RMX におけるメール本文生成機能の実現

出口　萌子†　　遠山元道††

† 慶應義塾大学理工学研究科情報工学専修　〒 223–8522 横浜市港北区日吉 3–14–1
E-mail: †moe@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし　RMX とは, ルールをあらかじめ定義しておくことで, データベースから動的に宛先を取得し, メール配信を行うシステムである. 動的に配送先の取得を行うことができる本システムの利点を利用し, 本論文では, メール本文でも同様にデータベースから動的に値を取得し, その値を用いてメール本文へ埋め込みを行う, 本文生成機能を提案する. この機能では, 宛先に対応するデータを非スカラデータとしてテーブルから取得し, その情報を小さな単位で本文に埋め込む. その際, メールの形式はテキスト形式だけでなく, リッチテキスト形式についても可能とする. また送信先アドレス取得と同時に, 埋め込み文書の情報を一度に取得できるアルゴリズムを生成する.

キーワード　RMX, メール, メーリングリスト

1. はじめに

Rule-based e-Mail eXchange(RMX) は管理者があらかじめ定義したルールに基づき, データベースから動的にメールを転送するメール転送エージェントである [1, 2]. 従来のメールアドレスはアカウント名とドメイン名から構成される 2 次元的なものであるのに対し, RMX では配送ルール, パラメータ, ドメイン名の 3 つから構成されている, 3 次元的なものである. これらの 3 種類の構成要素を組み合わせることによって, データベースからアドレスの集合を得ることができ, 従来のメールと比べ広がりを持つ.

現在の RMX では, データベースや Web からデータを取得し, 送信者へデータのみを返すことは可能であるが, 本文中に取得データ以外のデータを埋め込むことはできない. また得られた宛先に対し, 本文中にそれぞれの名前などの値を, 単体で埋め込むことは容易だが, 複数の値の埋め込みを行う際は冗長な記述が必要になってしまい, 本文生成効率が落ちてしまう.

そういった背景から, 本論文では, 配送ルールと同様に, 本文生成用のルールを定義することで, 宛先に対応する情報を動的に取得し, 本文に対して, それぞれの情報を埋め込むことが可能な, 本文生成機能を提案する. 同時に, 配送ルールと本文生成ルール間の SQL の実行を 1 度にするための合成アルゴリズムを生成する.

本論文の構成は以下の通りである. まず, 2 章で RMX の概要について述べる. 次に 3 章で関連研究について述べる. 4 章では RMX での本文生成機能の現状について, 5 章では提案する本文生成機能について述べ, 6 章では予備実験を行う. 最後に 7 章で結論を述べる.

2. RMX

Rule-based e-Mail eXchange(RMX) は慶應義塾大学が提案している電子メールの配信方式である. RMX 形式のメールアドレスを用いることで, 配送先のアドレスを動的に取得しメールを転送することが可能である. RMX のメールアドレスの記述には, 関数形式と自然形式の 2 種類が存在し, 以下に記述する形式は, それらのうち関数形式の記述方式である.

＜*RMX* のメール配信先指定 (関数形式) ＞:=
＜配送ルール名＞{＜パラメータ＞}@＜サブドメイン＞.＜ドメイン＞

次に, 関数形式を用いたアドレスの例を示す.

student{2012}@keio.krmx.jp

この例では, サブドメインは *keio* であるので, *keio.properties* という設定ファイルを参照する. また *student* の部分は *2012* をパラメータに持つ配送ルールを参照する. この際, 配送ルールは設定ファイル内にパラメータ付き SQL 文で記述する. これらの情報を用いることで, データベースから問い合わせ処理を行い, 配送先のアドレスを取得し, それらのアドレスに対してメールの配送を行う. この RMX の処理の流れを図示したものが以下の図 1 である. この図からも分かるように, *student{2012}@keio.krmx.jp* にメールを送信すると, 2012 年に入学した学生に対してメールが送信されることとなる.

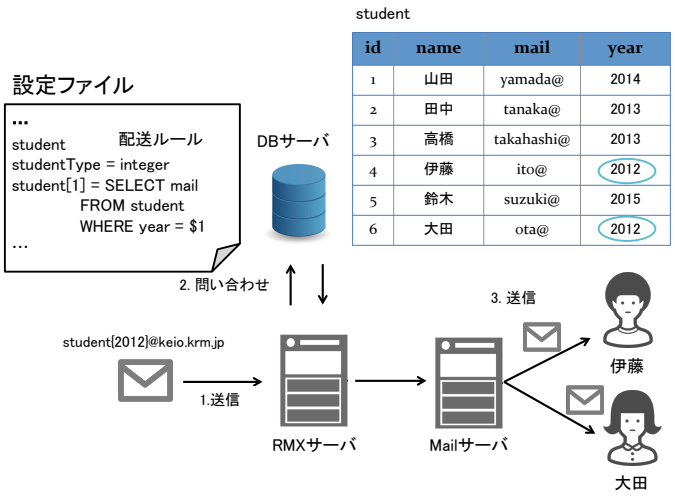


図 1　RMX におけるメール配信の流れ

以下, 2.1 章でサブドメイン設定ファイル, 2.2 章で配送ルールについての詳細, 2.3 章で複数の配送ルールの組み合わせについて概説する. 説明に際し, 図 2 の例題を使用する. この図は大学が保持している学生の情報で, student テーブルは学生の個人情報を, adviser テーブルは担当教員の個人情報を, faculty テーブルでは学部の一覧を, 示す. それぞれのテーブルはエンティティとなっている. student テーブルの adviser 属性は adviser テーブルの主キーに対応する外部キーであり, student テーブルの faculty 属性は faculty テーブルの主キーに対応する外部キーである.

student

id	name	mail	faculty	year	adviser
1	山田	yamada@	10	2014	112
2	田中	tanaka@	30	2013	150
3	高橋	takahashi@	40	2013	176
4	伊藤	ito@	60	2012	112
5	鈴木	suzuki@	10	2015	115
6	大田	ota@	60	2012	197
7	安藤	ando@	30	2012	155
...	...	...	...	...	...

adviser

id	name	ename	mail
112	阿部	abe	abe@
115	石井	ishii	ishi@
150	上田	ueda	ueda@
155	遠藤	endo	endo@
176	岡村	okamura	okamura@
197	加藤	kato	kato@
...	...	...	...

faculty

id	name	ename	campus
10	法	law	mita
30	経営	business	hiyoshi
40	医	medical	yagami
60	理	science	mita

図 2　大学の所持する学生情報

2.1　設定ファイル

設定ファイルの中では, データベースの管理と, 配送ルールの設定を行う. 設定ファイルはサブドメイン名*.properties* という名前を使用する. 例えば, *keio* というサブドメインに対しては, *keio.properties* という設定ファイルを用意しておく. *keio* という名前のサブドメインでメールが送られてくると *keio.properties* という設定ファイルに記述されているデータベースが参照され, そこに書かれているルールが使用される.

データベース管理のために設定ファイルに記述する項目は以下のとおりである.

- dbDriver = データベースドライバ
- dbUrl = 接続データベースの URL
- dbId = データベースのユーザ ID
- dbPassword = ユーザ ID に対するパスワード

2.2　配送ルール

配送ルールとは, 配送範囲記述と, それに基づき送信先のメールアドレスを取得するクエリを関連付けるルールである. 配送ルールは設定ファイルの中に以下のように定義する.

```
配送ルール名
Type:パラメータの型
query: 送信先メールアドレスを得るためのクエリ
```

クエリは SQL によって記述する. 配送ルール名に対応するクエリの *'$i'* で示されるプレースホルダーにパラメータを挿入し, データベース問い合わせを行うことで送信先メールアドレスの集合を取得する. このような配送ルールを用いることで, ユーザは簡潔な記述で配送範囲を記述することができる. 以下に配送ルールの定義の例を示す. 例で使用する *krmx.jp* は慶應義塾大学で使用さている RMX のためのドメインである.

keio.properties

```
student
studentType = integer
student[1] = SELECT mail FROM student WHERE year = $1
student[2] = SELECT mail FROM student
             WHERE year >= $1 AND year <= $2

adviser
adviserType = String
adviser[1] = SELECT s.mail FROM student s, adviser a
             WHERE s.adviser = a.id AND a.ename = $1

faculty
facultyType = String
faculty[1] = SELECT s.mail FROM student s, faculty f
             WHERE s.faculty = f.id AND f.ename = $1

campus
campusType = String
campus[1] = SELECT s.mail FROM student s, faculty f
             WHERE s.faculty = f.id AND f.campus = $1
```

これらの例は図 2 に対応している. 配送ルールのクエリによって図 2 のような既存のデータベースに対して, 配送ルールを定義できる. この例では student ルールは全ての入学年度の学生に対してそれぞれにメールを送信することが可能である. もし *student{2012}@keio.krmx.jp* にメールが送信されると, RMX サーバは *keio.properties* の設定ファイルから student ルールを参照する. このメールアドレスは 1 つのパラメータで構成されているので, 一番上の配送ルールが呼び出され, *$1* の部分にパラメータの *2012* が挿入される. そして, 取得されたアドレスへメールが送信される. この例では 2012 年に入学した学生 (伊藤, 大田, 安藤) へメールが送信される. また *student{2012-2013}@keio.krmx.jp* に対してメールを送信すると, 2 つの引数をもつ配送ルールが呼び出され, 2012 年から 2013 年に入学した学生へメールが送信される. このように RMX を用いることで, 既存のデータベースから様々なグループの人へメールを送信できる.

2.3　複数の配送ルールの組み合わせ (関数形式)

RMX では, 配送ルール間に, 演算子を使用することによって, ユーザに対してより詳細な配送範囲の指定を可能にする.

2.3.1 積 集 合

Syntax: $<name_1>\{<par_1>\}. \ \ldots \ .<name_n>\{<par_n>\}@<subdomain>.<domain>$

Semantics: $name_1\ (par_1) \ \cap \ \ldots \ \cap \ name_n\ (par_n)$

“.” は，複数の配送ルールを使用したい時に用いられる演算子である．指定された複数の配送ルールの結果の積集合を取り，その積集合によって得られた結果に対して配送を行う．

例:student{2012-2013}.faculty{science}@keio.krmx.jp

2.3.2 和 集 合

Syntax: $<name_1>\{<par_1>\}+ \ \ldots \ +<name_n>\{<par_n>\}@<subdomain>.<domain>$

Semantics: $name_1\ (par_1) \ \cup \ \ldots \ \cup \ name_n\ (par_n)$

“+” は，論理和によって複数の配送ルールを指定する際に用いられる．各パラメータをそれぞれの配送ルールのクエリに代入し，得られた結果の和集合から配送を行う．

また，配送ルール間の積集合をとる “.” と和集合をとる “+” が同時に使用された場合，積集合をとる “.” の方が優先順位が高いものとする．

例：student{2012-2013}.faculty{science}+
student{2015}.faculty{science}@keio.krmx.jp

例えばこの様なメールアドレスの場合，student ルールと faculty ルールの積集合をそれぞれとった後，2 つの和集合をとる．つまりこの例の場合は，2012 年，2013 年，2015 年に理学部を入学した学生にメールを送信するという結果になる．

これらの演算子は組み合わせて使用することもでき，ユーザが配送範囲を指定する際の手助けを行う．メールアドレスは以下の形式を取る．

$ListOpe := -$
$UnionOpe := +$
$RuleOpe := . \mid + \mid -$
$Arg :=$ string | integer
$Para := Arg \mid Arg\ ListOpe\ Para$
$ParaList := Para \mid Para\ UnionOpe\ ParaList$
$Exp :=$ rule { $ParaList$ }
$ExpList := Exp \mid Exp\ RuleOpe\ ExpList$
$Address := ExpList$ @ subdomain . domain

3. 関 連 研 究

RMX のメーリングリストとしての使用方法に焦点を当て，RMX と同様にメーリングリストシステムとして挙動する，既存のメール配信システムとの比較を行う．今回比較対象とするのは blaynmail [3], Benchmark Email [4], Acces Mail [5] の 3 つである．これらは全て，企業がマーケティングメールを送信する手助けをするために開発されたシステムで，これ以外にも多くのメール配信システムが存在する．以下の表 1 は RMX と 3 つのメール配信システムの機能を比較した表である．

表 1 RMX と既存のメール配信システムとの比較

	RMX	blaynmail	Benchmark Email	Acces Mail
グループの登録	X	O	O	O
既存の DB の利用	O	X	X	X
複数テーブルからの情報抽出	O	X	X	X
専用エディタ	X	O	O	O
差しこみコード数	-	∞	28 項目	7 項目

この表から RMX の大きな特徴として以下の 3 点があげられる．

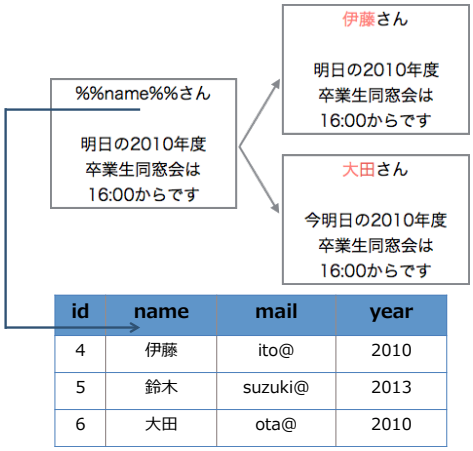
（1） グループごとの宛先の登録が不要
（2） 既存のデータベースに組み込み可能
（3） RDB を用いて広い範囲での宛先指定が可能

1 点目は，メールアドレスからその場で複数の送信先を決定するのでグループごとの宛先の登録が不要であるという点である．Benchmark Email では，送信するグループごとに登録が必要だが，RMX ではグループを個々で登録する必要はない．2 点目は，既存のデータベースに組み込み可能であり，メール配信のために，改めて宛先の情報を登録する必要が無い点である．blaynmail ではメール送信のために宛先の登録が必要だが，RMX は既存のデータベースの使用が可能である．3 点目は，SQL で記述できる範囲であれば，RMX では配送ルールを用いて，どのような組み合わせの宛先も抽出することができる点である．既存のメール配信システムでは，宛先はあらかじめグループごとで決められているか，同じテーブルの範囲内のみでの抽出しか行うことができない．こういった点から，RMX は，既存のシステムと比べ，複数のテーブルをまたぐような広い範囲での宛先指定が可能であるといえる．これは既存のメール配信システムと大きく異なる点であると考えられる．

また，メール配信システムの中でも特に使用される，差しこみコードについて既存のメール配信システム内での比較を行った．差しこみコードとは，メールマガジン等を配信する際に，メールの中に名前やアドレスなどを差し込み，個人的にメールを送信しているように見せかけるシステムである (図 3)．

既存のメール配信システムでは，図 3 のような差し込みを行う機能が存在しており，簡単にメール本文に差しこみ文書を記述できる．しかし，差しこみ内容は，メールアドレスが格納されている個人情報テーブルに記述されている属性のみである．つまり図 3 において，既存のメール配信システム内で差し込み可能な属性は id, name, mail, year の 4 つである．複数のテーブルからのデータの差し込みは行うことができない．また，差し込みできる数は配信システムによって制限されている．例えば Acces Mail では差し込みコード数は 7 項目のみしか記述することができない．

そこで，本論文では先ほどの RMX の特徴と既存のメール配



id	name	mail	year
4	伊藤	ito@	2010
5	鈴木	suzuki@	2013
6	大田	ota@	2010

図 3　差しこみコード一例

信システムの現状から, 配送先だけでなく, 個々の情報を他のテーブルから取得して, メール本文に内容として埋め込むような, 宛先に合わせたメール本文の生成機能を提案する. これは例えば, ある大学が学生の成績をメールで配信する際などに, データを打ち込むことなく, データベースから情報を取得することで, メール本文を生成し, メールを送信することができるようにするといったものである.

4.　RMX での内容生成機能の現状

RMX では, 得られた宛先に対し, 本文に個々の内容を生成して送信する内容生成機能が既に存在する [6]. しかしそれらにはいくつかの問題点がある. 以下にそれぞれの特徴と問題点を示す.

4.1　生成ルール

一つ目の内容生成機能として, データベースや WebAPI から個人情報や天気などを取得し, それらの文字列をメールを送信者に返す機能がある. 配送ルールに類似した形式で, ルール記述を行うことで, 情報取得先からデータを取得し, そのデータが本文に含まれるメールを生成, 送信者に返送する.

しかし, これはデータのみを返すもので, データを単体で使用することはできない. 従って, 取得データ以外をメール本文に記述することはできない.

4.2　挿入ルール

もう 1 つは, メールアドレスと名前, メールアドレスと学年など, メールアドレスと 1 つの属性をセットにしておき, 宛先のメールアドレスを通して埋め込みを行うものである.

```
name
nameType = String
name[0] = select s.mail, s.name from student s;
```

こちらも, 上記のように配送ルールと類似した形式で, “挿入ルール名 [0]” を定義をしておくことで, アドレスと埋め込み内容をセットにし, < *value* 挿入ルール > を用いて埋め込みを行う (図 4).

しかしこれは, メールアドレス 1 度の記述に対して, 1 つのスカラ情報のみを返すだけである. 従って, 複数の属性を埋め込むためには, セットを何度も記述しその情報を取得してこなければならず, 冗長であるといえる.

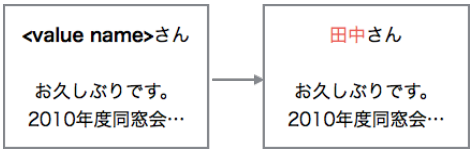


図 4　先行研究での本文生成機能

こういった現状を踏まえ, 本論文では, 宛先に対応するデータをスカラデータのみでなく, 非スカラデータも含めたデータとして, テーブルから複数取得し, その情報を小さな単位で本文に埋め込むことができるような内容生成機能の実装を提案する. 情報は複数のテーブルから取得可能であり, 情報の取得形式としては JSON や XML, CSV を用いる. 埋め込まれるデータとしては, テキストだけでなく, テーブルや画像, 計算式等も扱う. また, 一度に複数の値を取得できるように, ルールの SQL 同士を合成するアルゴリズムも提案する.

5.　本文生成機能

提案する本文生成機能も, 既存手法のようにルールを用いて行う. そこで, 既存の本文生成機能と, 提案する本文生成機能を比較するため, 提案手法で記述するルールを本文生成ルールと名づけて説明を行う.

図 5 は提案する本文生成機能の流れを一例として図示したものである. *faculty{science}@keio.krmx.jp* というメールを送信すると, サブドメイン *keio* からプロパティファイルが読み込まれ, 理工学部の学生の宛先が取得される. その宛先と, 送信されたメールの本文から, 本文生成ルールを参照することで, 個人にあった情報を取得し, その情報が本文に埋め込まれ, それぞれにメールが送信される.

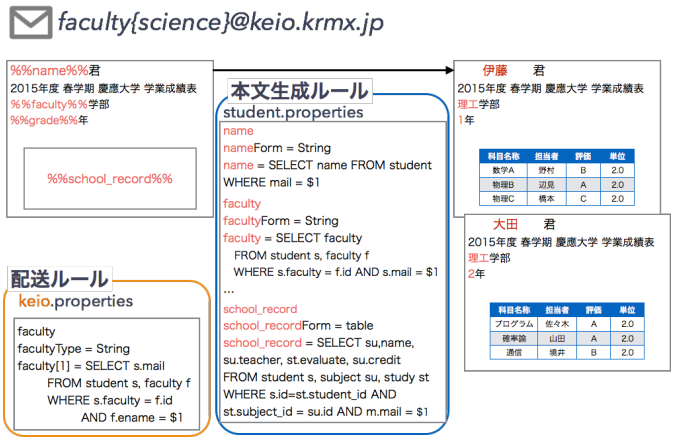


図 5　本文生成機能の流れ

本文生成ルールの記述方法は以下のとおりである.

```
本文生成ルール名
Form:出力形式
query: 挿入値取得のために送信先アドレスとの対応付けを行うクエリ
```

この記述方式によって, 宛先に対応するデータを取得し, 出力形式に合わせて出力を行う. 出力形式が *table* のような場合は, プレーンテキストメールでなくリッチテキスト形式のメールを用いる. これによってメールユーザに対し, 見やすい形でのメールを提供する.

次に, 図 5 の例を用いて, 従来の RMX に本文生成機能を追加することを仮定する. 宛先アドレスの取得のために, 配送ルールの SQL が 1 度実行された後, それぞれのメールに対し, 本文生成ルールが実行されるので, 1 通に対し, 4 回の SQL が実行されることとなる. 従って, 理工学部の学生が 100 人いることを仮定すると, 合計で, $1 + 100 \times 4 = 401$ 回 の SQL が実行される. しかし, SQL の実行回数の増加とともに, 情報取得時間, メール生成速度も増加することが予想される. また, 挿入ルールの問題点である, “複数のデータ挿入に対して, SQL の実行が冗長” という点をカバーできていない. そこで, 配送ルールと本文生成ルールの SQL を 1 つの SQL に合成し, 1 度の SQL の実行によって情報の取得が可能なアルゴリズムを生成する. このアルゴリズムによって合成した SQL の実行結果を, メモリ上に保管しておき, メール作成時にそのメモリ上の値を参照することで, SQL の実行回数を 1 度に抑え, 情報取得の速度, 結果としてメール生成の速度の向上を測る.

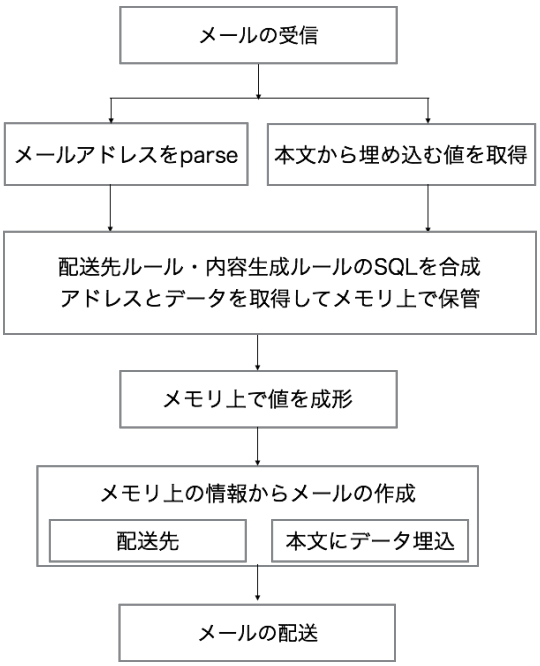


図 6　提案する本文生成機能

6. 予 備 実 験

提案する, 配送ルールと本文生成ルールの SQL の合成による, 情報取得の速度を SQL の合成を行わない方式と比較した. 図 5 の例を用いて実験を行った. 実際に student テーブル 1000 タプル, subject テーブル 5000, study テーブル 50000 タプルを用意し, 理工学部の学生 x 人に対して, 本文生成ルールを用いて成績情報のメールを送信する場合を仮定した. 情報取得は, 合成なしの場合, 配送ルール (faculty ルール) の実行時間と内容生成ルール (name, faculty, grade, schoo_record ルール) をそれぞれ Prepared Stetement を用いて実行し, 時間の合計を計算した. 合成ありの場合については, 配送ルールと内容生成ルールを手動で合成し, 実行時間を測った. 結果を図 7 に示す.

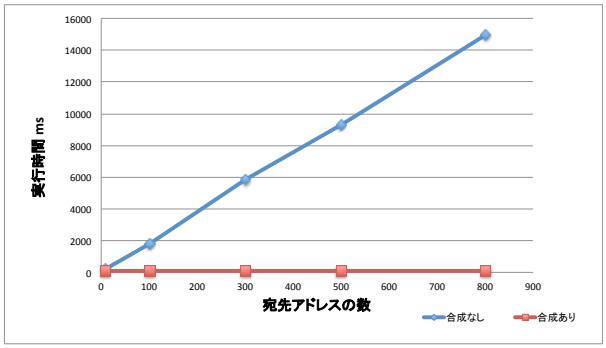


図 7　SQL の合成による実行速度の比較

上記の図から, 合成なしで $4x + 1$ 回の SQL を実行する場合は, 宛先アドレスの数に応じて実行時間が線形的に増えていることがわかる. それに場合に比べ, SQL を合成し, 1 度にまとめて実行を行った場合は宛先アドレスの数が増えても実行時間はあまり変動しない. このことから, SQL の合成の重要性が理解できる. 今回は簡単な例を用いているので, SQL の合成も手動で容易に導くことができた. しかし, 今後扱うであろう, 複雑な合成を行うため, 配送ルール内でのルールの合成・本文生成ルール内でのルールの合成それぞれに対して, 抽象化・一般化をし, それらの組み合わせ方についても考慮しながら合成をおこなう.

7. お わ り に

本論文では, RMX の特徴を活かし, RMX 内で, 複数のデータベースから宛先に対応する個人情報を動的に取得し, メールの本文内に小さな単位で複数埋め込むことが可能な本文生成機能を提案する. またそのために, ルール間の SQL を合成するアルゴリズムを考案する必要があることを示した.

文　　献

[1] 高畑 理, 藤沼 健太郎, 石橋 玲, 遠山 元道. “Magic Mirror Mailing:個人情報データベースを利用する柔軟なメイル配送システム”, 情報処理学会データベースシステム研究報告 Pages:123-128 July 2001.

[2] Kim Hanki, Sang-Gyu Shin, Motomichi Toyama. “A Rule-Based Mailing System for an Organization”, International Workshop on INformation Processing over Evolving Networks, June 2006

[3] “blaynmail”, http://blaynmail.jp/

[4] “Benchmark Email”, http://www.benchmarkemail.com/jp/

[5] “Acces Mail”, http://www.accessmail.jp/

[6] 松澤 慧, 北圃 達也, 小船井 寛, 遠山 元道. “RMX における受信者別メール本文生成機能および本文参照型アドレスの実装”, DEIM2013