データ変換支援ツール評価のためのベンチマーク構築フレームワークの提案

大河原俊明† 森嶋 厚行†† 杉本 重雄††

† 筑波大学大学院 図書館情報メディア研究科 〒305-8550 茨城県つくば市春日 1–2
†† 筑波大学大学院 図書館情報メディア研究科/知的コミュニティ基盤研究センター
〒305-8550 茨城県つくば市春日 1–2
E-mail: †{okwr,mori,sugimoto}@slis.tsukuba.ac.jp

あらまし 近年，異なるデータベース間のデータ変換の重要性が増大しており，データ変換を支援するためのツールが多数開発されている．一方，これら多数のツールの性能を評価するための方法については議論されてこなかった．DBMS の性能を評価するためには様々なベンチマークを利用するように，データ変換支援ツールを評価するためのベンチマークがあれば便利であると考えられる．しかし，データ変換支援ツールの性能に影響を与える要因が，DBMS の性能に影響を与える要因と全く異なるため，ベンチマーク構築にも異なるアプローチが必要である．本稿では，データ変換支援ツールを評価するためのベンチマークはどうあるべきかを議論し，ベンチマーク構築のためのフレームワークを提案する．さらに，提案ベンチマークを用いてデータ変換支援ツールを評価した例を示す．

キーワード ベンチマーク，情報統合，XML，スキーママッチング

1. はじめに

近年，異なる情報源間のデータ変換の重要性が増大している．我々は，リレーショナルデータベースや XML などのような，(半) 構造的なスキーマを持った情報源に焦点を当てる (以後，そのような情報源を**データベース**と呼ぶ)．本稿では，データ変換を次のようにモデル化する．すなわち，変換元データベーススキーマ S，変換先データベーススキーマ T，および S のインスタンス I^S が与えられたとき，データ変換は $I^T = F(I^S)$ で表される．一般に関数 F は何らかのプログラムで実装されるが，我々の問題ではデータベース問合せ (以後，**データ変換問合せ**) で実装される．具体的には，データ変換問合せは XQuery や SQL などの問合せ言語で記述される．ここで問題となるのは，変換元スキーマ S および変換先スキーマ T が与えられたとき，どのように F を構築するかである．しかし，F の構築を人手で行うことは多大なコストがかかることが知られている．したがって，どのように効率よく構築コストを減少させるかが重要な課題として認識されている [5]．さらに，XML の普及に従い，このデータ変換の問題に対する効率的な解への要求がますます増大している．なぜなら，XML データのスキーマ (DTD など) は，伝統的なリレーショナルデータベースのスキーマと比較して，複雑かつ大規模であるからである．例えば，XHTML1.0 Strict [9] の DTD のサイズは約 1,000 行もある．

これまで，データ変換を支援するツールが多数開発されてきている [2] [3] [4]．一般に，これらのツールは次のように利用される (図 1)．まず，入力として，2 つのスキーマ S，T，および S のインスタンス I^S を与える．次に，S の構成要素と T の構成要素を対応付けるために，利用者がデータ変換記述 D(しばしば，**スキーママッピング**と呼ばれる) を作成する．最後に，ツールが，これらの入力を基にデータ変換問合せ F を構築し，出力する．

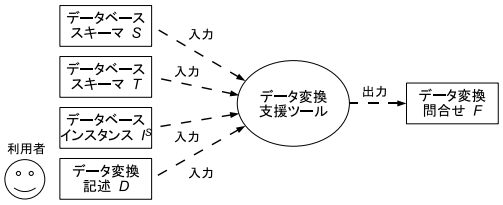


図 1 データ変換支援ツールの概要

これまでのデータ変換の研究における焦点は，与えられたスキーマおよび D から F を構築するための方法におかれており，F を構築するための**人的コスト**に関してはほとんど議論されてこなかった．ここで，F 構築のための人的コストとは，ツールの利用者がデータ変換記述 D を作成するために必要となるコストである．本稿では，我々は D の作成に必要となる時間に焦点を当てる．以後，この時間を $cost(D)$ と表記する．

一般に，$cost(D)$ は非常に大きくなる．例として，データ変換問合せの作成を支援する代表的なツールである IBM の Clio [2] を用いて説明する．図 2 は，ある大学の情報を格納するための変換元データベースから教授と学生の給与を格納するための変換先データベースへのデータ変換を示したものである．ここで，Clio への入力となるデータ変換記述 D は $\{f_1, f_2, f_3, f_4\}$ である．各 f_i は次の 2 つの部分から構成される．(1)S の属性と T の属性間の対応関係 (線を用いて表記)，(2)T の属性の値を計算するための式 (線の横に表記)．例えば，図 2 の f_4 は (1)T の Sal 属性が S の HrRate 属性 (アルバイト

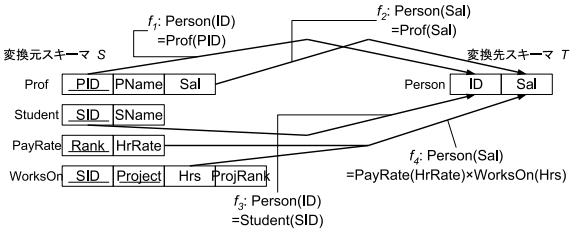


図 2　Clio におけるデータ変換記述 D

の時給) と Hrs 属性 (時間) に対応し，(2)T の Sal 属性の値は Person(Sal)=PayRate(HrRate)×WorksOn(Hrs) で計算されることを表している．一般には，各 f_i を発見することは簡単ではない．この例では，f_1 から f_3 を発見することは比較的簡単であるが，f_4 を発見することが難しい．なぜなら，Student, PayRate，および WorksOn の各リレーション間の外部キー制約を理解しなければならないためである．以上の例で示したように，小さなスキーマ間のデータ変換でもデータ変換記述 D の作成は自明ではなく，より大きく複雑なスキーマ間のデータ変換の場合はその困難さがさらに増大する．

一般に，既存の様々なツールはデータ変換記述 D のための異なる言語を持ち，変換元スキーマと変換先スキーマ間の関係を記述するための様々な方法を提供する．加えて，各ツールは，利用者に対してデータ変換記述作成を支援するための，様々な機構およびインタフェースを持つ．例えば，スキーマの詳細を隠すための機構や，指定された種類のスキーマの構成要素をハイライトするための機構などである．

以上のような様々な言語や支援機構を持つツールの効果を評価するためにはベンチマークがあれば便利であるが，この目的に合うようなベンチマークはこれまで存在しなかった．The Lowell Database Research Self-Assessment [1] でも，情報統合のテストベッドの構築が重要と報告されており，データ変換のためのベンチマーク構築の重要性は高いと考えられる．

しかし，これまで，この目的に合うようなベンチマークは我々の知る限り存在しなかった．実は，そのようなベンチマークの構築は自明ではない．なぜなら，問合せの効率的な作成に影響を与える要因が，DBMS の性能に影響を与える要因とは全く異なるため，既存の DBMS のベンチマークと同様のフレームワークを利用できないためである．

本稿では，データ変換支援ツールの評価用のベンチマークを構築するためのフレームワークを提案する．本稿で提案するのは，ある特定のベンチマークではなく，ベンチマーク構築の**フレームワーク**である．本研究の最も重要な貢献は，データ変換の問題のバリエーションを**体系的な方法**で導出するための機構を開発したことである．データ変換の問題のバリエーションは，データ変換支援ツールの効果を様々な側面から評価するために重要である．

本稿の構成は次である．2 章では，データ変換支援ツールのベンチマークが DBMS のベンチマークとのどのように異なるのかを説明する．3 章では，我々の要求を満たす，データ変換支援ツールのベンチマーク構築のためのフレームワークを提案

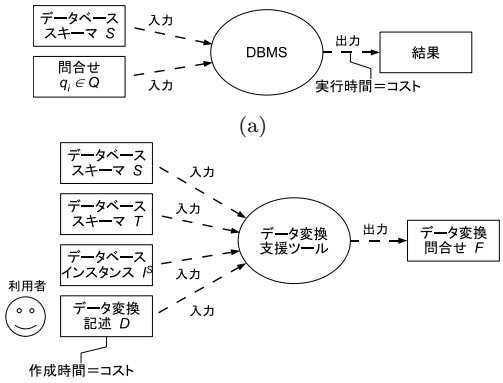


図 3　DBMS の評価 (a) とデータ変換支援ツールの評価 (b)

する．4 章では，提案フレームワークを利用して構築したサンプルベンチマークと，そのサンプルベンチマークでデータ変換支援ツールを評価した例を説明する．5 章では関連研究を説明する．6 章はまとめである．

2. データ変換支援ツールのベンチマークにおいて考慮すべき要因

まず，比較のために，DBMS の問合せ実行性能を評価するためのベンチマーク構築の一般的なフレームワークを説明する．DBMS の性能を評価するためのベンチマーク BM は式 1 のように定義できる．

$$BM = (S, g, Q) \tag{1}$$

ここで，S はデータベーススキーマ，$g : Int \rightarrow S$ は，データベースのサイズを表す数を引数としスキーマ S に従うデータベースインスタンスを生成するための関数 (データベースインスタンスジェネレータ)，Q は生成されたデータベースインスタンスに対する問合せ集合である．このベンチマークの利用は次のように行われる．まず，g を用いて与えられたサイズ (例えば 10GB) のデータベースインスタンスを生成する．次に，生成されたデータベースインスタンスを持つ DBMS に対して各問合せ $q_i \in Q$ を実行し，その実行時間を計測する．そこで計測された時間を DBMS の性能を表すコストとする (図 3(a))．

一方，データ変換支援ツールを評価するためのベンチマークは，同じフレームワークを利用できない．その理由は，ツールの入出力が DBMS における入出力と異なる (図 3(b)) こともあるが，より本質的には，計測されるコストに影響を与える要因が全く異なるからである．まず，DBMS の性能に影響を与える要因は，(1) 各問合せ $q_i \in Q$，(2) データベースインスタンスのサイズである．一方，データ変換問合せの作成コストに影響を与える重要な要因は，次の 3 つである．

（1） **人的要因．**データ変換支援ツールに対してデータ変換記述 D を入力するのは利用者である．したがって，人的要因が影響を与えるのは自然であると考えられる．特に，$cost(D)$ は利用者が変換元および変換先のデータベースに関して，どれほど「知識」を持つかによって影響を受ける．したがって，べ

ンチマークには利用者に与える知識を制御できる機構が必要となる．

（2） **データベーススキーマのサイズ．**データベーススキーマのサイズが大きくなれば，$cost(D)$ は増大する．

（3） **データベーススキーマ間の関係の複雑さ．**データベーススキーマ間の関係がより複雑になると $cost(D)$ は増大する．

以上の要因のうち，要因 (2)(3) を考慮すると，データ変換支援ツールのためのベンチマークには異なるサイズや複雑さといったデータベーススキーマのバリエーションが必要となることがわかる．しかし,「データベーススキーマの生成」は，高度に知的な作業であるため，これを自動的に行うデータベーススキーマジェネレータをソフトウェアツールとして用意するのことは困難である．これは，DBMS のベンチマークで必要な，与えられたサイズの「データベースインスタンスの生成」が比較的容易であることと対照的である．

3. 提案フレームワーク

これまでの議論を考慮して，我々はデータ変換支援ツールのベンチマークのためのフレームワークを提案する．提案するフレームワークの基本的なアイデアは，次の通りである．

- ツールの利用者にあらかじめ与えられる知識を制御するための仕組みを，ベンチマークに組み込む．この仕組みによって，利用者が持つ知識の量を変化させながら，データ変換問合せの作成コストを計測することが可能となる．
- スキーマの自動生成は困難であるため，これは行わない．その代わり，あらかじめある程度大きなサイズの問題を作成しておき，様々なスキーマのサイズや複雑さを持つ「部分問題」を柔軟に導出できる仕組みをベンチマークに用意する．

3.1 ベンチマークの構造

我々はデータ変換支援ツールのためのベンチマーク BM を式 2 のように定義する．

$$BM = (S, I^S, T, K, C, P^C, B^S, B^T) \quad (2)$$

ここで，S は変換元スキーマ (例えば，図 4(a))，I^S は S のあるインスタンス (図 4(b))，および T は変換先スキーマ (図 4(c)) である．本稿では，変換対象のデータベースを XML データとして表記する．なぜなら，XML スキーマは，リレーショナルデータベーススキーマよりも表現能力が高いからである．しかし，提案フレームワークは XML に依存するものではないことに注意して欲しい．次に，K は**知識表** (図 7) である．知識表は，データ変換支援ツールの利用者があらかじめ知っているべき知識を記述している．知識表の詳細は 3.4 節で説明する．

このベンチマークを利用して実験を行う際に，被験者 (ツールの利用者) に提示されるものは，S，I^S，T，および K だけである．具体的には，ベンチマークの利用の全体の流れは次のようになる．まず，ベンチマークを用いて実験を行う前に，知識表 K のどの部分を被験者に提示するかを決定する．次に，S，I^S，および T をデータ変換支援ツールに与える．さらに，あらかじめ決めていた K の一部を被験者に提示し，被験者は S の構成要素と T の構成要素間の対応関係を指定するために D を作成する．このとき，D を作成するためにかかった時間である $cost(D)$ を計測する．計測結果である $cost(D)$ を公開するときは，計測したときに被験者に提示した K の部分を共に公開しなければならない．

ベンチマーク BM の残りの構成要素 C，P^C，B^S，および B^T は，BM の部分問題を導出する際や，出力されたデータ変換問合せが正しいか否かを検証する際に用いられる．以下でそれぞれを説明する．まず，C はデータベーススキーマ S および T が実装しているデータの概念モデル (クラス図)(例えば，図 5(a)) である．$P^C : Int \rightarrow ClassDiagram$ は，ある数が与えられると，概念モデル C の一部を導出する関数 (**プロジェクション関数**と呼ぶ) である．プロジェクション関数の詳細は次節で説明する．$B^S : S \rightarrow C$ は，データベーススキーマ S のインスタンス I^S から C のインスタンス (3.3 節で説明する) を計算する関数である．同様に，$B^T : T \rightarrow C$ は T のインスタンス I^T から C のインスタンスを計算する関数である．

次節からは，図 4 から図 7 に示すベンチマーク例を用いて，ベンチマーク BM のそれぞれの構成要素について順に説明する．

3.2 スキーマ，クラス図，プロジェクション関数

変換元スキーマ S と変換先スキーマ T は，共に概念モデル C(もしくはその一部) を実装したデータベーススキーマである．ここではデータの変換が目的であるので，S と T は同じ概念モデルを実装していても構造は異なっている必要がある．例えば，図 4(a) に示す変換元スキーマ S と図 4(c) に示す変換先スキーマ T は図 5(a) のクラス図 C に示す同じ概念モデルを実装している．このベンチマークでは，S では Prof クラスと Student クラスがそれぞれ別々の XML 要素として実装されているのに対し，T ではその 2 つのクラスの上位クラスである Person クラスとして実装されている．また，S では概念モデル C の構成要素がすべて実装されているのに対し，T では Person クラスの ID と Sal だけを実装している．

概念モデル C の各構成要素 (クラス，関連，あるいは属性) は，それぞれ identifier となる数を持つ (図 5(a))．したがって，各構成要素をその identifier を用いて同定できる．

C の**部分クラス図**を C_n と表記する．ここで，n は部分クラス図 C_n に含まれる構成要素が，C のどの構成要素に対応するかを表すための数であり，**プロジェクション数**と呼ぶ．このプロジェクション数 n は，C の構成要素の identifier のうち，C_n に含まれる構成要素だけを用いて次のように計算される．すなわち，C 中で identifier id を持つ構成要素が C_n に含まれているとき，n の id 番目のビットを 1 にし，それ以外を 0 にする．例えば，図 5(a) の C において Person クラスとその属性だけからなる C_n を必要とする場合，$n = 1111_{(2)} = F_{(16)}$(すなわち，$C_n = C_F$) である．以後では，特に断らない限り 16 進数を用いてプロジェクション数 n を表記する．

形式的には，与えられた n から C_n を導出する役割を持つのは，ベンチマークに含まれるプロジェクション関数 $P^C : Int \rightarrow ClassDiagram$ である．例えば，C の一部 C_F は $C_F = P^C(F)$ により計算される．このプロジェクション関数

```
Data = (Prof*, Student*, PayRate*, WorksOn*)
Prof = (@PID:ID, PName, Sal)
Student = (@SID:ID, SName)
PayRate = (@Rank:ID, HrRate)
WorksOn = (@SID:IDREF, @Project:ID, Hrs, @ProjRank:IDREF)
```

(a)

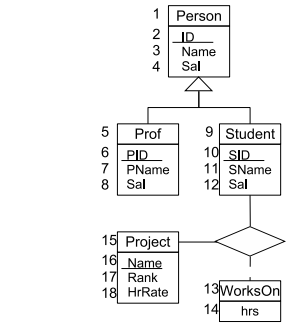
```
<Data>
  <Prof PID="P001"><PName>Morishima</PName>
    <Sal>7000</Sal></Prof>
  <Prof PID="P002"><PName>Sugimoto</PName>
    <Sal>9000</Sal></Prof>
  <Student SID="S001"><SName>Okawara</SName></Student>
  <Student SID="S002"><SName>Tanaka</SName></Student>
  <Student SID="S003"><SName>Eiju</SName></Student>
  <PayRate Rank="1"><HrRate>70</HrRate></PayRate>
  <PayRate Rank="2"><HrRate>60</HrRate></PayRate>
  <WorksOn SID="S001" Project="SMART" ProjRank="1">
    <Hrs>100</Hrs></WorksOn>
  <WorksOn SID="S002" Project="Clio" ProjRank="2">
    <Hrs>70</Hrs></WorksOn>
  <WorksOn SID="S003" Project="SMART" ProjRank="1">
    <Hrs>30</Hrs></WorksOn>
</Data>
```

(b)

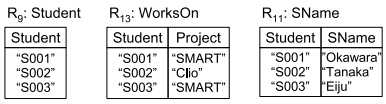
```
Data = (Person*)
Person = (ID, Sal)
```

(c)

図 4　ベンチマーク例: 変換元スキーマ S(a), S のインスタンス I^S(b), および変換先スキーマ T(c)



(a)



(b)

図 5　ベンチマーク例: 概念モデル C(a), および C のインスタンスの一部 (b)

P^C は，3.5 節で説明するベンチマークの部分問題を導出するための重要な役割を担う．

3.3　クラス図のインスタンス

関数 $B^S : S \to C$(あるいは $B^T : T \to C$) は，データベーススキーマ S(あるいは T) のインスタンスからクラス図 C のインスタンスを計算する．それぞれの関数は我々が **binding expression** と呼ぶ式 (例えば，図 6) の集合で表される．本節では，まずクラス図のインスタンスについて説明し，次に binding expression の説明をする．

クラス図 C のインスタンスとは，C の各構成要素 (クラス，関連，あるいは属性) にそれぞれ対応するリレーションの集合である．クラス図 C のインスタンスと，データベースインスタンスの関係は次の通りである．すなわち，C を実装したデータベーススキーマのインスタンスが与えられると，そのデータベースインスタンスからクラス図 C のインスタンスを計算できる．例えば，図 5(b) は図 4(b) に示すインスタンスから計算されたクラス図 (図 5(a)) のインスタンスの一部である．各リレーションは，対応する構成要素の種類 (クラス，関連，あるいは属性) によって，次のいずれかのリレーションスキーマとなる．

- **クラス** c: クラスの ID 属性に対応する単項リレーション (例えば，図 5(b) R_9)
- **関連** r: 関連で接続されている各クラスの ID 属性に対応する n 項リレーション (図 5(b) R_{13})
- **属性** a: その属性が所属するクラスの ID 属性とその属性の組に対応する 2 項リレーション (図 5(b) R_{11})

```
expr_{S→9}:
  for $x9 in //Student/@SID return $x9 as Student
expr_{S→13}:
  for $x131 in //Student/@SID, $x132 in //WorksOn/@Project
  return $x131 as Student, $x132 as Project
expr_{S→11}:
  for $t11 in //Student, $k11 in $t11/@SID, $x11 in $t11/SName
  return $k11 as  Student,  $x11 as SName
```

図 6　ベンチマーク例: B^S 中の binding expression の一部

Binding expression の説明: ベンチマークに含まれる関数 $B^S : S \to C$ は，S のインスタンス I^S が与えられたとき C のインスタンスを計算する関数である．この関数は，binding expression の集合で表現される．形式的には，B^S は次のように定義される:

$$\{expr_{S\to i} | i \in component(C)\}$$

ここで，i はクラス図 C の各構成要素が持つ identifier であり，$expr_{S\to i}$ は各構成要素に対応するリレーションを計算するための binding expression である．

Binding expression は XQuery [10] 風の構文を持つ．例えば，図 6 は図 4(a) に示す変換元スキーマと図 5(a) に示すクラス図のための B^S の一部である binding expression を表す．図 6 において，$expr_{S\to 11}$ は Student クラスの SName 属性に対応する Student 属性と SName 属性から構成される 2 項リレーション (図 6(b) R_{11}) を計算する．`for` 節は XQuery の flwr 節のものと同じである．唯一の違いは，`return` 節「式 `as` 属性名」である．これは，**式**を計算した結果が拘束された**属性**の値になることを表す．

3.4　知　識　表

2 章で説明したように，データ変換記述 D の作成に必要な $cost(D)$ は，ツール利用者が持つ (問題に関する) 知識に影響を受ける．したがって，ベンチマーク適用前にツールの利用者に提示される知識を制御できる仕組みが必要である．本フレームワークでは**知識表** K を用いて実現する．この表は，我々が **facts** と呼ぶ複数の知識の項目から構成される．K に含まれる facts は次の 2 つに分類される．

Indispensable facts. これを知らなければ，利用者がデータ変換記述 D を作成することができない知識．

Optional facts. これを知っていることによって，利用者が効

Indispensable facts
person は professor あるいは student である student の salary は student が働いている project の時間と時給を乗算したものである
Optiona facts (変換元スキーマ S)
Person: //Prof ⊎ (//Student ⋈ // PayRate ⋈ //WorksOn) Prof: //Prof (*) Student: //Student ⋈ //PayRate ⋈ //WorksOn WorksOn: //WorksOn Project: //WorksOn
Optional facts (変換先スキーマ T)
Person: //Person Prof: //Person (*) Student: //Person

図 7　ベンチマーク例: 知識表 K

率的に D を作成できる知識.

図 7 に indispensable facts と optional facts の例を示す．まず，indispensable facts を説明する．indispensable facts は自然言語で与えられる．indispensable facts の例としては「student の salary は student が働いている project の時間と時給を乗算したものである」がある．これを知らなければ，どのように学生 (student) の給与 (salary) を計算するかをツールの利用者が正しく指定することを保証できない.

次に，optional facts を説明する．Optional facts は，データベーススキーマ S および T に対してそれぞれ存在する．S(あるいは T) のための各 optional facts は，クラス図 C の各クラスおよび関連が，S(あるいは T) の各構成要素のそれぞれどの部分に対応するかを表している．本質的には，optional facts は B^S および B^T 中の binding expression をより簡単にしたものである．facts は (拡張) リレーショナル代数と XPaht 式を組み合わせた式として表現される．例えば，図 7 の変換元スキーマ S に関する最初の optional facts は，図 5(a) の Person クラスの構成要素のインスタンスが，Prof 要素の集合と，Student 要素，PayRate 要素，および WorksOn 要素を join した結果集合の outer-union(注1)で計算されることを表している．このように，optional facts に含まれる情報は，データベーススキーマやインスタンスなどを精査することで知ることができるが，あらかじめ与えられていればデータ変換記述 D の作成のヒントとして利用することができるものである.

クラス図 C と同じように，知識表 K の一部 K_n をプロジェクション数 n で表すことができる．K_n は次の facts で構成される．(1) すべての indispensable facts．(2)n で表されるクラス図 C 中のクラスと関連に対応する optional facts．例えば，図 5(a) に示すクラス図 C が与えられた場合，$n = 3FF0F$ とすると n を 2 進数表記したときの 5 番目から 8 番目のビットは 0 であり，したがって，K_{3FF0F} には，C 中で identifier が 5 から 8 の構成要素に対応する optinal facts(Prof クラスに関する optional facts．図 7 中では (*) 記号で表されている) は含まれない.

3.5　部分問題の導出方法

本章の始めで説明したように，$cost(D)$ はデータベーススキーマのサイズや複雑さに大きく影響を受けるので，ベンチマークに様々なサイズや複雑さを持つ問題を含ませることが重要である．本提案フレームワークでは，ベンチマークでは (ある程度大きな) 問題を 1 つ定義しておき，そこから様々な部分問題を柔軟に導出できるようにする．このように設計した理由は次の通りである．(1) データ変換支援ツールには，それらの機能を記述するための標準化された言語やボキャブラリが存在しないので，多様な問題の集合をあらかじめ定義しておくことが困難である．一方，RDBMS においては，問合せの表現能力を記述するための標準化された言語として SQL が存在する．したがって，RDBMS の問合せ実行性能を評価するためのベンチマークに含まれる各問合せの性質は SQL の構成要素を用いて表現できる．例えば,「この問合せは equijoin を含む.」などである．(2) データ変換の問題を構築することは非常に知的な作業であり，自動化することが非常に困難である．したがって，部分問題ではなく「より複雑な問題」を自動生成することは現実的ではない.

以下では，部分問題の導出方法について説明する．例として，図 4 から図 7 に示すベンチマーク $BM_{full} = (S, I^S, T, K, C, P^C, B^S, B^T)$ の部分問題を導出するとする.

最初に，クラス図 C の一部を表すプロジェクション数 n が与えられたとき，部分問題 BM_n を導出する方法を説明する．例えば，$n = 3FF0F$ とした場合，部分問題 $BM_{3FF0F} = (S_{3FF0F}, I^S_{3FF0F}, T_{3FF0F}, K'_{3FF0F}, C_{3FF0F}, P^C_{3FF0F}, B^S_{3FF0F}, B^T_{3FF0F})$ は次のように導出される.

（1） C_{3FF0F} は自明である (図 8 に示す)．同様に，P^C_{3FF0F} は自明である．B^S および B^T を構成する binding expression は，C の各構成要素 (クラス，関連，および属性) ごとに存在するので，B^S_{3FF0F} および B^T_{3FF0F} も自明である.

（2） S_{3FF0F} は，S から B^S_{3FF0F} 中の binding expression において参照されないスキーマの構成要素 (XML の要素および属性) を除去したものである (T_{3FF0F} も同様である)．この例では，S_{3FF0F} および T_{3FF0F} は図 9 のようになる．元の S や T と比較すると，Prof 要素およびその属性は，対応する binding expression から参照されないので除去されている.

もし，除去される構成要素が XML スキーマの葉ではない場合は，除去される構成要素の子は除去される構成要素の親の子になる．例えば，ある DTD { `A = (B, C), C = (D, E)` } から `C` を除去する場合は，{ `A = (B, D, E)` } となる．XML スキーマの root 要素が除去される場合は，特別な要素「Data」を代わりの root 要素とする.

（3） I^S_{3FF0F} は，I^S から S_{3FF0F} に無関係な部分を除去することで計算される.

（4） K'_{3FF0F} は，B^S_{3FF0F} および B^T_{3FF0F} に従って，無関係な部分を K から除去することで得られる．K'_{3FF0F} は，3.4 節で説明した K_{3FF0F} とは次のように異なる．すなわち，K_{3FF0F} は単にプロジェクション数 $n = 3FF0F$ に無関係な optional facts を除去したものであるが，K'_{3FF0F} は C_{3FF0F}(図 8) と矛盾がないように K_{3FF0F} の indespensable facts を修正したものである．この例では，K_{3FF0F} は図 7 の K から (*) で表されている Prof クラスに関する optional facts を除去したものであ

(注1)：⊎ は，outer-union 演算を表している．この演算は，和両立を必要としない.

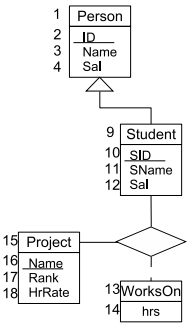


図 8　C_{3FF0F}

```
Data = (Student*, PayRate*, WorksOn*)
Student = (@SID:ID, SName)
PayRate = (@Rank:ID, HrRate)
WorksOn = (@SID:IDREF, @Project:ID, Hrs, @ProjRank:IDREF)
```

(a)

```
Data = (Person*)
Person = (ID, Sal)
```

(b)

図 9　S_{3FF0F}(a) および T_{3FF0F}(b)

る．一方，K'_{3FF0F} は，K_{3FF0F} の最初の indispensable facts を「person は student である」に修正したものとなる．なぜなら，C_{3FF0F} 中には Prof クラスが存在しないからである．

以上で，BM_n の導出方法の説明は終わりである．次に，ツールの特定機能を利用しないような部分問題の導出方法を説明する．例えば，ツールが算術演算を扱えない場合に，算術演算を含まないような部分問題 BM_{-arith} を導出する方法を説明する．BM_{-arith} は BM_{full} から次のように導出される．

（1） B^S および B^T 中で，算術演算が必要な binding expression を同定する．次に，C からその binding expression に関係のある構成要素 (クラス，関連，および属性) を除去する．この例では，student の salary を計算するために算術演算が必要であるため，C から identifier が 4 あるいは 12 である Sal 属性を除去する (図 5(a))．

（2） (1) で得られたクラス図を C' とする．このとき，n を $C' = P^C(n)$ となるようなプロジェクション数とする．ここで，$P^C : Int \rightarrow ClassDiagram$ は n を用いて C の部分クラス図を導出するプロジェクション関数である．図 5(a) では，4 番目，12 番目のビットが 0 なので，$n = 3F7F7$ である．

（3） BM_{-arith} は BM_{3F7F7} と等しい．BM_{3F7F7} の導出方法は先に説明した通りである．

3.6　データ変換問合せの検証

あるベンチマーク $BM = (S, I^S, T, K, C, P^C, B^S, B^T)$ が与えられたとき，ツールにより出力されたデータ変換問合せ F が正しいか否かは次のように検証できる．

（1） $F(I^S)$ を実行し，$I^T = F(I^S)$ を得る．

（2） $B^T(I^T)$ を計算し，C のインスタンスを得る．

（3） $B^S(I^S)$ を計算し，C のインスタンスをもう 1 つ得る．

（4） $B^T(I^T) = B^S(I^S)$ が成立すれば F は正しい．

一見，このようなことをしなくとも，あらかじめベンチマーク BM で正しいデータ変換問合せを用意しておき，ツールが出力した問合せと比較すれば，わざわざ出力された問合せを実行する必要がないようにみえるかも知れない．しかしながらこのように設計した理由は次である．(1) 正しい結果を出力する問合せは複数存在するが，それらが等価であると証明することは自明ではない．(2) ある問題とその正しい問合せが与えられたとする．このとき，この問題の部分問題のための正しい問合せの導出が自明ではない．

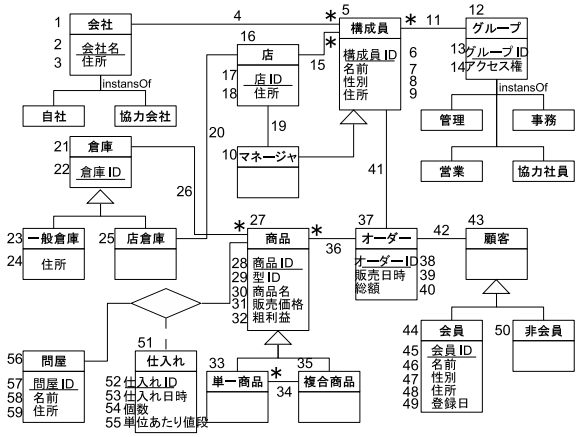


図 10　サンプルベンチマークの概念モデル C

4.　ベンチマークの適用例

本章では，提案フレームワークを利用して構築したサンプルベンチマークの一例を説明する．また，それを用いて代表的なデータ変換支援ツールである Clio [2] を評価した実験の結果を示す．

構築したサンプルベンチマーク $BM_{sample} = (S, I^S, T, K, C, P^C, B^S, B^T)$ は，ある家電量販店の営業に関するデータを管理するデータベース中のデータを，新システムのデータベースに移行するためのデータ変換を示したものである．BM_{sample} の概念モデル C を図 10 に示す．この家電量販店では，様々なデータをデータベース中に格納している．例えば，構成員 (社員) のデータ，会員登録されている顧客のデータ，商品の販売データ，商品の仕入れデータなどである．C を実装したデータベーススキーマ (変換元 S および変換先 T) の主な特徴は次である (データベーススキーマの全体を含むベンチマークの詳細は付録に示す)．

- この家電量販店では，2000 年に顧客の会員登録を開始し，2000 年以前と以後で販売記録を分けている．
- 1 つの商品ごとにしていた販売記録を，新データベースでは 1 回のオーダーごとにする．
- 新データベースではデータベーススキーマの構成要素に商品の粗利益を含ませ，商品の販売価格と仕入れ価格から計算させることにする．

4.1　サンプルベンチマークを用いたデータ変換支援ツールの評価実験

本節では，このサンプルベンチマークを用いてデータ変換支援ツール Clio [2] の評価実験を行った結果を示す．なお，Clio のコードが入手できなかったため，パワーポイントを用いて

	特徴			Clio の結果評価		
	要素数	missing	optional	$cost(D)$	$cost(D)$/要素数	間違い
BM_{NO_1}	14	なし	あり	220.7	15.8	0
BM_{NO_2}	53	なし	あり	643.7	12.1	0.3
BM_{NO_3}	27	あり	あり	533.7	19.8	0
BM_{NO_4}	30	なし	なし	765.3	25.5	3.7

図 11　サンプルベンチマークの適用結果 (アンダーラインは各部分問題の特徴を示す)

Clio のシミュレータを作成し，実験に用いた．実験の手順を次に示す．

（1） サンプルベンチマーク BM_{sample} を基に，それぞれ特徴を持たせた 4 つの部分問題を作成する．

（2） 被験者 12 人をそれぞれの部分問題に対してランダムに 3 人ずつ割当てる．

（3） 各部分問題の知識表を被験者に示す．

（4） 変換元スキーマ S と変換先スキーマ T を画面に表示し，被験者はデータ変換記述 D を作成する．図 2 に示したように，Clio の利用者による D の作成は，本質的には，異なる 2 つのデータベーススキーマに関して，次を行うことである．(1) 対応する構成要素を結ぶ線を引く，(2) 必要ならば計算式を指定する．D を作成する過程で，被験者が D の指定を間違った場合には，間違えていることを指摘し修正させる．

（5） (4) の様子をビデオで撮影する．このビデオを解析し，被験者が D を作成するのにかかった時間 $cost(D)$ を計測する．また，間違いをカウントする．

本実験では，ツールを様々な側面から評価するために，サンプルベンチマーク BM_{sample} を基に異なる特徴を持たせた 4 つの部分問題を作成した．部分問題ごとの特徴を図 11 に示す．図 11 において，要素数とは，変換先スキーマ S と変換元スキーマ T に含まれる，対応付けの対象となる構成要素の数である．missing とは，図 5(a) の Student クラスの Sal 属性のように，概念モデル C には存在するが，S の構成要素としては存在しない属性 (図 4(a)) が存在するか否かである．optional とは，optional facts を利用者に提示するか否かである．

各部分問題の性質を次に説明する (各部分問題のプロジェクション数 n については付録に示す)．まず，部分問題 BM_{NO_1} は，比較的単純な部分問題である．部分問題 BM_{NO_2} は，対応付けの対象となるスキーマの構成要素が増加したときの影響をみるための部分問題である．部分問題 BM_{NO_3} は，S の構成要素としては存在しない，図 10 の C の粗利益を計算させる算術演算を含むような部分問題である (この部分問題については，付録に 3.5 節で説明した導出方法を示す)．部分問題 BM_{NO_4} は，optional facts の影響をみるための部分問題である．

4.2 考　　察

実験結果を図 11 に示す．$cost(D)$ は，被験者がデータ変換記述 D を作成するのにかかった時間 (秒単位) の平均である．また「間違い」は，被験者の D 作成における間違いの数の平均である．

この実験結果より次のことがわかる．BM_{NO_1} と BM_{NO_2} を比較することにより，属性数が増加すると $cost(D)$ も増加する．BM_{NO_1} と BM_{NO_3} を比較することにより，1 対 1 対応でない要素が存在すると，S 中のどの構成要素から求めるべき構成要素を計算するのかを被験者が発見しなければならず，$cost(D)$ は増加する．BM_{NO_1} と BM_{NO_4} を比較することにより，利用者が対象となるデータベーススキーマに関して知識を持たない場合，$cost(D)$ は増加することがわかる．また，多くの間違いが起こることがわかる．

本稿で提案したフレームワークでは，あらかじめツールの利用者の知識を制御するための仕組みである知識表が含まれているが，会社などで実際にデータ変換を行う場合は，知識表は存在しない．したがって，もし，ツールの利用者がデータベーススキーマやデータベースインスタンスに詳しくない場合は，これらを精査することで optional facts と同等の情報を得なければならない．これが大変な労力であることは，BM_{NO_4} の $cost(D)$ が本実験で最も大きいことからもわかる．さらに，BM_{NO_4} で多くの間違いが起こることから，大変な労力をかけても利用者は D の作成を間違いやすいことがわかる．

5. 関 連 研 究

XML の普及により，複雑なスキーマ間のデータ変換への要求は増大している．そこで，多数のデータ変換支援ツールが開発されてきている [2] [3] [4]．しかし，それらのツールを利用してデータ変換問合せを作成する効果を比較するための体系的なフレームワークは存在しなかった．我々は，本研究がこの問題を解決するための第 1 段階であると考えている．

DB エンジンの問合せ実行性能を評価するために TPC-C [6] および TPC-H [7]，XML エンジンのために XMark [8]，XBench [11] などの様々なベンチマークが存在する．それぞれ目的は異なっているものの，これらのベンチマークはすべて式 1 で記述されるような共通のフレームワークに基づいている．一方，データ変換支援ツールは全く異なったフレームワークを必要としており，本研究はその要求を満たすべく設計したものである．

6. おわりに

本稿では，データ変換支援ツールの性能を評価するためのベンチマーク構築のフレームワークを提案した．データ変換支援ツールの性能の評価は，計測されるコストに影響を与える要因が既存の DBMS の問合せ実行性能の評価とは全く異なっているため，簡単ではない．提案フレームワークでは，この問題を解決するために，利用者に提示される知識およびデータベーススキーマのサイズやそれらの間の対応関係などの，データ変換の問題問合せの作成効率に影響を与える要因を変更するための仕組みを持たせた．今後の課題としては，提案フレームワークを利用して，Clio だけでなく，様々なツールの性能を比較評価することが挙げられる．

謝　　辞

ゼミなどでご議論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田畑孝一教授，阪口哲男助教授，および永森光晴講師に感謝いたします．本研究の一部は日本学術振興会科学

データ = (会社+)
会社 = (@会社名:ID, 住所, 構成員+, 店+, 倉庫+, 商品*, 販売記録:2000 年以前*, 販売記録*, 会員*, 問屋*, 仕入れ*)
構成員 = (@構成員 ID:ID, 名前, 性別, 住所, グループ)
グループ = (@グループ ID:ID, アクセス権)
店 = (@店 ID:ID, 性別, 構成員 ID+, マネージャID, 倉庫 ID*)
倉庫 = (@倉庫 ID:ID, 住所)
商品 = (@商品 ID:ID, 型 ID, 商品名, 販売価格, 構成商品*, @倉庫 ID:IDREF)
販売記録:2000 年以前 = (@商品 ID:IDREF, 販売日時, @構成員 ID:IDREF)
販売記録 = (@商品 ID:IDREF, 販売日時, @構成員 ID:IDREF, @会員 ID:IDREF?)
会員 = (@会員 ID:ID, 名前, 性別, 住所, 登録日)
問屋 = (@問屋 ID:ID, 名前, 住所)
仕入れ = (@仕入れ ID:ID, 仕入れ日時, 個数, 単位あたり値段, @問屋 ID:IDREF, @商品 ID:IDREF)

(a)

データ = (会社+, 構成員+, マネージャ+, グループ+, グループ所属+, 店+, 一般倉庫*, 店倉庫+, 単一商品*, 複合商品*, 含む*, オーダー*, オーダー商品*, 会員*, 問屋*, 仕入れ*)
会社 = (@会社名:ID, 住所)
構成員 = (@構成員 ID:ID, 名前, 性別, 住所, @会社名:IDREF, @店 ID:IDREF)
マネージャ = (@構成員 ID:IDREF, @店 ID:IDREF)
グループ = (@グループ ID:ID, アクセス権)
グループ所属 = (@グループ ID:IDREF, @構成員 ID:IDREF)
店 = (@店 ID:ID, 住所, @倉庫 ID:IDREF)
一般倉庫 = (@倉庫 ID:ID, 住所)
店倉庫 = (@倉庫 ID:ID, 住所)
単一商品 = (@単一商品 ID:ID, 型 ID, 商品名, 販売価格, 粗利益, @倉庫 ID:IDREF)
複合商品 = (@複合商品 ID:ID, 型 ID, 商品名, 販売価格, 粗利益, @倉庫 ID:IDREF)
含む = (@複合商品 ID:IDREF, @単一商品 ID:IDREF)
オーダー = (販売日時, 総額, @構成員 ID:IDREF, @会員 ID:IDREF)
オーダー商品 = (販売日時, @構成員 ID:IDREF, @会員 ID:IDREF)
会員 = (@会員 ID:ID, 名前, 性別, 住所, 登録日)
問屋 = (@問屋 ID:ID, 名前, 住所)
仕入れ = (@仕入れ ID:ID, 仕入れ日時, 個数, 単位あたり値段, @問屋 ID:IDREF, @商品 ID:IDREF)

(b)

図 A·1　サンプルベンチマークの変換元スキーマ S(a) および変換先スキーマ T(b)

研究費補助金若手研究 (B)(課題番号 15700108) による．

文　　献

[1] Serge Abiteboul, Rakesh Agrawel, Phil Bernstein, et al.: The Lowell Database Research Self-Assessment. Communications of the ACM. 48(5): 111-118 (2005)

[2] Laura M. Haas, Mauricio A. Hernandez, Howard Ho, Lucian Popa, Mary Poth: Clio Grows Up: From Research Prototype to Industrial Tool. SIGMOD Conference 2005: 805-810

[3] Jayant Madhavan, Philip A. Bernstein, Erhard Rahm: Generic Schema Matching with Cupid. VLDB 2001: 49-58

[4] Atsuyuki Morishima, Toshiaki Okawara, Jun'ichi Tanaka, Ken'ichi Ishikawa: SMART: A Tool for Semantic-Driven Creation of Complex XML Mappings. SIGMOD Conference 2005: 909-911

[5] Erhard Rahm, Philip A. Bernstein: A survey of approaches to automatic schema matching: VLDB J. 10(4): 334-350 (2001)

[6] Transaction Processing Performance Council. An On-Line Transaction Processing Benchmark. http://www.tpc.org/tpcc/

[7] Transaction Processing Performance Council. An Ad-Hoc, Decision Support Benchmark. http://www.tpc.org/tpch/

[8] Albrecht Schmidt, Florian Waas, Martin Kersten, Michadl J. Garey, Ioana Manolescu, Palph Busse: XMark: A Benchmark for XML Data Management, VLDB 2002: 974-985

[9] World Wide Web Consortium. XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). http://www.w3.org/TR/xhtml1/

[10] World Wide Web Consortium. XQuery 1.0: An XML Query Langrage. http://www.w3.org/TR/xquery/

[11] Benjamin Bin Yao, M. Tamer Özsu, Nitin Khandelwal: XBench Benchmark and Performance Testing of XML DBMSs, ICDE 2004: 621-632

付　　録

1. サンプルベンチマークの詳細

本章では，サンプルベンチマークの変換元スキーマ S と変換先スキーマ T を図 A·1 に示し，また，サンプルベンチマークの各部分問題を表すプロジェクション数 n を図 A·2 に示す．

BM_{NO_1}: $n = 000000000003DF0$
BM_{NO_2}: $n = 7FC00057FF03DFF$
BM_{NO_3}: $n = 7FC0001FC000000$
BM_{NO_4}: $n = 001FF597C0000F0$

図 A·2　各部分問題を表すプロジェクション数 n

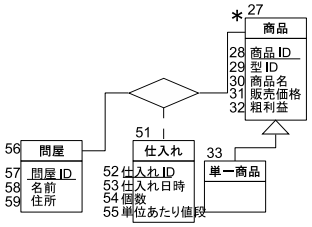


図 A·3　部分問題 BM_{NO_3} のクラス図 $C_{7FC0001FC000000}$

データ = (商品*, 問屋*, 仕入れ*)
商品 = (@商品 ID:ID, 型 ID, 商品名, 販売価格)
問屋 = (@問屋 ID:ID, 名前, 住所)
仕入れ = (@仕入れ ID:ID, 仕入れ日時, 個数, 単位あたり値段, @問屋 ID:IDREF, @商品 ID:IDREF)

(a)

データ = (単一商品*, 問屋*, 仕入れ*)
単一商品 = (@単一商品 ID:ID, 型 ID, 商品名, 販売価格, 粗利益)
問屋 = (@問屋 ID:ID, 名前, 住所)
仕入れ = (@仕入れ ID:ID, 仕入れ日時, 個数, 単位あたり値段, @問屋 ID:IDREF, @商品 ID:IDREF)

(b)

図 A·4　部分問題 BM_{NO_3} の変換元スキーマ $S_{7FC0001FC000000}$(a) および変換先スキーマ $T_{7FC0001FC000000}$(b)

Indispensable facts
商品とは単一商品である 商品の粗利益は商品の販売価格から仕入れ価格を減算したものである
Optional facts (変換元スキーマ S)
商品: //商品 ⋈ //仕入れ 問屋: //問屋 仕入れ: //仕入れ
Optional facts (変換先スキーマ T) 商品: //単一商品 問屋: //問屋 仕入れ: //仕入れ

図 A·5　部分問題 BM_{NO_3} の知識表 K

2. 部分問題の導出方法の例

本章では，部分問題 BM_{NO_3} の導出方法を順に説明する．なお，BM_{NO_3} は算術演算を含む問題である

（1） 算術演算 (この例では，商品の粗利益を求めるための計算) が必要であるように，クラス図 $C_{7FC0001FC000000}$ を BM_{full} の C から導出する (図 A·3 に示す)．$P^C_{7FC0001FC000000}$，$B^S_{7FC0001FC000000}$，および $B^T_{7FC0001FC000000}$ は自明である．

（2） 上で求めた $C_{7FC0001FC000000}$ から，変換元スキーマ $S_{7FC0001FC000000}$ および変換先スキーマ $T_{7FC0001FC000000}$ を導出する (図 A·4)．この例では，$T_{7FC0001FC000000}$ にある粗利益を求めるために，$S_{7FC0001FC000000}$ にある商品の販売価格から仕入れの単位あたり値段を減算する．

（3） $I^S_{7FC0001FC000000}$ は $S_{7FC0001FC000000}$ から自明である．

（4） $K_{7FC0001FC000000}$ は図 A·5 に示す通りになる．すなわち，K から $n = 7FC0001FC000000$ に無関係な optional facts を除去し，$C_{7FC0001FC000000}$ と矛盾がないように indispensable facts を修正する．

（5） 以上で，$BM_{NO_3} = BM_{7FC0001FC000000}$ が導出される．