データベースシステム　124－12
情報学基礎　62－12
（２００１．　５．２２）

データベース言語処理システムの自動生成に関する一検討

松浦 雅彦　　宝珍 輝尚　　都司 達夫

福井大学 工学部 情報・メディア工学科
〒 910-8507 福井市文京３丁目９－１

hochin@pear.fuis.fukui-u.ac.jp

本論文では，応用分野独自のデータベース管理システムの構築際の負荷の軽減を目的として，スキーマ管理部とデータベース言語処理部からなるデータベース言語処理システムをユーザからの仕様書により自動生成する手法について述べる．仕様書には，データモデル仕様書，データベース言語仕様書，言語制約仕様書，演算仕様書がある．データモデル仕様書は，データモデルの構造を指定する仕様書である．データベース言語仕様書は，データベース言語の言語文法を指定する仕様書である．言語制約仕様書は，データベース言語の文法制約を指定する仕様書である．演算仕様書は，演算と言語の関係を指定する仕様書である．プロトタイプシステムを構築し，関係モデル・ＳＱＬの一部に対するデータベース言語処理システムを生成させたところ，１０９行の仕様書で，Ｃ言語で約３７００行のプログラムを生成することができた．これにより，データベース言語処理システムの構築負荷が軽減できることを明らかにした．

On the Automatic Generation of Database Language Processing System

Masahiko MATSUURA　　Teruhisa HOCHIN　　Tatsuo TSUJI
Dept. of Information Science, Faculty of Eng., Fukui University
3-9-1, Bunkyo, Fukui-shi, Fukui 910-8507
hochin@pear.fuis.fukui-u.ac.jp

This paper presents a method of generating a database language processing system in order to decrease the burden of constructing an application specific database management system. A database language processing system consists of a parser, a semantic checker, a plan generator, an optimizer, a plan executor, and a schema manager. These modules are tried to be generated from the specifications. Specifications are a data model, a database language, a restriction, and an operation specification. A prototype generation system is constructed. This prototype system can generate about 3700 lines in C by taking 109 lines of specifications as inputs. This shows high productivity of this generation system.

1　はじめに

最近，様々な分野でデータベースを利用したいという要求がますます強まってきている．例えば，ＣＡＤ／ＣＡＭ，ＬＳＩ設計，ＣＡＳＥ，マルチメディアやネットワーク管理の分野である．それぞれの応用分野では，その分野特有のデータ表現や処理が必要となる．例えば，ＣＡＤ／ＣＡＭでは線図形を管理し，類似した構造を持つ線図形を求める機能が必要となる．一方，写真データベースでは類似した対象物が写っている写真を求めたり，類似した印象を持つ写真を求める機能が必要である．また例えば，写真データベースで必要とする機能は楽曲データベースで不要となる場合が多い．従って，全ての分野に対応した機能を持つデータベース管理システム（ＤＢＭＳ）を構築することは不可能に近く，万一構築できたとしても，ある応用分野にとっては不要な機能ばかりのものになる恐れが十分にある．従って，応用分野に特化したＤＢＭＳが必要であるが，ＤＢＭＳ構築には多大な労力と費用，そして時間が必要である．

この問題の解決策の一つとしてＤＢＭＳを拡張可能としておき，必要な機能のみを作成・付加することで，それぞれの応用分野に特化したＤＢＭＳを構築するという拡張可能ＤＢＭＳの研究が行なわれている．拡張可能ＤＢＭＳの研究アプローチは大きく２つに分けられる [1]．一つは，既存の完全な機能を持つＤＢＭＳの一部分を拡張可能とする方法である．この方法には，ＤＢＭＳの完全な機能があり，ＤＢＭＳ構築負荷も少なく，ソフトウェアとしても安全であるという利点があるが，拡張性が一部分に限定されるという問題がある．もう一つは，ＤＢＭＳ部品の組み合わせや自動生成によりＤＢＭＳを生成するという方法である．この方法は，高い拡張可能性を持たせることができるという利点があるが，ＤＢＭＳ構築負荷が大きく，ソフトウェアとして安全性にも問題があるという欠点を持つ．本論文では，高い拡張可能性を持つことに重点をおき，後者のアプローチを採用する．

本論文では，データモデルやデータベース言語の拡張可能化について検討する．オブジェクト指向データモデルに特化してデータモデルの拡張性を可能とする研究 [9] や，関連のみを拡張可能とする研究 [13] があるが，これらは限られた範囲でしかデータモデルを拡張可能としていない．一方，核となるデータモデルとデータモデルとデータベース言語を提供し，応用独自のデータモデルやデータベース言語からそれらへのマッピングを実装させる研究がある [11]．この方法は高度の拡張可能性を持つが，ＤＢＭＳ構築負荷が極めて大きくなると考えられる．従って，データモデルやデータベース言語を少ない負荷で利用可能にでき，しかも高度の拡張可能性を持たせる方法が望まれる．ここで，仕様に基づいてＤＢＭＳを生成する方式 [6] は，この要求を満足できると考えられるが，言語処理システムの自動生成はサポートされていない．

そこで本論文では，データベース言語処理システムの作成負荷を軽減するために，ユーザからの仕様書を基にデータベース言語処理システムを自動的に生成する方法について検討する．これにより，データベース言語処理システムを少ない負荷で構築可能とし，かつ，高い拡張性を持たせることを可能とする．

以下，２．では，本論文で前提とするデータベース言語処理システムについて説明する．３．では，データベース言語処理システムの自動生成法について述べる．４．では，生産効率を評価する．最後に５．でまとめる．

2　データベース言語処理システム

データベース言語処理システムは，データ定義言語やデータ操作言語を処理するシステムである．データベース言語処理システムの構成を図 1に示す．

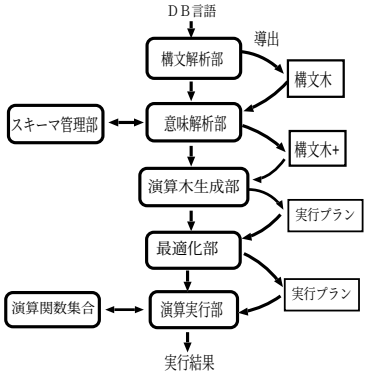

図 1: データベース言語処理システムの構成

図 1に示したように，システムは，データベース言語の文を入力とし，それを解釈・実行し，求められた結果を返す．以下に，各構成要素について説明する．

- スキーマ管理部 … スキーマ情報を管理する．例えば，関係モデルでは，テーブルの名前やカラム数，各カラムの名前やデータ型等を管理する．また，システム内部でテーブルやカラムに内部番号を付与する場合は，その内部番号を管理する．
- 構文解析部 … 入力文をトークンごとに区切り，言語の文法に合致しているか検査する．また，その文の構成を表す木 (構文木) を作成する．

- 意味解析部 … 入力文がデータ定義の下で適正であるかを検査する．また，スキーマ管理部からデータ定義情報をもらい，システム内部情報に変換する．
- 演算木生成部 … 演算優先度を基に，演算実行木を作成する．
- 最適化部 … データベース処理の手順を最適化し演算順序を表す演算実行木を作成する．
- 演算実行部 … 最適化された演算順序通りに演算を実行し，結果を返す．
- 演算関数集合 … 演算を実行する関数の集合．

また各構成要素間でデータを授受するために中間木を作成する．以下に中間木について述べる．

- 構文木 … 文構造を表す多分岐構造の木である．図3は，図 2のような SELECT 文を表現する構文木である．ルートノードには，SELECT 文の各節を表すノードへのポインタがある．

```
SELECT student. name ,  student. age
FROM   student
WHERE  student. age > 20
```

図 2: ＳＥＬＥＣＴ文の例

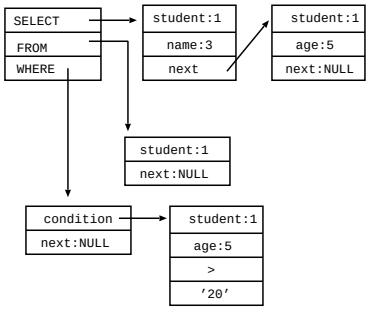

図 3: 構文木の例

- 構文木＋ … 構文木にスキーマ情報を加えた木である．図 2の SELECT 文では，SELECT 節に name と age が指定されている．図 3では，テーブル student の内部番号である 1，カラム name の内部番号である 3 等が加わっている．
- 実行プラン … 与えられた文に対応する処理を行うために演算の実行順序を表現した木である．図 4は２つのテーブルを結合 (JOIN) し，それに対して検索条件を適用する SELECT 文に対応する実行プランである．各ノードはデータベース処理の基本的な演算となっている．SCAN は，テーブルからデータを取り出す．JOIN は２つのテーブルを結合する．SELECT は検索条件を満足するレコードのみとし，PROJ は指定されたカラムのみにする．演算実行木の葉から順に基本演算を実行することで，対応するデータベース処理が行える．

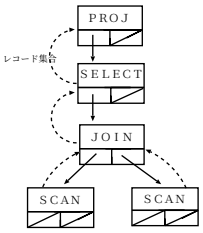

図 4: 実行プランの例

3　自動生成法

本章では，データベース言語処理システムの自動生成について述べる．まず，システムの概要について述べる．次に，各仕様書について述べる．最後に，データベース言語処理システムの各構成部を生成する方法を示す．

3.1　概要

データモデルやデータベース言語を少ない負荷で拡張可能とするために，ユーザからシステムの仕様書を入力として受け取り，要求にあったデータベース言語処理システムを自動生成する．図 5にデータベース言語処理システムの自動生成システムの構成を示す．

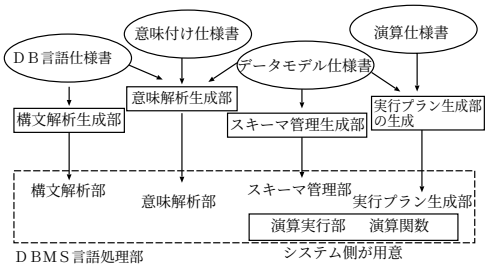

図 5: システム構成

仕様書には，データモデル仕様書，データベース言語仕様書，意味付け仕様書，演算仕様書がある．ユーザは，仕様書の規約に基づいて各々の仕様書を記述する．各生成部では，その情報によりデータベース管理システムの部品（プログラム）を自動的に生成する．データベース言語処理システムは，それらの部品の集まりによって作られる．ここで，演算実行部は自動生成せずシステム側で提供する．これは，演算木生成部で演算実行部の入力となる演算

木を生成できるようにできること，具体的な処理は演算関数で行うことが理由である．

3.2 データ表現と操作

3.2.1 データ表現

本システムでは，データをレコードの集まり (レコード集合と呼ぶ) として統一的に表現する．レコードは項目のｎ項目である．

レコード集合を表現するデータ構造を図 6に示す．レコード集合は，項目情報（スキーマ情報）とレコード実体をリスト構造で保持する．レコード情報は，レコードの集まりの先頭アドレスを持ち，レコードの集まりは，リスト構造で表す．また，各レコード内の項目もリスト構造で表す．項目情報は，各項目の項目 ID・データ型・サイズ・オリジナルのレコード番号・オリジナルの項目 ID を保持する．

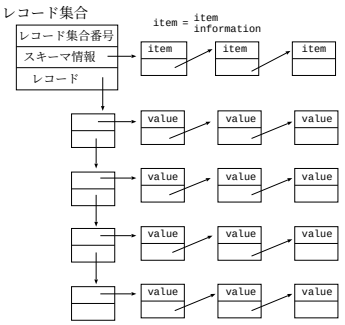

図 6: レコード集合を表すデータ構造

3.2.2 データの操作

本システムでは，基本演算の組み合わせでデータベース処理が行なえるとする．各基本演算は，レコード集合を入力とし，レコード集合を出力する．以下に演算の概略を示す．

SCAN 演算：ＳＣＡＮ演算は，データベースからレコード集合番号によりレコード集合をメモリ上に確保する演算である．

射影演算：射影演算は，レコード集合から項目 ID を基に項目単位に切り出す操作である．

選択演算：選択演算は，レコード集合からレコードの部分集合を取り出す操作である．

結合演算：結合演算は，結合条件から２つのレコード集合を１つのレコード集合にする操作である．

内部航行演算：内部航行演算は，１レコードに対してある項目のデータを求める操作である．

表 1: 記号一覧

記号	意味
[]	配列の大きさ
<>	構成要素記号
{}	ＯＲな構成要素集合
|	ＯＲ要素の区切り記号
()	省略可能記号
＋	要素の一回以上の繰り返し

外部航行演算：外部航行演算は，あるレコードに対してある項目のデータと目的のレコード集合の項目のデータが等しい目的のあるレコードの指定されたある項目のデータを求める操作である．

3.3 仕様書

3.3.1 データモデル仕様書

データモデル仕様書では，データモデルの構造を，一種の文法として定義式の集まりによって記述される．定義式は，基本的に以下の形式で記述する．

〈データモデル要素〉{=|:|#}〈構成要素〉…ここで，=は通常の定義を示す．:は，=に加え，独立して存在するデータモデル要素であることを示す．さらに，#は，:に加え，データ実体にデータ実体を一意に識別する識別子が付与されることを示す．構成要素の記述に使用する記号の意味を表 1に示す．

：または，＃で定義したデータモデル要素は，一意に識別するための要素を identified by〈構成要素〉によって宣言しなければならない． また,〈構成要素〉は〈データモデル要素〉として定義されていなければならない．定義式の集まりの開始と終了を文字%で表す．データモデル要素に型の制約を付けることができる．それには，開始の%の前にデータ型を以下の形式で宣言する．

〈制約要素〉＝〈データ型〉

宣言した〈制約要素〉は，制約を付けたいデータモデル要素である．表 2に〈データ型〉に記述できるデータ型を示す．関係モデルの定義例を図 7に示す．

```
Datatype = Integer, Char, Float, Double
%
name = char [16]
Datatype = int
column = <name><Datatype>
table : <name><column>+  identified_by name
%
```

図 7: 関係モデルの定義例

表 2: システムが用意するデータ型

型名	データ型
Oid	オブジェクトＩＤ (int 型)
Integer	整数型
Char	文字列型
Float	単精度浮動小数型
Double	倍精度浮動小数型
Boolan	真か偽
All	上記の全て

3.3.2 言語仕様書

データベース言語仕様書では，言語の構造や文法を定義する．まず，言語で使用するトークン (予約語) を定義する．トークンは，"と "で囲って宣言する．次に，言語要素を宣言する．言語要素は，以下の形で定義する．

〈言語要素〉:〈構成要素〉;

また，言語要素の構成が複数ある時は，|を使って定義する．なお，言語のスタート要素（一番始めの解析要素）には，＃を先頭に付ける．表 3に使用する記号とその意味を示す．

表 3: 言語仕様書の記号一覧

記号	意味
%	開始終了記号
"	トークン宣言
[]	トークン宣言
#	スタート言語要素
:	宣言文区切り記号
;	宣言文終了記号
|	構成分岐記号
’	区切りトークン宣言

記述上の制約として,〈構成要素〉内では同じ名前の構成要素を書くことができないことが挙げられる．これは，言語の構成要素を名前で一意に識別しているためである．また，[] で宣言されるトークン（文字列・整数・浮動小数・演算子）については，システム側で用意するトークン名を使用しなければならない．トークンの一覧を表 4に示す．また，図 8にＳＱＬの一部の定義例を示す．ここでは，簡単な SELECT 文を定義している．

表 4: システムが用意するトークン名

トークン名	意味
IDENT	文字列
NUMBER	整数
DOUBLE	浮動小数
OPERATOR	演算子
;	文字

```
%
[\’.(,)]   ;
"SELECT"
"FROM"
"WHERE"
"AND"
[a-zA-Z0-9_]*   IDENT
[0-9]+          NUMBER
[<>=]           OPERATOR
%
# sentence
            : specification
            ;
specification
            : SELECT selectlist
              FROM   fromlist
              WHERE  condition
            ;
selectlist
            : pair_element
            | selectlist ’,’ pair_element
            ;
pair_element
            : columnname
            | tablename ’.’ columnname
            ;
tablename
            : IDENT
            ;
columnname
            : IDENT
            ;
fromlist
            : tablename
            | fromlist ’,’ tablename
            ;
%
```

図 8: データベース言語仕様書の例

3.3.3 言語制約仕様書

言語制約仕様書では，論理演算と言語要素の関係の記述，データモデル要素と言語要素の対応関係の定義，ならびに，データベース言語の文法制約を定義する．言語制約仕様書では，まず論理演算を定義する．言語で論理演算として使うトークンを以下の形で定義する．

＜ Logic-Operator ＞ ＜トークン列＞

そして，論理積や論理和に対応するトークンを順に以下の形で定義する．

表 5: 制約

制約	意味
distinct	リスト内の要素の重複を許さない
identified by	要素の一意識別の単位
correspond	対象要素⊂対応要素の関係

＜ xxx-Operator ＞=＜トークン＞

ここで xxx は，ＡＮＤ，ＯＲ，または，ＮＯＴである．

次に，データモデル要素と言語要素の対応関係を定義する．これは，ＩＤＥＮＴトークンで構成されるすべての言語要素について以下の形で定義する．

＜言語要素＞=＜対応するモデル要素＞

最後にデータベース言語の文法制約を定義する．文法制約には，表 5に示すように言語要素に３つの制約をつけることができる．制約定義の例を図 9に示す．

```
%
Logic-Operator AND OR NOT
AND-Operator = "AND"

tablename = <table>-<name>
columnname = <column>-<name>
%
selectlist  <distinct>
fromlist    <distinct>
pair_element <identified_by> tablename, columnname
pair_element-tablename <correspond> fromlist-tablename
%
```

図 9: 言語制約仕様書の例

3.3.4 演算仕様書

演算仕様書は，演算と言語の関係について記述する．演算仕様書では，システム側で用意した演算に必要な情報がユーザの設計した言語のどの言語要素と対応するかを定義する．

3.4 各実行部の生成

各生成部では，図 5で示したように仕様書からの情報を使用して，データベース言語処理プログラムを自動生成する．出力するプログラムは，Ｃ言語のプログラムである．ここでは，各部の役割と共に自動生成のアルゴリズムの概要を示す．

```
Scantable "Scan"
          Record_name = fromlist-tablename
Select    "Select"
          condition-predicate-select_pred
             Rdata={pair_element2|NUMBER|IDENT}
             Ldata={pair_element|NUMBER|IDENT}
             Operator = OPERATOR
Proj      "Projection"
          Navi_list = selectlist-pair_element
Join      "Join"
          condition-predicate-join_pred
             Rdata = pair_element2
             Ldata = pair_element
             Operator = OPERATOR
             Table1 = pair_element-tablename
             Table2 = pair_element2-tablename
Tablenavi "Navigation"
          pair_element
             Record = tablename
             Item   = columnname
          pair_element2
             Record = tablename
             Item   = columnname
%
```

図 10: 演算仕様書の例

3.4.1 スキーマ管理生成部

スキーマ管理部のソースプログラムは，スキーマ構造を表すヘッダファイル model．h ならびに，スキーマの操作 (挿入・削除・検索) を担う model．c で構成される．スキーマ管理生成部では，データモデル木の作成，ヘッダファイルの作成，スキーマ管理部のソース作成の手順でスキーマ管理部を生成する．

(a) データモデル木

データモデル木は，データモデル仕様書の情報をプログラムで扱いやすいように木構造で表現したものである．

(b) ヘッダファイルの生成

ここでは，独立したデータモデル要素を一つの構造体で表現する．内部番号と全体長，ならびに，個々の構成要素の内部通番も保持するようにしている．

(c) スキーマ管理部のソースコードの生成

管理する構造体が決定すると，その構造体の生成，削除，更新，検索を行うプログラムを生成することができる．例えば，検索では，名前や内部番号が与えられたときに，それらに基づいてスキーマ情報を管理するプログラムを生成することになる．他の関数も同様である．

3.4.2 構文解析生成部

ここでは，構文解析支援ツールである lex・yacc を使用して構文解析部を生成する．従って，構文解析生成部では，lex・yacc を実行させるファイル，lex の仕様書ファイル，ならびに，yacc の仕様書ファイルをユーザからの仕様書から生成する．構文解析生成部では，データベース言語情報木の作成，字句解析ファイルの生成，構文解析ファイルの生成の手順で上記のファイルを生成する．

(a) データベース言語情報木

データベース言語情報木は，データベース言語仕様書の情報をプログラムで扱い易いように情報を木構造で表現したものである．

(b) 字句解析（lex）ファイルの生成

lex の仕様書ファイルは，データベース言語仕様書から容易に生成できる．

(c) 構文解析（yacc）ファイルの生成アルゴリズム

yacc の仕様書ファイルの作成にあたっての問題は，アクション，すなわち，構文木を生成するためのアクションの生成である．ここでは，SQL に代表される高度の高水準言語を対象とするとすると，構文木は言語要素を表す構造体同士がリスト構造で繋がっている程度である．従って，構文木生成プログラムは，以下の処理を行えば良い．

- 言語要素の情報を表す構造体の領域確保
- その構造体の初期化
- 言語要素情報の入力
- リスト処理

この程度の処理と考えると，自動生成は可能である．

3.4.3 意味解析生成部

意味解析生成部では，以下の手順に従って意味解析部を実装する．

(a) 言語制約情報木の作成

言語制約情報木は，言語制約仕様書をプログラムで扱い易いように情報を木構造で表現したものである．

(b) システム番号変換コードの作成

システム番号変換処理は，スキーマ情報に関する情報を構文木より抽出し，スキーマ管理部にその情報について問い合わせ，システム内部番号を返却してもらい，その番号を構文木に埋め込む処理をする．

(c) 整合性チェックコードの作成

対応付け情報処理は，対象となる情報が対応付けられる情報に含まれているかを検査する処理である．また，対象の情報が不足している場合，対応する情報に適切なスキーマ情報があれば，それにより対象の情報を補うことを行なう処理である．

(d) 重複チェックコードの作成

重複チェック処理は，ユーザから指定されたリスト構造の言語要素について，情報の重複を検査する処理である．

3.4.4 実行プラン生成部の生成

実行プラン生成部は，演算の優先順位を基に演算実行手順を表す実行プラン情報木を作成する．実行プラン生成部の生成では，実行プランの作成，実行プラン生成コードの作成という手順で実行プラン生成部を生成する．

(a) 実行プラン情報木の生成

実行プラン木は，演算仕様書をプログラムで扱い易いように情報を木構造で表現したものである．

(b) 実行プラン生成コードの作成

演算の優先度は，SCAN ＞結合＞選択＞射影とする．内部航行と外部航行は先ほどの演算内で随時呼ばれる形になる．それぞれの演算を表すノードを生成し，実行プランを生成する．

4 評価

データベース言語処理システムの自動生成システムの生産性を，以下に示す観点に基づいて評価する．

(1) 仕様書量と生成コード量：あるデータモデルとデータベース言語に対するデータベース言語処理システムを生成するために必要な仕様書の行数と，自動生成されたソースファイルの行数を比較して，生産性を評価する．

(2) データベース言語処理システムの作成工数：自動生成システムを使用した場合と使用しない場合でデータベース言語処理システム作成に要する工数を比較する．

関係モデルとＳＱＬの一部の定義のための各仕様書の行数を表6に示す．また，生成されたプログラムファイルの総行数は３６９７行である．これを手作業により構築した場合は１人で約６ヶ月かかり，Ｃ言語で４１５８行必要であった．これに対し，自動生成システムを使用すると，主に仕様書作成に４日ですんだ．

関数モデルとＤＡＰＬＥＸ言語の一部の定義のための各仕様書の行数を表6に示す．また，生成されたプログラムファイルの総行数は３８８２行である．これを手作業により構築した場合は１人で約４ヶ月かかり，Ｃ言語で４１３０行必要であった．これに対し，自動生成システムを使用すると，主に仕様書作成に４日ですんだ．

仕様書に対する生成コード量は，約３０倍にもなっており，生成効率が高いことが分かる．

表 6: 仕様書ファイルの行数

仕様書	関係モデル	関数モデル
データモデル仕様書	7	11
データベース言語仕様書	64	51
言語制約仕様書	14	12
演算仕様書	24	20
合計	109	94

作成工数は，システムを使用しない場合に比べて，使用した場合は約１／３０に短縮に減少している．

5 おわりに

本論文では，応用分野独自のＤＢＭＳの構築際の負荷の軽減を目的として，ＤＢＭＳの重要な部分であるスキーマ管理部とデータベース言語処理部からなるデータベース言語処理システムをユーザからの仕様書により自動生成する手法について述べた．ユーザからのデータモデル仕様書・データベース言語仕様書・言語制約仕様書・演算仕様書を入力とし，データベース言語処理システムのモジュールである，構文解析部・意味解析部・スキーマ管理部・演算木生成部・演算実行部・演算関数集合を生成する．プロトタイプシステムを構築し，関係モデル・ＳＱＬ，ならびに関数モデル・ＤＡＰＬＥＸ言語に対してデータベース言語処理システムを生成させたところ，行数で約３０倍，生成工数が約１／３０の生産性向上が達成できた．これにより，本システムは，ＤＢＭＳの構築の負荷を時間的に大幅に短縮できることを明らかにした．

本システムは，本論文で示した２つの例にしか適用していない．また，仕様書の制約によって定義できないデータモデルが存在する．例えば，オブジェクト指向データモデルのように，クラスの中にクラスが入れ子になる場合やふるまい (メソッド) を持つデータモデルは記述できない．また，システム側で準備する演算が十分でないことも問題点である．例えば，集約演算である．また，本システムでは，最適化部についての考慮をしていない．これらは今後の課題である．

参考文献

[1] Carey, M., and Haas, L. : "Extensible Database Management Systems," SIGMOD RECORD, Vol. 19, No. 4 (1990).

[2] Stonebraker, M., Rowe, L. A., and Hirohama M. : "The implementation of POSTGRES," IEEE Trans. of Know. and Data Eng., Vol.2, No.1, pp. 125-142 (1990).

[3] Haas, L. M. *et al* : "Starburst mid-flight: As the dust clears," IEEE Trans. of Know. and Data Eng., Vol.2, No.1, pp. 143-160 (1990).

[4] Batory, D. S. *et al* : "GENESIS: An Extensible Database Management System," IEEE Trans. on Soft. Eng., 14, 11, pp. 1711-1730 (1990).

[5] Carey, M. J. *et al* : "The Architecture of the EXODUS Extensible DBMS," Int'l Workshop on Object-Oriented Database Systems, pp. 52-65 (1986).

[6] Dittrich, K. R. : "Database Technology Research at the University of Zurich: Using and Engineering Object-oriented, Active, and Heterogeneous DBMS," IEICE Tech. Report DE91-60, Vol. 91, No. 538 (1992).

[7] Wells, D. L., Blakeley, J. A., and Thompson C. W. : "Architecture of an Open Object-Oriented Database Management System," COMPUTER, Vol. 25, No.10, pp. 74-82 (1992).

[8] Batory, D. S.: "Concepts for a Database System Compiler," Proc. of 7th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp.184–192 (1988).

[9] Hong. S. and Maryanski F. : "Representation of Object-Oriented Data Models," Inform. Sciences, No.52, pp. 247-284 (1990).

[10] Cooper, R. : "Configurable Data Modelling Systems", Entity-Relationship Approach: The Core of Conceptual Modelling (H. Kangassaro ed.), Elsevier Science Publishers B. V.(North-Holland), pp.57–73 (1991).

[11] Demurjian S. A. and Hsiao D. K. : "Towards a Better Understanding of Data Models through the Multilingual Databas e System," IEEE Trans. Softw. Eng., Vol. 14, No. 7, pp. 946-958 (1988).

[12] Geppert, A. and Dittrich, K. R. : "Constructing the Next 100 Database Management Systems: Like the Handyman or Like the engineer?," ACM SIGMOD RECORD, Vol. 23, No. 1, pp. 27–33 (1994).

[13] Zhang, N., Härder, T. : "On a Buzzword "Extensibility" – What We Have Learned from the ORIENT Project?," Proc. of 1999 International Database Engineering and Applications Symposium (IDEAS'99), pp. 360–369 (1999).

[14] Shipman,D.W.: The Function Data Model and the Data Language DAPLEX, ACM Transactions on Database Systems,Vol.6,No.1, pp.140-173(1981).

[15] Hochin, T. and Inoue, U. : "An Extensible DBMS Composed of Specific DBMSs," Proc. of International Symposium on Next Generation Database Systems and Their Applications, pp.180–187 (1993).

[16] 平林孝之, 寶珍輝尚, 都司達夫:データモデルに応じたスキーマ管理部の一生成法, 第５８回 情報処理全国大会 1V-6 (1999).