データベースに対する一括更新の正当性の検査方法

鈴木卓治† 小林光夫††

データベースの仕様を意味データモデルで与えることにより，データの抽象的な意味構造を明確にすることができる．しかるに，意味データモデルにおけるデータの一貫性や安全性の研究は，関係データモデルにおけるそれに比べて遅れている．本論文では，仕様が意味データモデルで与えられたデータベースが更新に対して一貫性制約を満たすことを，更新前に検査する方法について述べる．更新の正誤だけでなく，利用者に正しい更新をうながすための情報も抽出できることを示す．

A Verification Algorithm on an Updating Transaction of a Database with Integrity Constraints

Takuzi Suzuki† and Mituo Kobayasi††

This paper describes a verification algorithm of a transaction for updating a database. Here, the specification of the database is given in term of a semantic data model. The algorithm checks whether the transaction violates the integrity constraints, and also gives us useful information to construct a valid transaction.

1. はじめに

データベースがその運用中常に満たしているべき条件を仕様として与えるとき，蓄えるデータの種類，性質，データ間の関連などに着目して記述したものをデータモデル (data model) という．

データモデルの考え方は，ネットワークデータモデルや関係データモデル[1]から意味データモデル (semantic data model)[2),3)]へと，データベースの抽象的な意味構造を記述する方向に進歩してきた．データベースの更新や検索のためのデータベース管理システム (database management system, DBMS) も，O_2[4)] などのオブジェクト指向 DBMS のように，意味データモデルによる仕様記述を受け付けより複雑な構造をもつデータを扱えるものへと進歩している．OMT (object modeling technique)[5)] のように，意味データモデルを一般の情報処理システムの設計に積極的に取り込むアプローチも盛んになりつつある．

† 国立歴史民俗博物館情報資料研究部
Museum Science Department, National Museum of Japanese History
†† 電気通信大学情報工学科
Department of Computer Science and Information Mathematics, University of Electro-Communications

われわれは，既存の仕様記述法に広く適用可能な意味データモデルの厳密な定義を与え，その上で，データベースのより柔軟かつ安全な運用のための基礎技術を確立することを目指している．たとえばデータベースおよびその利用者インタフェイスの自動生成[6),7)]などがその例としてあげられる．

本論文では，データベースが常に意味データモデルで与えられた仕様を満たすように更新されるかどうか，すなわち一貫性制約 (integrity constraints) を満たすかどうかを，更新前に検査する方法について述べる．一貫性制約の問題は，関係データモデルの上ではよく研究されているが，意味データモデルの上での研究はわずかしかなく[8)]，検査プログラムが作成できるほどの詳細なアルゴリズムは示されていない．更新後に検査する方法では DBMS 側に更新破棄のための特別な仕組みが必要なのに対し，本論文の方式では不要である．

2章では，本論文で用いるデータベースの仕様の記述法を与える．意味データモデルの性質を厳密かつ一般的に扱うために数学の記法を用いる．既存の仕様記述法における諸概念との対応は2.4節に述べる．3章では，更新が正しく行われるかどうかを判定するアルゴリズムを示す．データベースを書き換えることなく，1回の更新で追加あるいは削除されるデータを検

査することで判定が可能であることを示す．4章では，3章で示したアルゴリズムが有効に働くことを，実際に作成した検査プログラムの実行結果により示す．更新の正誤のみでなく，利用者に正しい更新を促すための情報を抽出できることをあわせて示す．

2. データベースの仕様

これ以降，**図1**に示す仕様をもつ学生データベースを仕様記述の具体例として用いる．

学生データベースは，在籍する学生のデータを含んでいる．それぞれの学生について，学籍番号（整数），氏名（文字列），住所（文字列）がわかる．学籍番号と氏名は必ずわかるが，住所はわからない場合もある．学籍番号は一意に定めなければならない．学生は学部生や大学院生のどちらかである．大学院生については所属（文字列）が必ずわかる．学生の中には下宿生がいる．下宿生によっては帰省先（文字列）がわかる．

学生データベースは，履習可能な科目のデータを含んでいる．それぞれの科目について科目名（文字列）および単位数（整数）がわかる．科目名および単位数はすべての科目について必要である．さらに科目名は一意に定めなければならない．

学生データベースは，学生の科目履修のデータを含んでいる．1人の学生について，履修は1つもないこともあるし，複数あることもある．履修によっては，その成績(整数)がわかるものがある．在籍していない学生のデータや存在しない科目のデータは，履修データには含まれないものとする．

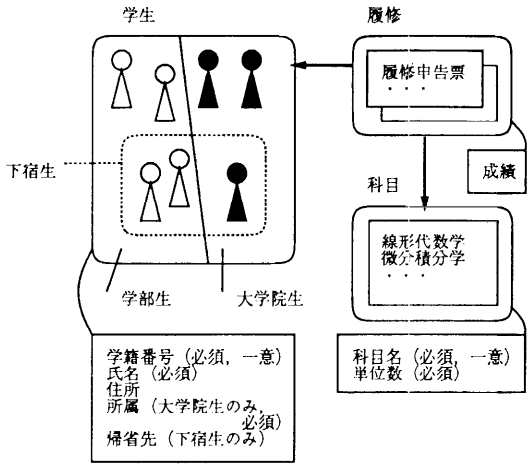


図1 学生データベースの例
Fig. 1 An example of a database of students.

2.1 集合の種類

データベースは，性質によって分類されたデータの集合の集まりである．同じ性質をもつデータの集合を1つの識別子で表すことにしよう．このような識別子の有限集合を I とし，データベースが蓄えうるすべての集合の集合を D とする．そしてデータベースを，識別子を与えるとそれが表すデータの集合を返す関数 $d: I \to D$ として定義する．$\forall a \in I$ に対し，データの集合 $d(a)$ のとりうる範囲 $[a] \in D$ があらかじめ定められているものとしよう．

定義1（*DB* の定義） $d: I \to D$ が，$\forall a \in I$ に対して $d(a) \subseteq [a]$ を満たすとき，このような d の集合を $DB(\subseteq I \to D)$ と定める．DB の要素を，組 $(D, [\cdot], I)$ を仕様とするデータベース，あるいは単にデータベースという．

学生データベースの例では，仕様に含まれる単語を用いて I を次のように定める．D および $[\cdot]$ は適当に与えられているものとする．

I = {学生，学籍番号，整数，氏名，文字列，
住所，学部生，大学院生，所属，下宿生，
帰省先，科目，科目名，単位数，履修，
成績}． (1)

[学生] は，すべての学生を要素として含む集合を表す．d (学生) は，データベース d が有する学生集合のデータを表す．DB は，学生データベースがその運用中にとりうる状態の全体を表す．

データベースに蓄えられるデータの集合は，データベースの更新によって変化しうる可変集合と変化しない不変集合とに分けられる．さらに可変集合は，集合間の関係を表す関係集合とそうでない（それ以上分解できない）集合とに分けられる．関係集合でない可変集合を，以下では単に可変集合ということにする．不変集合，可変集合，関係集合を表す識別子の集合を，それぞれ I_C, I_V, I_R (I_C, I_V, I_R は互いに素) とする．$I = I_C \cup I_V \cup I_R$ である．

学生データベースの例では，

I_C = {整数，文字列}， (2)

I_V = {学生，学部生，大学院生，下宿生，科目}， (3)

I_R = {学籍番号，氏名，住所，所属，帰省先，
科目名，単位数，履修，成績} (4)

となる．

データベース d の不変集合および可変集合が満たすべき性質を次に示す．

不変集合の性質：

$$\forall a \in I_C \text{ に対して } d(a)=[a]. \quad (5)$$

可変集合の性質：

$$\forall a \in I_V \text{ に対して } d(a) \subseteq [a]. \quad (6)$$

学生データベースの例では，

d(整数)＝[整数],

d(学生)⊆[学生]

などが成り立つ．

本論文では，簡単のため，関係を2項関係に限定する．集合 A, B 間の2項関係 $R \subseteq A \times B$ に対して，A を R の定義域集合，B を R の値域集合ということにする．定義域集合および値域集合を表す識別子を関係集合の識別子から求める関数を，それぞれ Dom, $Ran : I_R \to I$ とすると，識別子 a が表す関係集合の定義域集合は $[Dom(a)]$，値域集合は $[Ran(a)]$ と表せる．

データベース d の関係集合が満たすべき性質を次に示す．

関係集合の性質：

$$\forall a \in I_R \text{ に対して } d(a) \subseteq [a]. \quad (7)$$

ただし，$[a]=[Dom(a)]\times[Ran(a)]$ は成り立っているものとする．また，関係集合の定義域集合および値域集合にその関係集合自身が含まれるような定義を避けるため，$\forall a \in I_R$ に対して Dom および Ran を有限回適用して再び a になることはないものとする．

学生データベースの例では，

$$\begin{cases} Dom\ (\text{学籍番号}) = \text{学生}, \\ Dom\ (\text{氏名}) = \text{学生}, \\ Dom\ (\text{住所}) = \text{学生}, \\ Dom\ (\text{所属}) = \text{大学院生}, \\ Dom\ (\text{帰省先}) = \text{下宿生}, \\ Dom\ (\text{科目名}) = \text{科目}, \\ Dom\ (\text{単位数}) = \text{科目}, \\ Dom\ (\text{履修}) = \text{学生}, \\ Dom\ (\text{成績}) = \text{履修}, \end{cases} \quad (8)$$

$$\begin{cases} Ran\ (\text{学籍番号}) = \text{整数}, \\ Ran\ (\text{氏名}) = \text{文字列}, \\ Ran\ (\text{住所}) = \text{文字列}, \\ Ran\ (\text{所属}) = \text{文字列}, \\ Ran\ (\text{帰省先}) = \text{文字列}, \\ Ran\ (\text{科目名}) = \text{文字列}, \\ Ran\ (\text{単位数}) = \text{整数}, \\ Ran\ (\text{履修}) = \text{科目}, \\ Ran\ (\text{成績}) = \text{整数} \end{cases} \quad (9)$$

となる．これから，

d(学籍番号)⊆[学籍番号]＝[学生]×[整数]

などが成り立つ．

DB クラスに I_C, I_V, I_R, Dom, Ran を加えたクラスを次のように定義する．

定義 2（DB_1 の定義） 与えられた I_C, I_V, I_R, Dom, Ran に対し，データベース $d \in DB$ が不変集合の性質(5)，可変集合の性質(6)，関係集合の性質(7)を満たすとき，このような d の集合を $DB_1 (\subseteq DB)$ と定める．DB_1 の要素を，組 $(D, [\cdot], I_C, I_V, I_R, Dom, Ran)$ を仕様とするデータベース，あるいは DB_1 クラスのデータベースという．

次に示す $d_0 \in I \to D$ は空のデータベースであり，DB_1 クラスのデータベースである．ここで $\emptyset$ は空集合を表す．

$$d_0(a)=\begin{cases} [a], & a \in I_C, \\ \emptyset, & a \in I_V \cup I_R. \end{cases} \quad (10)$$

学生データベースは DB_1 クラスのデータベースである．

2.2 可変集合に対する制約

データの集合の間には，何らかの制約が課されることが多い．学生データベースの例では，たとえば「学生は学部生か大学院生のいずれかである」という制約がある．これは，学生集合は学部生集合と大学院生集合の和集合であり，かつ学部生集合と大学院生集合は互いに素である，という制約を表している．最近提案されている仕様記述法はこのように，可変集合間の制約を仕様として記述できるものが多い．

以下では可変集合間の制約を識別子の2項関係で与える．部分集合の制約を表す識別子の関係を $V_{incl} \subseteq I_V \times I_V$，互いに素な集合の制約を表す識別子の関係を $V_{disj} \subseteq I_V \times I_V$，和集合の制約を表す識別子の関係を可変集合の識別子から求める関数を $V_{sum} : I_V \to \mathcal{P}(I_V \times I_V)$，積集合の制約を表す識別子の関係を可変集合の識別子から求める関数を $V_{prod} : I_V \to \mathcal{P}(I_V \times I_V)$ とする．($\mathcal{P}(\cdot)$ はべき集合を表す．) データベース d に対するこれらの制約の意味は，次のように与えられる．(以下，$a, b, c \in I_V$ とする．)

部分集合の制約：

$$(a, b) \in V_{incl} \Rightarrow d(a) \subseteq d(b). \quad (11)$$

互いに素な集合の制約：

$$(a, b) \in V_{disj} \Rightarrow d(a) \cap d(b) = \emptyset. \quad (12)$$

和集合の制約：

$$(b, c) \in V_{sum}(a) \Rightarrow d(a) = d(b) \cup d(c). \quad (13)$$

積集合の制約：

$$(b, c)\in V_{\mathrm{prod}}(a) \Rightarrow d(a)=d(b)\cap d(c). \quad (14)$$

学生データベースの例では，

$$V_{\mathrm{incl}}=\{(\text{下宿生}, \text{学生})\}, \quad (15)$$

$$V_{\mathrm{disj}}=\{(\text{学部生}, \text{大学院生})\}, \quad (16)$$

$$\forall a\in I_{\mathrm{V}}\bullet[V_{\mathrm{prod}}(a)=\emptyset], \quad (17)$$

$$\begin{cases} V_{\mathrm{sum}}(\text{学生}) = \{(\text{学部生}, \text{大学院生})\}, \\ V_{\mathrm{sum}}(\text{学部生})=\emptyset, \quad V_{\mathrm{sum}}(\text{大学院生})=\emptyset, \\ V_{\mathrm{sum}}(\text{下宿生})=\emptyset, \quad V_{\mathrm{sum}}(\text{科目}) = \emptyset \end{cases} \quad (18)$$

となる．これから，

$d(\text{下宿生})\subseteq d(\text{学生})$，

$d(\text{学生})=d(\text{学部生})\cup d(\text{大学院生})$，

$d(\text{学部生})\cap d(\text{大学院生})=\emptyset$

が成り立つ．

DB_1 クラスに V_{incl}, V_{disj}, V_{prod}, V_{sum} を加えたクラスを次のように定義する．

定義 3（DB_2 の定義） 与えられた V_{incl}, V_{disj}, V_{prod}, V_{sum} に対し，データベース $d\in DB_1$ が部分集合の制約(11)，互いに素な集合の制約(12)，和集合の制約(13)，積集合の制約(14)を満たすとき，このような d の集合を $DB_2(\subseteq DB_1)$ と定める．DB_2 の要素を，組 $(D, [\cdot], I_{\mathrm{C}}, I_{\mathrm{V}}, I_{\mathrm{R}}, Dom, Ran, V_{\mathrm{incl}}, V_{\mathrm{disj}}, V_{\mathrm{prod}}, V_{\mathrm{sum}})$ を仕様とするデータベース，あるいは DB_2 クラスのデータベースという．

空のデータベース d_0 は DB_2 の要素である．

学生データベースは DB_2 クラスのデータベースである．

2.3 関係集合に対する制約

実際の仕様記述では，関係に対するさらに細かい性質の記述が要求されることがある．これらの性質を関係集合に対する制約ととらえて整理した結果を以下に示す．本節で示す制約もまた，既存のほとんどの仕様記述法において記述可能である．

まず，データベース d に含まれる関係集合に対し，次の制約が考えられる．（$a\in I_{\mathrm{R}}$ とする．）

定義域および値域の制約：

$$d(a)\subseteq d(Dom(a))\times d(Ran(a)). \quad (19)$$

すなわち，識別子 a が表す関係集合の定義域集合を $d(Dom(a))$ に制限し，値域集合を $d(Ran(a))$ に制限する．

学生データベースの例では，たとえば

$d(\text{履修})\subseteq d(\text{学生})\times d(\text{科目})$

という制約が存在する．

つぎに，関係集合を構成するデータの一意性や必須性にかかわる制約を考えよう．

たとえば，学生データベースの例では，「学生の学籍番号は必ず一意でなければならない」という制約が存在する．これは，学生と学籍番号の関連を表す関係集合 d(学籍番号) が“関数的”（1人の学生に1つの学籍番号が定まる）であり，かつ“全域的”（すべての学生に対して学籍番号が定まる）であり，かつ“単射的”（学生ごとに学籍番号が異なる）であることを表している．また，「すべての科目について履修データが存在する」という制約をもし付け加えたとすると，これは履修を表す関係集合 d(履修) が“全射的”（すべての科目について履修データが存在する）であることを表している．

関係が関数的 (functional) であるとは，関係集合の要素の第1要素がすべて異なることであり，関係が単射的 (injective) であるとは，関係集合の要素の第2要素がすべて異なることである．関係が全域的 (total) であるとは，((19)式により制限された) 定義域集合のすべての要素が，関係集合の第1要素として存在することであり，関係が全射的 (surjective) であるとは，((19)式により制限された) 値域集合のすべての要素が，関係集合の第2要素として存在することである．

関係集合の識別子を与えて，その関係集合が関数的，単射的，全域的，全射的であることを表す識別子の集合を求める関数を $Rc: I_{\mathrm{R}}\to\mathcal{P}(\{Fun, Tot, Inj, Sur\})$ とする．データベース d における関係集合の制約は，次のように与えられる．(以下，$a\in I_{\mathrm{R}}$ とする．)

関数的な関係の制約：

$$Fun\in Rc(a)\Rightarrow \forall x, y\in d(a)\bullet[fst(x)=fst(y)\Rightarrow x=y]. \quad (20)$$

ここに，fst は順序対の第1要素を求める関数である．

単射的な関係の制約：

$$Inj\in Rc(a)\Rightarrow \forall x, y\in d(a)\bullet[snd(x)=snd(y)\Rightarrow x=y]. \quad (21)$$

ここに，snd は順序対の第2要素を求める関数である．

全域的な関係の制約：

$$Tot\in Rc(a)\Rightarrow \{fst(x)\mid x\in d(a)\}=d(Dom(a)). \quad (22)$$

全射的な関係の制約：

$$Sur\in Rc(a)\Rightarrow \{snd(x)\mid x\in d(a)\}=d(Ran(a)). \quad (23)$$

学生データベースの例では，

$$\begin{cases} R_C(\text{氏名}) = \{Fun, Tot\}, \\ R_C(\text{学籍番号}) = \{Fun, Tot, Inj\}, \\ R_C(\text{住所}) = \{Fun\}, \\ R_C(\text{履修}) = \emptyset, \\ R_C(\text{所属}) = \{Fun, Tot\}, \\ R_C(\text{帰省先}) = \{Fun\}, \\ R_C(\text{科目名}) = \{Fun, Tot, Inj\}, \\ R_C(\text{単位数}) = \{Fun, Tot\}, \\ R_C(\text{成績}) = \{Fun\} \end{cases} \tag{24}$$

となる．

DB_2 クラスに R_C を加えたクラスを次のように定義する．

定義 4 (DB_3 の定義) 与えられた R_C に対し，データベース $d \in DB_2$ が，定義域および値域の制約(19)を満たし，かつ関数的な関係の制約(20)，単射的な関係の制約(21)，全域的な関係の制約(22)，全射的な関係の制約(23)を満たすとき，このような d の集合を $DB_3 (\subseteq DB_2)$ と定める．DB_3 の要素を，組 $(D, [\cdot], I_C, I_V, I_R, Dom, Ran, V_{incl}, V_{disj}, V_{prod}, V_{sum}, R_C)$ を仕様とするデータベース，あるいは DB_3 クラスのデータベースであるという．

空のデータベース d_0 は DB_3 の要素である．

学生データベースは DB_3 クラスのデータベースである．

2.4 既存の仕様記述法との関係

本論文における意味データモデルの定義と既存の仕様記述法との対応を具体的に示す．ここでは，ER モデル (entity-relationship model)[9] およびその拡張(たとえば文献 10))，関数データモデル (functional data model) のひとつである DAPLEX[11] およびその拡張である EFDM[12]，そして OMT におけるオブジェクトモデルをとりあげる．

DB_1 を定義する仕様記述の範囲は，ER モデルから関連の制約に関する記述を除いたものに相当する．ER モデルにおける属性集合 (attribute set) が不変集合に，実体集合 (entity set) が可変集合に，関連集合 (relationship set) および属性 (attribute) が関係集合に，それぞれ対応する．EFDM における実体の型 (entity type) が不変集合および可変集合に対応し，単値関数 (single-valued function) および多値関数 (multi-valued function) が関係集合に対応する．OMT におけるデータ型 (data type) が不変集合に，クラス (class) が可変集合に対応し，リンク (link) および属性 (attribute) が関係集合に対応する．メソッド (method) の概念は重要であるが，本論文では扱わない．メソッドを含む仕様の無矛盾性や操作の安全性の証明は，プログラム意味論やデータ型論の分野における未解決の難問である．メソッドを除いた範囲でも有用性は高い．

ER モデルでは関連集合と属性の区別は明確に定義されていないが，値域集合が不変集合である関数的な関係集合が属性に相当すると考えるのが自然であろう．

オブジェクト指向 DBMS や OMT で用いられるクラスは，クラスのインスタンスを一意に識別するためのオブジェクト識別子 (object identifier, OID) と，インスタンスの属性 (メンバ変数) と操作 (メソッド) の組の集合ととらえられる．本論文では，このような複雑な組の形では扱わず，可変集合と関係集合に分解した形で扱う．可変集合は，OID の集合に対応する．関係集合は，OID と属性値の組の集合に対応する．OID の集合と属性の集合を分けて考えることにより，オブジェクト指向 DBMS で問題となる“インスタンスはただ一つのクラスに属し，所属クラスを変更することはできない”という制約[13]は生じない．オブジェクト識別性 (object identity) の概念[14]は，集合の基本的な性質として自然に導入される．

部分集合の制約は，ISA 関連という用語で知られている概念である．拡張された ER モデルはこの制約を扱うことができる．EFDM では引数をもたない関数としてこの制約を記述する．OMT では 2 つの集合をそれぞれスーパクラスとサブクラスの関係として定義することで記述する．このとき，サブクラスはスーパクラスの性質を継承 (inherit) しているという．

互いに素な集合の制約は，拡張された ER モデルでは和集合の概念と組み合わせた汎化 (generalization) の概念に含まれている．EFDM では“制約 (constraints)”の記法を用いて記述する．OMT ではクラスの継承を定義するときに，同じスーパクラスをもつサブクラス間に重複があるかどうか，という形で同時に定義する．

和集合の制約は，拡張された ER モデルでは汎化として記述する．EFDM では記述する手段がない．OMT のオブジェクトモデルでは抽象クラス (abstract class) の概念を用いて記述する．

積集合の制約は，ER モデルや EFDM では記述する手段がない．OMT では抽象クラスと多重継承 (multiple inheritance) の組み合わせによって記述す

る．

関係集合の制約は，歴史的には可変集合間の関係よりも先に認識された概念である．文献 2) であげられている関係集合の制約のうち，要素数 (cardinality) の制約は関数的あるいは単射的の概念に対応し，存在 (existence) の制約は全域的あるいは全射的の概念に対応する．

ER モデルは関係の制約として要素数の制約のみ記述することができる．拡張された ER モデルは存在制約を合わせて記述することができる．EFDM は単値関数と多値関数との使いわけで暗黙のうちに要素数の制約を記述し，存在制約は互いに素な集合を定める記法と同様に制約の記法を用いて陽に指定する．OMT のオブジェクトモデルは拡張された ER モデルを包含する記述能力をもつ．

先に述べた関数的，単射的，全域的，全射的という概念と関数との間には，次のような関連がある．集合 A, B およびその間の 2 項関係 $R \subseteq A \times B$ について，関数 $f: R \to A, g: R \to B$ を考え，$\forall x \in R$ に対して $(f(x)=fst(x)) \wedge (g(x)=snd(x))$ が成り立つとする．このとき，R が関数的であることは，f が単射であることに等しい．R が全域的であることは，f が全射であることに等しい．R が単射的であることは，g が単射であることに等しい．R が全射的であることは，g が全射であることに等しい．ちなみに，f および g が単射であることと要素数が 1 であることが対応し，全射であることと存在制約が満たされることが対応する．

この事実は，たとえばデータの検索を行うときなど，関係を関数として扱いたいときに有効に利用できる．たとえば，f が単射であるとき，R を A から B への部分関数とみなすことができる．すなわち，$\forall a \in A$ に対して $\{x \mid (x \in R) \wedge (f(x)=a)\}$ は空集合かただ 1 つの要素からなる集合となる．f が全単射であるとき，R は全域関数となる．さらに g が単射ならば R は全域関数の単射関数であり，g が全射ならば R は全域関数の全射関数である．

f, g, R および R を関数とみなしたときの制約の相互関係を**表 1** に示す．この表で，'多値' とあるのは，EFDM でいう多値関数にほかならない．すなわち，$\forall a \in A$ に対して $\{x \mid (x \in R) \wedge (f(x)=a)\}$ が一般に 2 個以上の要素を含みうることを表す．'多値' と '単射' が同時に書かれているのは，A の異なる要素に対して，対応する B の部分集合が共通要素をもたないことを表す．'多値' と '全射' が同時に書かれているのは，A のすべての要素に対して対応する B の部分集合の和が，B に一致することを表す．

3. データベースの更新

データベースの更新とは，可変集合あるいは関係集合に対し要素を追加したり削除したりすることである．更新の最小単位は，1 つの集合に対する 1 つの要素の追加および削除であるが，相互に関連のある複数の集合に対する追加および削除を "更新の単位" とするほうが，より実際的である．ここでは，仕様を満たすデータベースが，このような一括更新後も仕様を満たすための条件を示す．

定義 5 (更新の定義) データベース $d \in DB_1$ に対する 1 回の更新で $d(a)(a \in I_V \cup I_R)$ に追加されるデータの集合を $A(a)$，削除されるデータの集合を $S(a)$ とする．ここに，$A(a)$, $S(a)$ は $(A(a) \subseteq [a]-d(a)) \wedge (S(a) \subseteq d(a))$ を満たすとする．A, S はいずれも $I_V \cup I_R$ から D への関数と考えてよい．この (A, S) を "更新指令" と呼ぶことにしよう．$d' \in DB_1$ が $\forall a \in I$ に対して

表 1 関係と関数の対応
Table 1 Correspondence between relation and function.

f の性質	g の性質	R の性質	R を関数とみたときの性質
(なし)	(なし)	(なし)	多値，部分
(なし)	単射	単射的	多値，部分，単射
(なし)	全射	全射的	多値，部分，全射
(なし)	全単射	単射的，全射的	多値，部分，単射，全射
単射	(なし)	関数的	部分
単射	単射	関数的，単射的	部分，単射
単射	全射	関数的，全射的	部分，全射
単射	全単射	関数的，単射的，全射的	部分，単射，全射
全射	(なし)	全域的	多値，全域
全射	単射	全域的，単射的	多値，全域，単射
全射	全射	全域的，全射的	多値，全域，全射
全射	全単射	全域的，単射的，全射的	多値，全域，単射，全射
全単射	(なし)	関数的，全射的	全域
全単射	単射	関数的，全域的，単射的	全域，単射
全単射	全射	関数的，全域的，全射的	全域，全射
全単射	全単射	関数的，全域的，単射的，全射的	全域，単射，全射

$$d'(a)=\begin{cases} d(a), & a\in I_C, \\ d(a)\cup A(a)-S(a), & a\in I_V\cup I_R \end{cases} \quad (25)$$

を満たすとき，d' は d を (A,S) で更新して得られるデータベースであるという．((A,S) によって更新されたデータベースが常に DB_1 の要素であることは明らかである．)

学生データベースの例では，たとえばデータベース d が，氏名が A，学籍番号が 10 番の学生 s のデータを含んでいるとする．このとき d は

$s\in d$ (学生) $\wedge(s,$ "A") $\in d$ (氏名)
$\wedge(s,10)\in d$ (学籍番号)

を満たす．ここから学生 s のデータを削除して，かわりに氏名が B，学籍番号が 15 番の学生 t のデータを追加するときの更新指令 (A,S) は，

A (学生) $=\{t\}$，A (氏名) $=\{(t,$ "B")$\}$，
A (学籍番号) $=\{(t,15)\}$，
S (学生) $=\{s\}$，S (氏名) $=\{(s,$ "A")$\}$，
A (学籍番号) $=\{(s,10)\}$

となる．(学生データを削除するとき氏名および学籍番号も同時に削除しなければならない．) d を (A,S) で更新して得られるデータベース d' は，

$s\notin d'$ (学生) $\wedge(s,$ "A") $\notin d'$ (氏名)
$\wedge(s,10)\notin d'$ (学籍番号)
$\wedge t\in d'$ (学生) $\wedge(t,$ "B") $\in d'$ (氏名)
$\wedge(t,15)\in d'$ (学籍番号)

を満たす．

解くべき問題は，次のようになる．

問題 DB_3 クラスのデータベース d を (A,S) で更新して得られるデータベース d' が DB_3 クラスとなるかを判定すること．

以下，この問題に答えるために，9 つの補題および定理を示す．補題は，可変集合に対する 4 つの制約と，関係集合に対する 5 つの制約のそれぞれについて，更新後のデータベースが制約を満たすための条件を述べたものである．以下，d, $d'\in DB_1$ とする．

補題 1 (**部分集合の制約を満たす更新**) d に対して，$(a,b)\in V_{\mathrm{incl}}$ は部分集合の制約 (11) を満たすとする．(A,S) が次の条件 (26)，(27) を満たすならば，d を (A,S) で更新して得られる d' もまた (11) を満たす．

$$v\in A(a)\Rightarrow(v\in d(b)\wedge v\notin S(b))\vee v\in A(b). \quad (26)$$

$$v\in S(b)\Rightarrow(v\notin d(a)\wedge v\notin A(a))\vee v\in S(a). \quad (27)$$

(証明) 以下，$[i](i\in I)$ に関する $d(i), A(i), S(i)$ の補集合をそれぞれ $\overline{d(i)}, \overline{A(i)}, \overline{S(i)}$ と書くことにする．$d'(a)\subseteq d'(b)$ ならば $d'(a)\cap\overline{d'(b)}=\emptyset$ である．これに，$d'(a)=(d(a)\cup A(a))\cap\overline{S(a)}$, $\overline{d'(b)}=(\overline{d(b)}\cap\overline{A(b)})\cup S(b)$ を代入して整理すると，次式が得られる．

$(d(a)\cap\overline{S(a)}\cap\overline{d(b)}\cap\overline{A(b)}=\emptyset)$
$\wedge(d(a)\cap\overline{S(a)}\cap S(b)=\emptyset)$
$\wedge(A(a)\cap\overline{S(a)}\cap\overline{d(b)}\cap\overline{A(b)}=\emptyset)$
$\wedge(A(a)\cap\overline{S(a)}\cap S(b)=\emptyset)$

さらに $A(a)\cap\overline{S(a)}=A(a)$ および $d(a)\cap\overline{d(b)}=\emptyset$ を代入して整理すると，

$(A(a)\subseteq d'(b))\wedge(d'(a)\cap S(b)=\emptyset)$

が得られる．上式の左項から (26) が，右項から (27) が得られる．(証明終)

以下の補題についても，同様の方法で証明が可能なので，証明を省略する．

補題 2 (**互いに素な集合の制約を満たす更新**) d に対して，$(a,b)\in V_{\mathrm{disj}}$ は互いに素な集合の制約 (12) を満たすとする．(A,S) が次の条件 (28)，(29) を満たすならば，d を (A,S) で更新して得られる d' もまた (12) を満たす．

$$v\in A(a)\Rightarrow(v\notin d(b)\wedge v\notin A(b))\vee v\in S(b). \quad (28)$$

$$v\in A(b)\Rightarrow(v\notin d(a)\wedge v\notin A(a))\vee v\in S(a). \quad (29)$$

補題 3 (**和集合の制約を満たす更新**) d に対して，$a\in I_V$, $(b,c)\in V_{\mathrm{sum}}(a)$ は和集合の制約 (13) を満たすとする．(A,S) が次の条件 (30)～(35) を満たすならば，d を (A,S) で更新して得られる d' もまた (13) を満たす．

$$v\in A(a)\Rightarrow v\in A(b)\vee v\in A(c). \quad (30)$$

$$v\in S(a)\Rightarrow((v\notin d(b)\wedge v\notin A(b))\vee v\in S(b))\wedge((v\notin d(c)\wedge v\notin A(c))\vee v\in S(c)). \quad (31)$$

$$v\in A(b)\Rightarrow(v\in d(a)\wedge v\notin S(a))\vee v\in A(a). \quad (32)$$

$$v\in S(b)\Rightarrow((v\in d(c)\wedge v\notin S(c))\vee v\in A(c))\vee v\in S(a). \quad (33)$$

$$v\in A(c)\Rightarrow(v\in d(a)\wedge v\notin S(a))\vee v\in A(a). \quad (34)$$

$$v\in S(c)\Rightarrow((v\in d(b)\wedge v\notin S(b))\vee v\in A(b))\vee v\in S(a). \quad (35)$$

補題 4 (**積集合の制約を満たす更新**) d に対して，$a\in I_V$, $(b,c)\in V_{\mathrm{prod}}(a)$ は積集合の制約 (14) を満たすとする．(A,S) が次の条件 (36)～(41) を満たすならば，d を (A,S) で更新して得られる d' もまた (14) を満たす．

$$v\in A(a)\Rightarrow((v\in d(b)\wedge v\notin S(b))\vee v\in A(b))\wedge((v\in d(c)\wedge v\notin S(c)\vee v\in A(c)). \quad (36)$$

$$v\in S(a)\Rightarrow v\in S(b)\vee v\in S(c). \quad (37)$$

$$v\in A(b)\Rightarrow((v\notin d(c)\wedge v\notin A(c))\vee v\in S(c))\vee v\in A(a). \quad (38)$$

$$v\in S(b)\Rightarrow(v\notin d(a)\wedge v\notin A(a))\vee v\in S(a). \quad (39)$$

$$v\in A(c)\Rightarrow((v\notin d(b)\wedge v\notin A(b))\vee v\in S(b))\vee v\in A(a). \quad (40)$$

$$v\in S(c)\Rightarrow(v\notin d(a)\wedge v\notin A(a))\vee v\in S(a). \quad (41)$$

補題 5（定義域および値域の制約を保存する更新） d に対して，$a\in I_R$ は定義域および値域の制約(19)を満たすとする．(A,S) が次の条件(42)～(44)を満たすならば，d を (A,S) で更新して得られる d' もまた(19)を満たす．ただし，$b=Dom(a), c=Ran(a)$ とおいた．

$$v\in A(a)\Rightarrow((fst(v)\in d(b)\wedge fst(v)\notin S(b))\vee fst(v)\in A(b))\wedge((snd(v)\in d(c)\wedge snd(v)\notin S(c))\vee snd(v)\in A(c)). \quad (42)$$

$$v\in S(b)\Rightarrow \forall v'\in[a]\bullet[(v'\in d(a)\wedge v'\notin S(a))\vee v'\in A(a)\Rightarrow fst(v')\neq v]. \quad (43)$$

$$v\in S(c)\Rightarrow \forall v'\in[a]\bullet[(v'\in d(a)\wedge v'\notin S(a))\vee v'\in A(a)\Rightarrow snd(v')\neq v]. \quad (44)$$

補題 6（関数的な関係の制約を保存する更新） d およびそれを (A,S) で更新して得られる d' に対して，$a\in I_R$ は定義域および値域の制約(19)を満たすとする．$Fun\in R_C(a)$ であり，d に対して a が関数的な関係の制約(20)を満たすとき，(A,S) が次の条件(45)を満たすならば，d' もまた(20)を満たす．

$$v\in A(a)\Rightarrow \forall v'\in[a]\bullet[(v\neq v')\wedge(((v'\in d(a)\wedge v'\notin S(a))\vee v'\in A(a))\Rightarrow fst(v')\neq fst(v))]. \quad (45)$$

補題 7（単射的な関係の制約を満たす更新） d およびそれを (A,S) で更新して得られる d' に対して，$a\in I_R$ は定義域および値域の制約(19)を満たすとする．$Inj\in R_C(a)$ であり，d に対して a が単射的な関係の制約(21)を満たすとき，(A,S) が次の条件(46)を満たすならば，d' もまた(21)を満たす．

$$v\in A(a)\Rightarrow \forall v'\in[a]\bullet[(v\neq v')\wedge(((v'\in d(a)\wedge v'\notin S(a))\vee v'\in A(a))\Rightarrow snd(v')\neq snd(v))]. \quad (46)$$

補題 8（全域的な関係の制約を保存する更新） d およびそれを (A,S) で更新して得られる d' に対して，$a\in I_R$ は定義域および値域の制約(19)を満たすとする．$Tot\in R_C(a)$ であり，d に対して a が全域的な関係の制約(22)を満たすとき，(A,S) が次の条件(47)，(48)を満たすならば，d' もまた(22)を満たす．ただし，$b=Dom(a)$ とおいた．

$$v\in S(a)\Rightarrow \exists v'\in[a]\bullet[(v'\neq v)\wedge(fst(v')=fst(v))\wedge((v'\in d(a)\wedge v'\notin S(a))\vee v'\in A(a))]\vee fst(v)\in S(b). \quad (47)$$

$$v\in A(b)\Rightarrow \exists v'\in[a]\bullet[fst(v')=v\wedge v'\in A(a)]. \quad (48)$$

補題 9（全射的な関係の制約を保存する更新） d およびそれを (A,S) で更新して得られる d' に対して，$a\in I_R$ は定義域および値域の制約(19)を満たすとする．$Sur\in R_C(a)$ であり，d に対して a が全射的な関係の制約(23)を満たすとき，(A,S) が次の条件(49)，(50)を満たすならば，d' もまた(23)を満たす．ただし $b=Ran(a)$ とおいた．

$$v\in S(a)\Rightarrow \exists v'\in[a]\bullet[(v'\neq v)\wedge(snd(v')=snd(v))\wedge((v'\in d(a)\wedge v'\notin S(a))\vee v'\in A(a))]\vee snd(v)\in S(b). \quad (49)$$

$$v\in A(b)\Rightarrow \exists v'\in[a]\bullet[snd(v')=v\wedge v'\in A(a)]. \quad (50)$$

定理（正しい更新の判定） DB_3 クラスのデータベース d に対して更新指令 (A,S) が補題1から補題

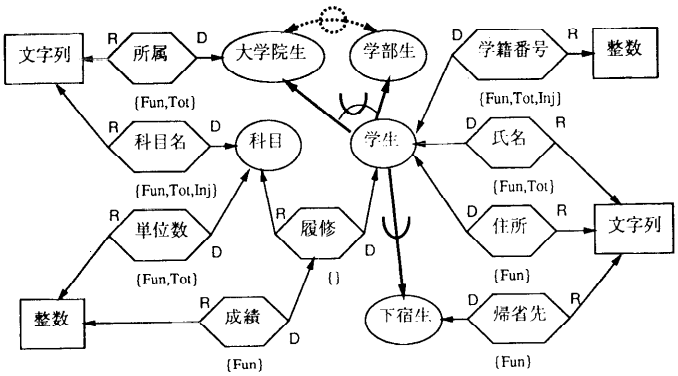


図 2 学生データベースの仕様図
Fig. 2 Specification diagram of a database of students.

9までに述べられている条件をすべて満たすならば，d を (A, S) で更新して得られるデータベース d' は DB_3 クラスである．

(証明) 上記から明らか．

この定理により，(A, S) がデータベース d を正しく更新するかどうかを更新前に検査することが可能であることがわかる．

系（データベースが仕様を満たすことの判定） データベース d が与えられたとき，空のデータベース d_0 を (d, d_0) で更新して得られるデータベース d' が DB_3 クラスならば，d もまた DB_3 クラスである．

(証明) (25)式より，$\forall a \in I$ に対して

$$d'(a)=\begin{cases} d_0(a)=d(a), & a \in I_C, \\ d_0(a) \cup d(a) - d_0(a) = d(a), & a \in I_V \cup I_R \end{cases}$$

であるから，$d'=d$．(証明終)

この系から，データベースが仕様を満たすかの検査が可能であることもわかる．

4. プログラムによる確認

検査方法の正しさを確認するために，検査プログラムを実際に作成し，学生データベースに適用してみた．

2章の式(1)〜(4)，(8)，(9)，(15)〜(18)，(24)で与えられる学生データベースの仕様をグラフで表したものを**図2**に示す．

グラフの節は，正方形が不変集合を，円形が可変集合を，菱形が関係集合を，それぞれ表す．関係集合を始点とする有向辺は，Dom および Ran の要素を表す．可変集合を結ぶ有向辺は，V_{incl} の要素を表すものと，V_{sum} の要素を表すものがある．自宅生と下宿生を結ぶ両方向の矢をもつ辺は，2本の有向辺を略記したものであり，V_{disj} の要素を表す．

```
0:可変 ∈ d(学生)
0:可変 ∈ d(下宿生)
0:可変 ∈ d(学部生).
2:関係:(0:可変,1:不変:文字列:<A>) ∈ d(氏名)
4:関係:(0:可変,3:不変:整数:<10>) ∈ d(学籍番号)
6:関係:(0:可変,5:不変:文字列:<P県Q市>) ∈ d(住所)
8:関係:(0:可変,7:不変:文字列:<R県S市>) ∈ d(帰省先)
9:可変 ∈ d(学生)
9:可変 ∈ d(大学院生)
11:関係:(9:可変,10:不変:文字列:<B>) ∈ d(氏名)
13:関係:(9:可変,12:不変:整数:<15>) ∈ d(学籍番号)
15:関係:(9:可変,14:不変:文字列:<P県T市>) ∈ d(住所)
17:関係:(9:可変,16:不変:文字列:<C研究室>) ∈ d(所属)
18:可変 ∈ d(科目)
20:関係:(18:可変,19:不変:文字列:<プログラム言語論>) ∈ d(科目名)
22:関係:(18:可変,21:不変:整数:<4>) ∈ d(単位数)
23:可変 ∈ d(科目)
25:関係:(23:可変,24:不変:文字列:<ソフトウェア工学特論>) ∈ d(科目名)
27:関係:(23:可変,26:不変:整数:<2>) ∈ d(単位数)
28:関係:(0:可変,18:可変) ∈ d(履修)
29:関係:(9:可変,23:可変) ∈ d(履修)
31:関係:(29:関係:(9:可変,23:可変),30:不変:整数:<80>) ∈ d(成績)
```

図3 学生データベースのデータ例
Fig. 3 An example of data stored in the database of students.

いま，学生データベースに2人分のデータが入っているものとする（**図3**）．氏名A，学籍番号10の学生は学部生かつ下宿生であり，住所はP県Q市，帰省先はR県S市である．氏名B，学籍番号15の学生は大学院生であり，住所はP県T市，所属はC研究室である．学生Aはプログラム言語論を，学生Bはソフトウエア工学特論を履修している．学生Bのソフトウエア工学特論の成績は80点である．

4.1 新しい学生データを付け加える

新しい学生データを追加しようとした例を**図4**に示す．学生集合に新しい要素を追加しようとしたとこ

```
34:可変 ∈ A(学生)
--
全域的な関係の制約に違反！: 34:可変 ∈ A(学生).
        ***** 検査リスト *****
        (34:可変, 任意の要素) が A(氏名) に 含まれない.
        **********
全域的な関係の制約に違反！: 34:可変 ∈ A(学生).
        ***** 検査リスト *****
        (34:可変, 任意の要素) が A(学籍番号) に 含まれない.
        **********
和集合の制約に違反！: 34:可変 ∈ A(学生).
        ***** 検査リスト *****
        34:可変 が A(学部生) に 含まれない.
        34:可変 が A(大学院生) に 含まれない.
        **********
```

図4 新しい学生データを追加しようとして失敗した例—学生集合に要素を追加
Fig. 4 Invalid transaction (1)—addition of a new element to the set of students.

```
34:可変 ∈ A(下宿生)
--
部分集合の制約に違反！: 34:可変 ∈ A(下宿生).
        ***** 検査リスト *****
        34:可変 が d(学生) に 含まれない.
        34:可変 が A(学生) に 含まれない.
        **********
```

図5 新しい学生データを追加しようとして失敗した例—下宿生集合に要素を追加
Fig. 5 Invalid transaction (2)—addition of a new element to the set of rooming students.

```
0:可変 ∈ A(大学院生)
0:可変 ∈ S(学部生)
--
全域的な関係の制約に違反！: 0:可変 ∈ A(大学院生).
        ***** 検査リスト *****
        (0:可変, 任意の要素) が A(所属) に 含まれない.
        **********
```

図6 学生の身分を変更しようとして失敗した例—学部生から大学院生に変更
Fig. 6 Invalid transaction (3)—subtraction of an element from the set of undergraduate students and addition of the same element to the set of graduate students.

ろ，エラーが3つ報告された．

- 氏名が登録されていないので全域的な関数の制約に違反している．
- 学籍番号が登録されていないのでやはり全域的な関数の制約に違反している．
- 学部生か大学院生のどちらにも登録されていないので和集合の制約に違反している．

このように，学生データと同時に登録すべきデータが正しく指摘されていることがわかる．

下宿生集合に要素を追加しようとした場合は**図5**のようになり，エラーが1つ報告される．

- 学生データとして登録されていないので部分集合の制約に違反している．

4.2 学生の身分を変更する

もともと学部生だった学生を大学院生に変更しようとした例を**図6**に示す．学部生集合から削除した要素を大学院生集合に追加しようとしたところ，エラーが1つ報告された．

- 所属が登録されていないので全域的な関係の制約に違反している．

学部生集合の要素を誤って大学院生としても登録しようとした例を**図7**に示す．図6の結果からエラーが1つ増える．

- 学部生のデータとして削除されていないので互いに素な集合の制約に違反している．

下宿生を下宿生でないように変更しようとした例を**図8**に示す．下宿生集合から要素を削除したところ，エラーが1つ報告された．

- 帰省先が削除されていないので定義域および値域の制約に違反している．

4.3 学籍番号を入れ換える

2人の学生の学籍番号が逆に入力されていたとして，これを修正することを考える．正しく修正するには，いったん2人の学籍番号のデータを削除し，新しい学籍番号を再登録すればよい．

誤って片方の学生の学籍番号の追加を忘れてしまった例を**図9**に示す．エラーが1つ報告される．

- 新しい学籍番号が登録されず，学生データが削除されたわけでもないので，全域的な関係の制約に違反している．

この場合は誤りの解決法として2つ方法があり，学籍番号を追加するか，学生データそのものを削除するか，どちらかを選択する必要があるが，プログラムはその両方について正しく発見していることがわかる．

```
0:可変 ∈ A(大学院生)
--
全域的な関係の制約に違反！: 0:可変 ∈ A(大学院生).
        ***** 検査リスト *****
        (0:可変, 任意の要素) が A(所属) に 含まれない.
        **********
互いに素な集合の制約に違反！: 0:可変 ∈ A(大学院生).
        ***** 検査リスト *****
        0:可変 が d(学部生) に 含まれる.
        0:可変 が S(学部生) に 含まれない.
        **********
```

図7 学生の身分を変更しようとして失敗した例—学部生を大学院生としても登録

Fig. 7 Invalid transaction (4)—addition of an element of the set of undergraduate students to the set of graduate students.

```
0:可変 ∈ S(下宿生)
--
定義域および値域の制約に違反！: 0:可変 ⊂ S(下宿生).
        ***** 検査リスト *****
        8:関係:(0:可変, 7:不変:文字列:<R県S市>) が d(帰省先) に 含まれる.
        8:関係:(0:可変, 7:不変:文字列:<R県S市>) が S(帰省先) に 含まれない.
        **********
```

図8 学生の身分を変更しようとして失敗した例—下宿生を下宿生でないように変更

Fig. 8 Invalid transaction (5)—subtraction of an element from the set of rooming students.

```
32:関係:(0:可変, 12:不変:整数:<15>) ∈ A(学籍番号)
4:関係:(0:可変, 3:不変:整数:<10>) ∈ S(学籍番号)
13:関係:(9:可変, 12:不変:整数:<15>) ∈ S(学籍番号)
--
全域的な関係の制約に違反！: 13:関係:(9:可変, 12:不変:整数:<15>) ∈ S(学籍番号).
        ***** 検査リスト *****
        (9:可変, 同一でない任意の要素) が d(学籍番号) に 含まれない.
        (9:可変, 同一でない任意の要素) が A(学籍番号) に 含まれない.
        9:可変 が S(学生) に 含まれない.
        **********
```

図9 学籍番号を入れ換えようとして失敗した例—片方の学生の学籍番号の追加を忘れた

Fig. 9 Invalid transaction (6)—exchange of student ID of two students: lack of an adding command of a new element to the relation set of student IDs.

```
32:関係:(0:可変, 12:不変:整数:<15>) ∈ A(学籍番号)
33:関係:(9:可変, 3:不変:整数:<10>) ∈ A(学籍番号)
13:関係:(9:可変, 12:不変:整数:<15>) ∈ S(学籍番号)
--
関数的な関係の制約に違反！: 32:関係:(0:可変, 12:不変:整数:<15>) ∈ A(学籍番号).
        ***** 検査リスト *****
        4:関係:(0:可変, 3:不変:整数:<10>) が d(学籍番号) に 含まれる.
        4:関係:(0:可変, 3:不変:整数:<10>) が S(学籍番号) に 含まれない.
        **********
単射的な関係の制約に違反！: 33:関係:(9:可変, 3:不変:整数:<10>) ∈ A(学籍番号).
        ***** 検査リスト *****
        4:関係:(0:可変, 3:不変:整数:<10>) が d(学籍番号) に 含まれる.
        4:関係:(0:可変, 3:不変:整数:<10>) が S(学籍番号) に 含まれない.
        **********
```

図10 学籍番号を入れ換えようとして失敗した例—片方の学生の学籍番号の削除を忘れた

Fig. 10 Invalid transaction (7)—exchange of student ID of two students: lack of a subtracting command of an old element from the relation set of student IDs.

逆に片方の学生の学籍番号の削除を忘れてしまった例を**図10**に示す．エラーが2つ報告される．

- 重複した学生番号のデータが削除されていないので，関数的な関係の制約および単射的な関係の制約に違反している．

このように，検査プログラムが返す検査結果は，正しい指令を作成するための情報として有用であることがわかる．

5. おわりに

データベースの仕様が意味データモデルで与えられたとき，正しい更新を保証するための更新指令の検査方法を示した．これにもとづく検査プログラムを作成して有用性を確かめた．本論文の成果は多くの仕様記述法に対して適用が可能である．とくに現在盛んに研究が進められているオブジェクト指向 DBMS の分野において，一貫性制約の問題に対する理論的裏付けの一部（メソッドの扱いを除く）を与えることになる．

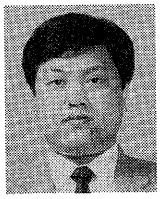
本論文の方法は DBMS と独立に利用することができ，データ管理機能の貧弱な DBMS を安全に利用するための方法として有効である．また利用者インタフェイスに本論文の方法を組み込むことにより，更新操作の正しさをインタフェイス側で検査することができるので，DBMS に負担をかけない迅速な対話処理が可能となる．このことを実際の処理系で評価すべく，処理系の試作をすすめている．仕様から更新用および検索用のインタフェイスの設計を導出する方法や，検査プログラムがインタフェイスに与える効果の評価について，機会を改めて報告したい．

多項関係やデータの視点（view）の概念を扱えるように意味データモデルの定義を拡張すること，データをなるべく壊さない安全な仕様変更[13]の技術を開発すること，は今後の課題である．

参考文献

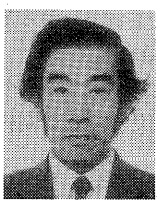
1) Ullman, J. D.: *Principle of Database Systems*, Second Edition, Computer Science Press, Rockville, Md. (1982).
2) Hull, R. and King, R.: Semantic Database Modeling: Survey, Applications, and Research Issues, *ACM Computing Surveys*, Vol. 19, No. 3, pp. 201-260 (1987).
3) 有澤 博：意味データモデルの最近の動向，情報処理，Vol. 32, No. 9, pp. 1023-1031 (1991).
4) Deux, O. et al.: The Story of O_2, *IEEE Transaction on Knowledge and Data Engineering*, Vol. 2, No. 1, pp. 91-108 (1990).
5) Ramburgh, J., Blaha, M. et al.: *Object Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, N. J. (1991).
6) 鈴木卓治，小林光夫：データ間の関係からデータベースおよびその利用者インタフェースを自動生成する方法について，第45回情報処理学会全国大会論文集，4S-5 (1992).
7) 鈴木卓治，小林光夫：データベースの利用者インタフェースの自動作成—問い合わせ機能の組込み，第46回情報処理学会全国大会論文集，8G-5 (1993).
8) Abiteboul, S. and Hull, R.: IFO: A Formal Semantic Database Model, *ACM Transaction on Database Systems*, Vol. 12, No. 4, pp. 525-565 (1987).
9) Chen, P. P.: The Entity-Relationship Model—Toward a Unified View of Data, *ACM Transaction on Database Systems*, Vol. 1, No. 1, pp. 9-36 (1976).
10) Teorey, T. J., Yang, D. and Fry, J. P.: A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model, *ACM Computing Surveys*, Vol. 18, No. 2, pp. 197-222 (1986).
11) Shipman, D. W.: The Functional Data Model and the Data Language DAPLEX, *ACM Transaction on Database Systems*, Vol. 6, No. 1, pp. 140-173 (1981).
12) Gray, P. M., Kulkarni, K. G. and Paton, N. W.: *Object-Oriented Databases: A Semantic Data Model Approach*, Prentice-Hall, New York (1992).
13) 増永良文，田中克己：次世代データベースシステムの展望，情報処理，Vol. 32, No. 5, pp. 602-613 (1991).
14) 田中克己：オブジェクト指向データベースの基礎概念，情報処理，Vol. 32, No. 5, pp. 500-513 (1991).

(平成5年4月21日受付)
(平成6年4月21日採録)



鈴木　卓治（正会員）

1965 年生．1988 年電気通信大学情報数理工学科卒業．1990 年同大学院情報工学専攻博士前期課程修了．1944 年同専攻博士後期課程単位取得退学．同年 4 月より国立歴史民俗博物館情報資料研究部助手．ソフトウェア学と数理工学を基礎として，情報システム開発支援環境の構築に取り組んでいる．日本ソフトウェア科学会，日本応用数理学会各会員．



小林　光夫（正会員）

1941 年生．1966 年東京大学大学院工学系研究科修士課程修了．東京大学工学部助手，埼玉大学理学部数学科講師を経て，現在電気通信大学情報工学科助教授．色彩画像の情報数理工学的研究に取り組んでいる．日本応用数理学会，日本色彩学会各会員．