# ベンチマークによるデータ変換支援ツール評価の一手法

## A Benchmark Approach to the Evaluation of Data Exchange Tools

大河原 俊明♡ 森嶋 厚行♢ 杉本 重雄♠

Toshiaki OKAWARA Atsuyuki MORISHIMA Shigeo SUGIMOTO

**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．しかし，これらのツール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．本稿では，ベンチマークを用いてデータ変換支援ツールを評価するための一手法を提案する．**

**Data exchange between different databases has increased its importance, and many tools to help data exchange have been developed. However, there has been no discussion on how to evaluate the effectiveness of such tools. This paper proposes a new framework to develop benchmark programs for evaluating data exchange support tools.**

## 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 等のような，構造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．関数 $F$ 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．さらに，XML の普及にしたがって，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．

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 [1][2][3]．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．

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．

♡ 学生会員 筑波大学大学院 図書館情報メディア研究科
okwr@slis.tsukuba.ac.jp
♢ 正会員 筑波大学大学院 図書館情報メディア研究科
mori@slis.tsukuba.ac.jp
♠ 正会員 筑波大学大学院 図書館情報メディア研究科
sugimoto@slis.tsukuba.ac.jp

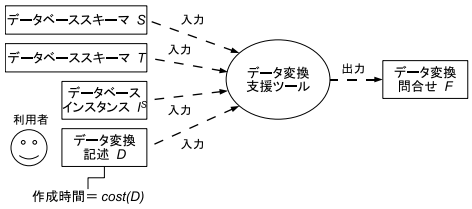



**図 1** データ変換支援ツールの概要
Fig. 1 Overview of a Data Exchange Tool

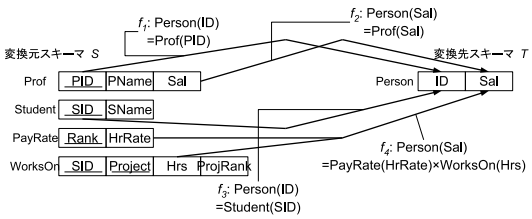



**図 2** Clio におけるデータ変換記述 $D$
Fig. 2 Data Exchange Description $D$ for Clio

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 [1] 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)Person の Sal 属性が PayRate の HrRate 属性 (アルバイトの時給) と WorksOn の Hrs 属性 (時間) に対応し，(2) その値は 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，$f_1, f_2, f_3$ 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，このようなベンチマークはこれまで存在しなかった．

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構築手法を提案する．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．

## 2. 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．

$$BM = (S, g, Q) \quad (1)$$

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．

DB エンジンの問合せ実行性能を評価するためには各種 TPC ベンチマーク，XML エンジンのためには XMark [4]，XBench [5] 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．**(a) 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，ベ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．**(b) 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．**(c) 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 (b)(c) 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 3(a))，$I^S$ は $S$ のあるインスタンス (図 3(b))，および $T$ は変換先スキーマ (図 3(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 6) 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 $(C, P^C, B^S, B^T)$ は，$BM$ の部分問題を導出する際に用いられる．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を表すクラス図 (例えば，図 4(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 3 から図 6 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 3(a) に示す変換元スキーマ $S$ と図 3(c) に示す変換先スキーマ $T$ は図 4(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 4(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 4(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

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)

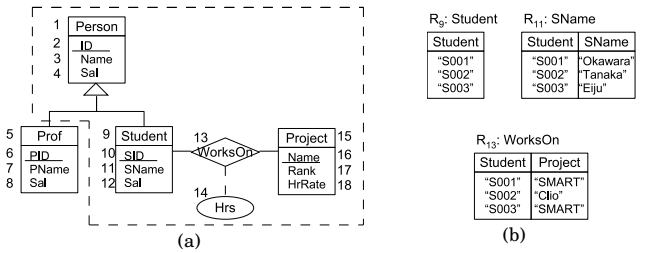
```
<Data>
  <Prof PID="P001"><PName>Morishima</PName>
    <Sal>7000</Sal></Prof>
  <Prof PID="P002"><PName>Sugimoto</PName>
    <Sal>9000</Sal></Prof>
  <Student SID="S001"><SName>Okawara</SName></Student>
  <Student SID="S002"><SName>Tanaka</SName></Student>
  <Student SID="S003"><SName>Eiju</SName></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 3** 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)
Fig. 3 Example of Benchmark: Source Schema $S$ (a), Instance $I^S$ of $S$ (b), and Target Schema $T$ (c)




(a) (b)

**図 4** 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)
Fig. 4 Example of Benchmark: Conceptual Schema $C$ (a) and Fragment of an Instance of $C$ (b)

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．$P^C$ 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 $B^S : S \rightarrow C$(あるいは $B^T : T \rightarrow C$) 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 5) の集合で表される．下記では，まずクラス図のインスタンス，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．$C$ 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，クラス図 $C$ のインスタンスを計算できる．例えば，図 4(b) は図 3(b) に示すインスタンスから計算されたクラス図 (図 4(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 4(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 4(b) $R_{13}$)

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

**図 5** ベンチマーク例: $B^S$ 中の binding expressions の一部
Fig. 5 Example of Benchmark: Some of Binding Expressions in $B^S$

| Indispensable facts |
|---|
| person は professor あるいは student である<br>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)<br>Prof: //Prof (*)<br>Student: //Student ⋈ //PayRate ⋈ //WorksOn<br>WorksOn: //WorksOn<br>Project: //WorksOn |
| **Optional facts (変換先スキーマ $T$)** |
| Person: //Person<br>Prof: //Person (*)<br>Student: //Person |

**図 6** ベンチマーク例: 知識表 $K$
Fig. 6 Example of Benchmark: Knowledge Table $K$

- **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 4(b) $R_{11}$)

**binding expression** の説明: ベンチマークに含まれる関数 $B^S : S \rightarrow C$ 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:

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

ここで，$i$ はクラス図 $C$ の各構成要素が持つ identifier であり，$expr_{S \rightarrow i}$ 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 風の構文を持つ．例えば，図 5 は図 3(a) に示す変換元スキーマと図 4(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 5 において，$expr_{S \rightarrow 11}$ 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 5(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指定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，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 種類に分類される．

**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，利用者が効率的に $D$ を作成できる知識．

図 6 に indispensable facts と optional facts の例を示す．まず，indispensable facts を説明する．各 indispensable fact は自然言語で与えられる．indispensable fact 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 fact 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す．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．各 fact 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 6 の変換元スキーマ $S$ に関する最初の optional fact は，図 4(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 要素の各集合の結合演算結果の outer-union[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．

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 4(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 6 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

本節では，部分問題の導出方法について説明する．例として，図 3 から図 6 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．

まず，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 (図 4(a) の点線で囲まれた部分) である．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 3(a)(c) から下線部を除去したものになる．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 4(a) の点線で囲まれた部分) 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 6 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る．一方，$K'_{3FF0F}$ は，$K_{3FF0F}$ の最初の indispensable fact 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．

同様に，ツールの特定機能を利用しないような部分問題の導出もできる．例えば，ある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 $BM_{-arith}$ は，$BM_{full}$ の構成要素のうち算術演算を含まない部分だけを表すプロジェクション数 $n$ を求め，$BM_n$ を導出すればよい．

[1] ⊎ は，和両立を必要としない outer-union 演算を表す．

## 4. 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．

## [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研究費補助金若手研究 (B)(課題番号 15700108) による．

## [文献]

[1] 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

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

[3] 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

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

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

**大河原 俊明 Toshiaki OKAWARA**
筑波大学大学院 図書館情報メディア研究科博士前期課程修了．日本データベース学会学生会員．

**森嶋 厚行 Atsuyuki MORISHIMA**
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助教授．1998 年 筑波大学大学院 工学研究科修了．博士 (工学)．ACM，IEEE-CS，情報処理学会，電子情報通信学会，日本データベース学会各正会員．

**杉本 重雄 Shigeo SUGIMOTO**
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教授．京都大学大学院 工学研究科電子工学専攻博士後期課程修了．工学博士．ACM，IEEE-CS，情報処理学会，日本データベース学会各正会員．