々は，これらの人々も情報源と見なしてデータ管理やデータ集約型処理を行うというアプローチをとる．

上の例の様な問合せを考えたとき，問題となるのは次の 3 つである．(1) 計算機と人の協調によるデータ管理やデータ集約型処理を表現するための基本的な構成要素は何か．(2) 問合せを人に転送すべき場合，“適切な人” をどのように判定するか．(3) 彼らにどうインセンティブを与え，問合せの最終的な完成をどのようにサポートするか．本論文では，MW-Datalog におけるこれらの問題に対する対応について述べる．

本論文の第二の貢献は，MW-Datalog の提案を通じて，計算機と人が協調した処理を議論するためのアブストラクションを提供することである．言語のベースとして Datalog を用いた理由は，近年の計算機と人による協調処理はデータ集約型の処理が多いため，データベース言語の拡張として設計することが自然であると考えたことと，Datalog が理論的な解析に向いていることである．現在，計算機が苦手な処理を人間が行い，その結果を利用する Human Computation の研究[6][7] が行われているが，特定のアプリケーションや処理に特化された形の議論であって汎用のアブストラクションの議論ではない事が一般的である．MW-Datalog は，単に人の計算結果を利用可能なプログラミング言語ではなく，計算機と人が協力してデータ処理を行うための汎用のアブストラクションを提供する．このようなアブストラクションの応用として，次の 2 つが考えられる．(1) 計算機と人の協調によるデータ管理やデータ集約型処理の理論的側面を議論するための基盤として利用する．(2) サービスやコミュニティのデータ管理やデータ集約型処理を実装するための “プログラム言語” として利用する．具体的な利用方法としては，プロトタイピングツールとして，MW-Datalog のインタプリタでプログラムを実行する方法と，実サービスや商用サービスを開発するために，PHP や Java 等の汎用言語で書かれたプログラムの” テンプレート” を自動生成し，それを変更する形でサービスを実装する事が考えられる．

本論文の以降の構成は次のとおりである．2 章では関連研究を説明する．3 章では Mixed-World Datalog について説明する．4 章は MW-Datalog とゲーム理論の関係について簡単に触れる．5 章はまとめと今後の課題である．

2. 関 連 研 究

閉世界仮説を前提としないデータベース言語はこれまでも存在した．例えば，Open-world DATALOG (Relfun[8] のサブセット言語) は問合せに対して結果が空の場合，no ではなく unknown を返す言語である．しかし，その場合に結果を他の場所に探しに行くことはしない．

これまで，多くの情報統合プロジェクトが行われてきた[9][10] が，これらにおいては統合対象とする情報源とは全て計算機上の情報資源であり，人を情報源と見なした枠組みは我々の知る限り存在しない．

計算機から人に問合せを行う処理を実装可能な API や枠組みは数多く存在する．例えば，アクティブデータベース[11] を用いれば，データベースの状態に応じて人に問合せを行うためのプログラムが実装可能である．Amazon Mechanical Turk[12] は，人に仕事を依頼するための Web サービス API である．また，Google Docs[13] は，ウェブ上で文書ファイルやスプレッドシートなどのドキュメントを編集・管理する機能を提供する Web サービスであるが，このサービスの Google フォームという機能を使用すると，ユーザは簡単にテキストボックスやチェックボックスを配置した問合せフォームを作成でき，その URL をメールで送信することができる．しかし，本質的に，これらはあくまでも人に問合せを行うためのソフトウェアを実装可能な API であり，MW-Datalog のように，計算機と人の協調処理を実装するための新しいアブストラクションは提供していない．

1 章で述べたように，Human Computation の分野では，API レベルではなく，人による計算結果をどのように集めどのように集約するかという観点からの研究が行われている．しかし，本研究は次の点で異なる．(1) 個別の問題に対するアプローチの議論でなく，新たなアブストラクションを提供するデータ指向型言語の提案であること (2) 多くの Human Computation の取り組みと異なり MW-Datalog のスコープは人による計算結果を集約するという形の協調に限定していないこと．

3. Mixed-World Datalog

本章では，本論文で提案するデータベース言語（Mixed-World Datalog）について説明を行う．3.1 節で MW-Datalog の実行環境を説明する．3.2 節で MW-Datalog が扱う情報空間の定義言語について説明する．3.3 節で MW-Datalog で扱う特別なリレーションを説明する．3.4 節は利用例を通して MW-Datalog の言語仕様を説明する．3.5 節は MW-Datalog の実行過程で問合せを送信する対象となる人の発見方法について説明する．

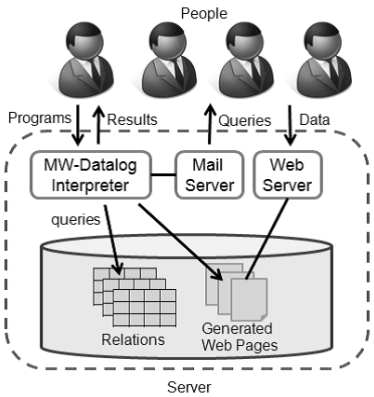

図 1 実行環境の概要図

図 2 Web フォームの例

```
1 CREATE INFOSPACE lab {
2    PERSON tada  NAME "只石" Email "tada@m.info";
3    PERSON anzai NAME "安西" Email "anzai@m.info";
4    PERSON ishii NAME "石井" Email "ishii@m.info";
5    PERSON ota   NAME "太田" Email "ota@m.info";
6    PERSON yama  NAME "山元" Email "yama@m.info";
7    SERVER mlab-server HOST "192.168.2.x";
8    GROUP isg {
9       MEMBER tada;
10      MEMBER anzai;
11      MEMBER yama;
12   };
13   GROUP wish {
14      MEMBER ishii;
15      MEMBER ota;
16   };
17 };
```

図 3 情報空間定義の例

3.1 実行環境

ここでは，MW-Datalog の実行環境について説明を行う．実行環境の概要図を図 1 に示す．まず，MW-Datalog は対象とする**情報空間**を持つ．そこには，計算機 (Server) と複数の人 (People) が含まれる．各人はデータを提供する人としての役割 (**Players** と呼ぶ) と，プログラムを投入する役割 (**Conductor** と呼ぶ) を持つことができる．Conductor は，MW-Datalog インタプリタにプログラムを入力し，結果を受け取る．MW-Datalog インタプリタは，プログラムの入力を受け取るとその処理を行う．処理過程で，必要に応じてディスク中のリレーションを参照する．また，必要に応じて，選ばれた Players に対してメールを通じて問合せを行う．メールには，問合せに対する入力を求めるフォームを持つ Web ページ (図 2) のアドレスが書かれている．Players はそのページにアクセスしてデータを入力し，MW-Datalog インタプリタに渡す．後述するように，問合せに応じてデータを入力した Players には**ポイント**が与えられる．

3.2 MW-Datalog における情報空間の定義

ここでは，MW-Datalog の実行が行われる情報空間を定義するための情報空間定義言語について説明する．例として，ある大学の研究室に所属する人と計算機の情報の定義記述を図 3 に示す．

情報空間定義は，`CREATE INFOSPACE` で始まり，まず情報空間の名前 (図 3 では `lab`) を与える．続いて，情報空間を構成する人と計算機の情報を与える．例では，`PERSON` で研究室に所属する人の情報を定義し，`SERVER` で MW-Datalog を処理する計算機の情報を定義している．各人は，`GROUP` によってグルーピングすることができる．この例では，所属する研究グループによってグループ化している．この他に，SERVER が管理するリレーションの定義を行う DDL があるが，これは通常の SQL と本質的に同じであるため省略する．

3.3 特別なリレーション

MW-Datalog インタプリタでは 3 つの特別なリレーションを保持する．ここでは，それらのリレーション `Member`，`Explain`，`Status` について説明する．

Member これは，情報空間に含まれる人に関する情報を持つリレーションである．情報空間定義言語で与えられた記述から自動生成される．例として，図 3 の情報空間に対する Member リレーションを図 4 に示す．属性として，各人の ID, Name, group, email, point を持つ．point の初期値は 0 となる．それ以外の値は，先に与えられた情報空間定義で指定された値が格納される．

Explain これは，各リレーションの属性の自然言語による説明を保持するリレーションである．図 5 に例を示す．本リレーションの値は次の 2 つの方法で入手される．(1) DDL によるリレーションスキーマ定義 (Create table) の各属性のコメントから自動生成する．(2) 後述する `explain` 命令を用いて明示的にタプルを挿入する．これが必要な理由は，本枠組みでは人に対して問合せを行う可能性があるため，その意味を伝えるこ

ID	Name	Group	Email	Point
tada	只石	isg	tada@m.info	0
anzai	安西	isg	anzai@m.info	0
ishii	石井	wish	ishii@m.info	0
ota	太田	wish	ota@m.info	0
yama	山元	isg	yama@m.info	0

図 4 Member リレーションの例

Relation	Attribute	Explanation
AddressList	Address	住所
AddressList	Tel	電話番号

図 5 Explain リレーションの例

Relation	Status
Member	closed
LabPapers	open
TitlesOfIshiiPapers	open

図 6 Status リレーションの例

とが必須であるからである．本リレーションに格納された情報は，図 2 のような Web フォームを作成する時に，その項目に何を入力すれば良いのかを人に伝えるために利用する．

Status これは，各リレーション毎に,「そこにデータが存在しないと言うことは現実世界でも存在しない」という事か (すなわち，閉世界仮説を適用して良いか) を指定する特別なリレーションである．図 6 に例を示す．適用して良い場合は closed となり，そうでない場合は open となる．この情報は，次の 2 つの方法により入手される．(1) 後述する `open` 命令および `close` 命令を用いて指定する．(2) 結合などで新しく作成されたリレーションの場合，次のルールを適用して自動生成する．すなわち，結合される全てのリレーションが closed である場合には新しく作成されたリレーションの状態も closed とし，そうでない場合は open とする．

本リレーションは，人に対する不必要な問合せを削減するために利用される．例えば，研究室名簿のリレーションを管理したいとし，研究室のメンバーは年度ごとに固定とする．その時，今年度の名簿情報をもつリレーションの内容の更新が終了しており新たなメンバーが追加されないことがわかっていれば，その情報は closed として指定される．したがって，名簿の情報を入手したい場合はそのリレーションを参照すれば良いことが保証されるため，他の人に照会をする必要が無いことがわかる．

3.4 MW-Datalog によるプログラミング

ここでは，簡単な例を用いながら，MW-Datalog によるプログラミングを説明し，MW-Datalog の構成要素を順に導入する．プログラム例によっては，実行過程で人 (Players) に問合せを送信する必要がある場合があるが，その時にどの人に問合せを送信するかは MW-Datalog インタプリタが決定する．この決定プロセスについては 3.5 節で説明することとし，各プログラム例では説明しない．

ID	Title	Author	Date
1	○○手法の提案	石井	2008-06-01
2	××システムの開発	石井	2008-12-01
3	△△手法の提案	只石	2008-09-01

図 7 LabPapers リレーション

3.4.1 最も単純なプログラム

最も単純なプログラムとして，先に説明した `Explain` と `Status` リレーションの更新のプログラムを示す．先頭の行番号は説明のためにつけたものであり，実際のプログラムでは記述しない．

```
1 theFirstProgram by anzai; {
2   explain(AddressList, Address, "自宅住所");
3   open(AddressList);
4 }
```

MW-Datalog プログラムは，プログラム名 (theFirstProgram) で始まり，続いて，`by` の後ろにこのプログラムを実行する Conductor 名 (anzai) を書く．Conductor 名は，情報空間定義で定義されたいずれかの人の ID でなくてはならない．最後に，MW-Datalog の**命令**と**ルール**の並びを囲むブロックで終わる．この例のプログラムでは，Explain リレーションを更新する `explain` 命令と，Status リレーションを更新する `open` 命令が実行される．手続き型プログラムと異なり，これらの間に実行順序の関係はない．

3.4.2 論文情報の問合せ

ここでは，anzai が，論文の情報が格納された図 7 の LabPapers リレーションに対して，石井さんが書いた論文のタイトルを問い合わせるプログラムを次に示す．

```
1 IshiiPapers by anzai; point(50) {
2     TitlesOfIshiiPapers(Title, Date) <- LabPapers(Title, Author, Date), Author = "石井";
3 }
```

本例のプログラムでは，1 行目でポイント指定を行っている．これは，プログラムの実行に関連して人に問合せが行った場合，それに答えた人に対して付与する**ポイント**を記述するものである．省略時にはデフォルトポイントが与えられるが，この例のように明示的に指定することもできる．さらに，本言語が持つ**集約オプション** (後述) によって，与えられるポイントを詳細にコントロールすることができる．与えられたポイントは，Member リレー

ション (図 4) の Point 属性に追加される．

Title	Date
○○手法の提案	2008-06-01
××システムの開発	2008-12-01

図 8 論文情報の問合せプログラムの実行結果の TitlesOfIshiiPapers リレーション

本例においては，ブロック内のルールは通常の Datalog ルールと同じである．しかし，実行時の処理は Status リレーション (図 6) で保持している closed/open の状態によって異なる．

closed の場合: 通常の Datalog と同じ処理を行い，結果を anzai に報告する．

open の場合: まずは closed の時と同じ処理を行う．次に，MW-Datalog インタプリタは，その結果が正しいかどうかの確認を依頼するための人を探し，その人にこれで良いかの確認の問合せを行う．問合せを行う人の発見方法については後述するが，本例では ishii(図 4 のリレーションより発見) にその問合せが行われる．問合せを受けた ishii は，結果の TitleOfIshiiPapers リレーションを確認し，正しければ Yes，正しくなければ No を返す．必要に応じて ishii は関連リレーションを修正する．ishii にはポイントが付与される．今回の場合は，付与するポイント数の指定がないため，デフォルトポイントが付与される．

3.4.3 住所録の作成

研究室の所属メンバーの情報が格納された図 4 の Member リレーションから，各人の連絡先などがかかれた住所録を作成するプログラムの例を次に示す．

```
1 CreateAddressList by anzai; expire("OneWeek") {
2     key(AddressList, ID);
3     AddressList(ID, Name, Address, Tel) <- Member(ID, Name);
4 }
```

1 行目の `expire` は，このプログラムの終了期限を明示的に記述する．一般に，MW-Datalog プログラムは，次のいずれかが最初に成立した時点で終了する．(1) プログラムの処理が全て終了したとき (2) 有効期限が過ぎたとき．したがって，本例のプログラムでは，全員から答えをもらっていなくても一週間で終了し，その時点の結果を anzai に報告する．終了期限が指定されない場合，デフォルト時間をもって終了期限となる．

2 行目の `key(AddressList, ID)` は，新しく作る AddressList リレーションのキー属性の

Name	Address	Tel
只石	つくば市春日	xxx-xxxx
安西	つくば市天王台	yyy-yyyy
石井	つくば市吾妻	zzz-zzzz
太田	つくば市天久保	xxy-yyzz
山元	つくば市竹園	xxx-yyyy

図 9 名簿作成プログラムの実行結果の AddressList リレーション

指定を行う．この `key` 命令でキーを指定すると，これまでキー定義が存在しなかったリレーションに対しては，そのままキー定義として保存される．キー定義が存在しないリレーションのデフォルト解釈は，全ての属性の組合せをキーとする事である．一方，既に存在するリレーションに対しては，このプログラムの実行時のみキーとして扱われる．ただし，指定された属性の値がそのリレーションの中でユニークでない場合にはエラーとなり，Conductor に報告される．

3 行目に注目すると，左辺の変数のうち `Address` と `Tel` が右辺に現れていない変数であることが分かる．この場合は，システムから人に問合せが送られる．このプログラムでは，名簿中の各人に対して Address と Tel 属性の値を求める問合せが送られる．問合せを受けた人は，図 2 のような Web フォームを介して Address と Tel の情報を返信し，ポイントを受け取る．

次の例は，年度毎に名簿の更新を行う例である．

```
1 CreateAddressList by anzai; expire("OneWeek") timing("EveryYear") {
2     key(AddressList, ID)
3     AddressList(ID, Name, Address, Tel)/update <- Member(ID, Name);
4 }
```

本プログラムでは，先のプログラムと 2 つの点が異なる．第一に異なる点は，1 行目の `timing(EveryYear)` である．これはプログラムの起動タイミングを指定するものであり，`EveryYear` は一年に一度起動するという意味である．この指示が省略された場合，デフォルト処理として，プログラムは投入後直ちに一度だけ実行される．

第二に異なる点は，3 行目に**更新オプション** (`/update`) がついている事である．通常の Datalog 処理系と異なり，MW-Datalog インタプリタでは作成されたデータは実体化してディスク上に格納され，明示的に削除を行わない限りディスク上に存在しつづける．したがって，同じ問合せを 2 度実行すると，2 回目には左辺のリレーションは既に存在していることになる．更新オプションは，そのような場合に左辺のリレーションを (キーが一致する

タプルの値を) 更新するために利用される．もし，更新オプションを付けずに実行した場合は，右辺の式 (この例では Member) に新たなタプルが現れた場合にのみ，その部分が左辺のリレーションに追加される．左辺のリレーションの既存データの自動的な更新は行われない．

3.4.4 分散管理された論文情報の一貫性管理

関連する情報を持つリレーションが複数の人によって別々に管理されている場合，しばしばそれらのデータの間には更新不整合が生じることがある．ここでは，Web 上で公開されている研究室の論文リストとある研究室メンバの論文リストをラッピングした 2 つのリレーション (`LabPaperList` と `AnzaiPaperList`) を対象に，それらの一貫性管理を行うためのプログラム例を示す．

```
1 PaperMaintenance by anzai; timing("EveryMonth") {
2     key(LabPaperList, title);
3     key(AnzaiPaperList, title);
4     AnzaiPaparList(title, conf, pages, year) <- LabPeperList(title, conf, pages, year);
5 }
```

4 行目では，左辺右辺のそれぞれのリレーションの実体が既に存在し，更新オプションが指定されていないので，左辺のリレーションの既存データの更新は行わない．しかしこの場合，既存データの更新は行われないが，左辺と右辺の値の整合性のチェックが行われる．具体的にはキーである title 属性が一致するタプル同士で，conf, pages, year が一致しているかチェックされる．もしも，一致しない論文が見つかった場合は，リレーション中のデータ間に矛盾があるとして，Conductor（anzai）に報告される．

3.4.5 ESP ゲーム

ここでは，ESP ゲームを単純化した変形版を MW-Datalog で記述する．ESP ゲームとは，人の力を借りて，画像コンテンツにタグ付けを行うゲームである．2 人のプレイヤーが同じ画像に対して，相手が入力するタグを想像しながらタグ付けを行い，タグが一致した時にスコアを獲得し，そのタグを画像のタグとして採用する．本例での ESP ゲームは情報空間の全員をプレイヤーとし，あるプレイヤーがつけたタグが他のプレイヤーと一致していた場合にスコアを与えるとする．

```
1 ESP_game by anzai; expire("OneWeek") timing("EveryWeek") exec_mode(Active) {
2     Metadata(File, Dir, Tag)/dupulicates <- inDir(File, Dir), Dir = "img";
3 }
```

1 行目の `exec_mode(Active)` は，プログラムの**実行モード**を指定する．これは，人との協調を行うための動作を指定するものである．実行モードには，Active, Normal, Passive の 3 種類がある．デフォルトの実行モードは Normal である．

Active 必要なとき，インタプリタは適切な人を探して問合せを転送する．適切な人が見つからない場合は全員に問合せをブロードキャストする．

Normal Active と同様に問合せを人に転送するが，適切な人が見つからない場合はブロードキャストせず，プログラムの Conductor に報告だけを行う．

Passive 問合せを人に転送せず，回答を入力するフォームをもつ Web ページ (群) だけを作成し，それらへのリンクを Conductor に報告する．

次に，ESP_game プログラムの本体について説明する．2 行目のルール中の `inDir` リレーションは，ファイルが作成された時にファイルの情報が格納されるリレーションであるとする．そのルールでは，左辺の変数のうち `Tag` が右辺に現れておらず，かつ更新オプションが指定されていないため，`inDir` リレーションで新しいファイルの追加 (タプルの追加) が確認された時のみ，人に Tag 属性の値を求める問合せが転送される．しかし，ルールやリレーションのデータ中に問合せをすべき人に関するヒントが全くないため，MW-Datalog インタプリタは転送すべき人を絞り込めない．したがって，実行モードが Active であることから，情報空間内の全員に問合せをブロードキャストする．

ブロードキャストした結果，MW-Datalog のインタプリタは複数の人から返信を受ける．Metadata リレーションはキー指定が無いため，一人あたり複数の Tag が返される．これらをどのように格納するかを指定するオプションが**集約オプション**である．このプログラムでは，3 行目の Metadata リレーションの後ろに記述されている `/dupulicates` がそれにあたる．`dupulicates` は，複数の人から返信された値 (一般に一人で複数の値) の中で，複数の人から共通して返された値のみを格納するオプションである．したがって，本プログラムは，イメージにつける Tag を求める問合せを全員にブロードキャストし，二人以上から重複して返信された値だけを Tag 属性に格納する．

MW-Datalog が提供する集約オプションには，`intersection` (全ての人が挙げた値), `union`(挙げられた全ての値)，`dupulicates` (2 人以上の異なる人で重複している値), `majority`（多数決で最も多い値），`unique` (他の人と重複していない値)，`identifical`(全員があげた値 (の集合) が完全に一致した場合のみその値 (の集合) を返す)，`last` (最も遅く与えられた値)，`first`(最も早く与えられた値)，がある．集約オプションを指定すると，答えが採用された人にだけポイントが与えられる．例えば，`/duplicates` オプションの場合は，他の人と同じデータを返した人だけがポイントを受け取る．したがって，後述するように，集約オプションは DW-Datalog プログラムが提供する**ゲーム**の構成要素になる．

3.4.6 価格投稿サービスのプロトタイピング

現在，利用者からの情報を集めて価値を提供する様々な Web サービスが提供されているが，毎日特価[2] はそのようなサービスの一つである．これは，サイトの会員が折込チラシの中の特売品の情報を入力すると対価としてお金が得られるサービスである．このサービスの簡略版の MW-Datalog によるプログラムを次に示す．ただし，ShopList リレーションには店舗情報が格納されているものとする．

```
1 PriceWebService by anzai; point(50) exec_mode(Passive) {
2     PriceList(shopname, address, name, price, date) <- ShopList(shopname, address);
3 }
```

このプログラムでは，1 行目で実行モードが **Passive** に指定されている．このため，MW-Datalog インタプリタは人に問合せの転送を行わず，値を入力する Web フォームを作成して，それらのアドレスを anzai に通知する．また，ポイントとして 50pt が指定されているため，この Web フォームを使って特売品の情報を入力したユーザには 50pt が付与される．

3.5 問合せ転送先の Player の決定

これまで説明を省略していた，問合せ転送先の Player の決定手順について説明する．MW-Datalog プログラムの実行時に Player に問合せの転送が必要な場合は次の 2 つである．(A) ルールの右辺に現れない変数が左辺に現れた時．(B) 答えのリレーションが open であり結果を確認する必要があるとき．

それぞれの場合の決定手順を次に説明する．

（A）ルールの右辺に現れない変数が左辺に現れた時

この場合は，次の手順によって，タプル単位で問合せを転送する Player を求める．

(1) ルールの右辺を計算し，その結果の各タプルの値を調べ，Member リレーションの属性 (Name や Email，Group 属性) の値とマッチする値があれば，その人に決定する．

(2) (1) で見つからない場合，右辺の計算結果のリレーションと，右辺に現れるリレーションと外部キーによって関連がある別のリレーションを結合する．その結果の各タプルの値を調べ，(1) と同様の方法で Player を決定する．

(3) (1)(2) で見つからない場合は，Active モードの場合は Member リレーション内の全員に対して問合せをブロードキャストする．Normal モードの場合は Conductor にその旨報告する．

（B）リレーションが open であり結果を確認する必要がある時

この場合は，次の手順によって，結果を確認するための問合せを転送する Player を決定する．

B の戦略＼ A の戦略	値 1	値 2
値 1	(1,1)	(0,0)
値 2	(0,0)	(1,1)

図 10 duplicates が提供するゲーム

(1) ルールの右辺に現れる定数を調べて，Member リレーションの Name 属性等と一致するものがあればその人に決定する．

(2) (1) で決まらない場合，右辺の計算を行った結果のデータを調べて Member リレーションの Name 属性と最も多くマッチするものを見つけ．その人に決定する．

(3) (1) (2) で決まらない場合，右辺の計算結果と，右辺に現れるリレーションと外部キーによって関連がある別のリレーションを結合する．その結果の値を調べ，(2) と同様の方法で Player を決定する．

(4) 以上で見つからなかった場合は，Active モードの場合は Member リレーション内の全員に対して問合せをブロードキャストする．Normal モードの場合は Conductor にその旨報告する．

以上の手順で適切な人が Player として選ばれない場合もあり得るため，問合せを受けた人が，その問合せを他の人に転送する仕組みを用意 (図 2 下) することにより，人の知識を活用し，最終的に最もデータに関連があるであろう人に問合せが転送されやすいように工夫している．

4. MW-Datalog とゲーム理論

MW-Datalog プログラムの振る舞いの解析はゲーム理論と関連が深い．例えば，Player が 2 名 (A,B) で，返信する値の候補が値 1, 値 2 だけの場合，集約オプションの `/duplicates` は図 10 で示すゲームを提供する．したがって，このゲームでは，他の人と同じ値を返す事に対するインセンティブが働くことになる．MW-Datalog が提供するアブストラクションとゲーム理論を用いてより複雑な人と計算機の協調処理を解析することは重要な課題の一つであるといえる．これにより，例えば Web サービスのプロトタイピングを行うだけでなく，振る舞いの解析を行うことによってより適切なサービスの開発支援ツールとして利用できる可能性がある．

5. まとめと今後の課題

本論文では，従来型の DBMS とデータベース言語では扱いが難しい問題を対象とし，計算

機と人との協調によりデータ管理やデータ集約型処理を行うためのデータベース言語を提案した．今後の課題としては，問合せを転送する人の決定方法のより詳細な検証，MW-Datalog プログラムによって人を効率的に動かすためのインセンティブ構造の検証，実務で使われるようなより複雑なサービスを記述可能にするための言語の拡張，計算機と人の協調によるデータ管理やデータ集約型処理の理論的側面のより深い議論などがある．

謝　　辞

ゼミなどでコメントいただきました筑波大学大学院図書館情報メディア研究科の杉本重雄教授，阪口哲男准教授，永森光晴講師に感謝いたします．本研究の一部は科学研究費補助金若手研究 (B)(#20700076) による．

参 考 文 献

1) ”Wikipedia”. http://wikipedia.org/, (accessed 2009-10-19).
2) NAVIT Co.,Ltd. ”毎日特売”. http://www.navit-tokubai.jp/, (accessed 2009-10-19).
3) GOGOLabs, Inc. ”ガソリン価格比較サイト”. http://www.gogo.gs/, (accessed 2009-10-19).
4) Serge, A.; Richard, H.; Victor, V. Foundations of Databases., Addison-Wesley Publishing Company, 1995.
5) Hector Garcia-Molina; Jeffrey D Ullman; Jennifer Wisdom. Database Systems: the Complete Book., Prentice Hall, 2002, p.463-502.
6) Luis von Ahn; Laura Dabbish. ”Laura Dabbish: Designing games with a purpose”. Communications of the ACM Vol.51(8). 2008. p.58-67.
7) Arase, Yuki; Xie, Xing; Duan, Manni; Hara, Takahiro; Nishio, Shojiro. ”A Game Based Approach to Assign Geographical Relevance to Web Images”. WWW2009. 2009.
8) Harold Boley. A Tight, Practical Integration of Relations and Functions., Springer, 1999, p.23-24.
9) Chen Li; et al. ”Capability Based Mediation in TSIMMIS”. ACM SIGMOD Record Vol.27(2), 1998, p.564-566.
10) Mary Tork Roth; et al. ”The Garlic Project”. ACM SIGMOD Record Vol.25(2), 1996, p.557.
11) Norman W. Paton; Oscar Diaz. ”Active database systems”. ACM Computing Surveys (CSUR), Vol. 31, No. 1, p. 63-103, 1999.
12) Amazon.com, Inc. ”Amazon Mechanical Turk”. https://www.mturk.com/mturk/welcome, (accessed 2009-10-19).
13) Google, Inc. ”Google Docs”. http://docs.google.com/, (accessed 2009-10-19).

付　　録

Mixed-World Datalog の構文

```
<Program> ::= <ProgName> <Conductor> [<Point>] [<Expire>] [<Timing>] [<Execute] '{'
                {<Open>|<Close>|<Explain>|<Key>|<ForeignKey>}* {<Rules>}*
            '}'

// プログラムの作成者の指定
<Conductor> ::= 'by' <UserID> ';'
// 付与するポイントの指定
<Point> ::= 'point(' <Number> ')'
// プログラムの終了期限の指定
<Expire> ::= 'expire(' <String> ')'
// プログラムを実行するタイミングの指定
<Timing> ::= 'timing(' <String> ')'
// プログラムの実行モードを指定
<Execute> ::= 'exec_mode(' <ExecMode> ')'
<ExecMode> ::= 'Active' | 'Normal' | 'Passive'
// リレーションの状態を指定
<Open>  ::= 'open(' <RelName> ');'
<Close> ::= 'close(' <RelName> ');'
// リレーションの属性の説明を記述
<Explain> ::= 'explain(' <RelName> ',' <Column> ',' <String> ');'
// リレーションの主キー・外部キーの指定
<Key> ::= 'key(' <RelName> {',' <Column>}+ ');'
<ForeignKey> ::= 'foreignKey(' <RelName> '.' <Column> ',' <RelName> '.' <Column> ');'
// 問合せ記述
<Rules> ::= <Rule> {'\n' <Rule>}* ';'
<Rule>  ::= <Head> '<-' <Body>
<Head>  ::= <Relation> ['/' ['update,'] <AggOption>]
<Body>  ::= <Atom> {',' <Atom>}*
<Atom>  ::= <Relation> | <Expression>
<Relation> ::= <RelName> '(' <Column> [':'<Label>] {','<Column> [':'<Label>]}* ')'
<Expression> ::= <Term> <Op> <Term>
<Term>       ::= <Name> | <Number> | <String>
<AggOption>  ::= 'intersection' | 'union' | 'dupulicates' | 'majority' |
                 'unique' | 'identifical' | 'last' | 'first'
<ProgName> ::= <Name>
<UserID>   ::= <Name>
<RelName>  ::= <Name>
<Column>   ::= <Name>
<Label>    ::= <Name>

// トークンの定義
<String> ::= '"' 文字列 '"'（全角文字列）
<Number> ::= [-]{0-9}*  (数値)
<Name>   ::= A-Za-z0-9_（半角英数文字列）
<Op>     ::= '>' | '<' | '<=' | '>=' | '=' | '!='
```