〔公　開〕

ＴＲ－Ｃ－００９１

# オブジェクト指向データベースのアクセス制御機構とセキュリティ設計支援手法

荒　木　禎　史
Tadashi　ARAKI

１９９３　９．２４

ＡＴＲ通信システム研究所

# オブジェクト指向データベースのアクセス制御機構とセキュリティ設計支援手法

荒木禎史

1993年9月24日

目次

# 1. はじめに

現在、社会の情報化は不断に進行しており、種々様々な情報がネットワークを介して伝達され、データベースに格納され、利用・処理されている。そのような状況下で、“セキュリティ”の重要性が強く認識されるようになってきている。セキュリティ問題には、不当な情報の漏洩・改竄、処理の妨害、行為の否認等が挙げられるが、これらはいずれも健全な情報化社会の進展を妨げるものであり、その解決手段を確立することは極めて重要な課題である。これまでに、情報の暗号化の研究を始め、ユーザー認証、アクセス制御、推論制御等の各分野で研究がなされ、報告されている。最近では、ネットワーク化の進行に伴って、分散・ネットワーク環境におけるセキュリティの研究も盛んになりつつある。

さて、データベースのセキュリティに関する研究としては、これまでは主として、関係データベースについて行われてきた。ところが、CAD/CAMデータベースやマルチメディア環境、ネットワーク管理への応用を意識して、次第にオブジェクト指向データベース（Object-Oriented Database System (OODB)）の必要性・重要性が言われるようになってきた（文献[1]-[3]）。これに対応してOODBのセキュリティに関する研究がいくつかなされ始めたが（文献[4]-[14]）、まだ十分な解決法を提示するものは現われていない。

本研究では、まず、OODBの“アクセス制御機構”を提案し、次に、その“セキュリティ設計支援手法”を提案する（文献[15], [16]）。

アクセス制御の目的は、OODB内部のデータの不当な情報漏洩・改竄、また、不正なメソッド実行を防止することである。そこでは基本的に“強制アクセス制御（階層アクセス制御）”のポリシー、即ち、「各エンティティ（ユーザー、データ、…）にセキュリティレベルを設定し、下位レベル→上位レベル、又は、同レベル間、の情報フローのみ許可する。」を採用する。アクセス制御機構の提案においては、レベル設定の矛盾を防止するための“レベル設定制約”、及び、レベルの上下とアクセス許可・不許可の関係に関する“アクセス制御規則”を明らかにする。

セキュリティ設計支援とは、ユーザーやデータベース管理者のセキュリティ上の要求を満足するようにデータベース設計者がOODB設計を行う際に、これを機械的に支援することである。その際、上記のアクセス制御機構に基づいた設計を行うものとする。ここでは具体的には、ユーザー・管理者の“データアクセス要求”と“機密漏洩防止要求”を入力とし、この両者の矛盾を検出し、次に矛盾を解消するための要求の変更やメソッドの再定義を行い、最後に矛盾の解消された要求を満足するように各エンティティにセキュリティレベルを設定する。このような設計過程を、データベース設計者と設計支援システムの対話的な処理で実現するための基本的手法を提案する。

以下、2. 節で本研究の概要を述べ、3. 節でOODBの特徴とセキュリティ上の問題点を詳細に考察し、4. 節でアクセス制御機構、及び、5. 節でセキュリティ設計支援手法の詳細をそれぞれ説明し、6. 節で提案手法の簡単な評価と今後の展開について述べる。

## 2. 本研究の概要

ここでは、まず、セキュリティ設計支援の意義・内容を述べる。次に、本研究で提案するOODBのアクセス制御機構とセキュリティ設計支援手法の概要を説明する。詳細・厳密な議論は3. 節以下で行なう。

### 2.1. セキュリティ設計支援とは

情報処理システムにおけるセキュリティ上の脅威には、不当な情報の漏洩・改竄、処理の妨害、等が挙げられる。これらの脅威に対抗するためのセキュリティ機能としては、脅威の防止・検出、損害からの回復、セキュリティ機能自体の訂正等がある（文献[17]）。情報処理システムのセキュリティ機能設計とは、具体的なセキュリティ上の要求（「Aさんにはファイルxを見せたくない。」、「BさんはデータYを書き換えてはならない。」等）を満足するように、これらの機能を実現することであると考えられる。その際、システムの入出力機構や内部構造による制約条件や、ユーザーのシステムに対する処理上の要求を考慮する必要がある。

しかし、複雑な処理や多種多量のデータを扱うシステムに対してこのような設計を人手のみで行うには多大な時間と労力を要し、かつ、見落としや誤りの発生する危険がある。よって、セキュリティ設計を容易かつ正確に行うために、これを機械的に支援するセキュリティ設計支援（図1）の開発が望まれる。

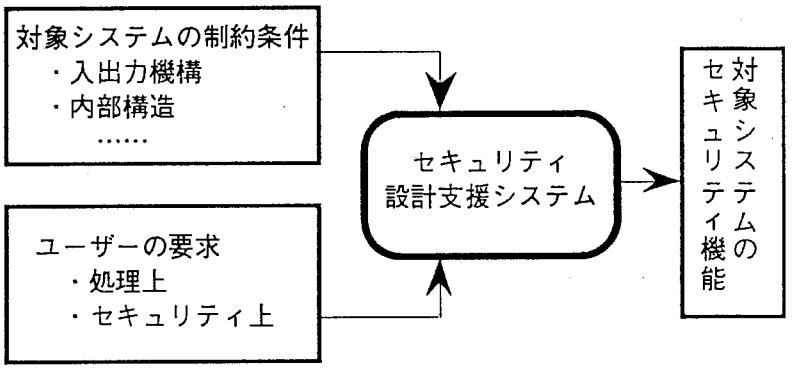



図1　セキュリティ設計支援

### 2.2. OODBのアクセス制御機構の概要

ここでは、図2の研究グループ人事ファイルの例を用いて説明する。但し、クラスはデータ型を定義するオブジェクト、インスタンスはその具体的な実現値のオブジェクトである。

さて、「ユーザーU1に対して“研究テーマクラス”に関する情報を秘密にしたい。」というセキュリティ上の要求があるとしよう。これを強制アクセス制御（階層アクセス制

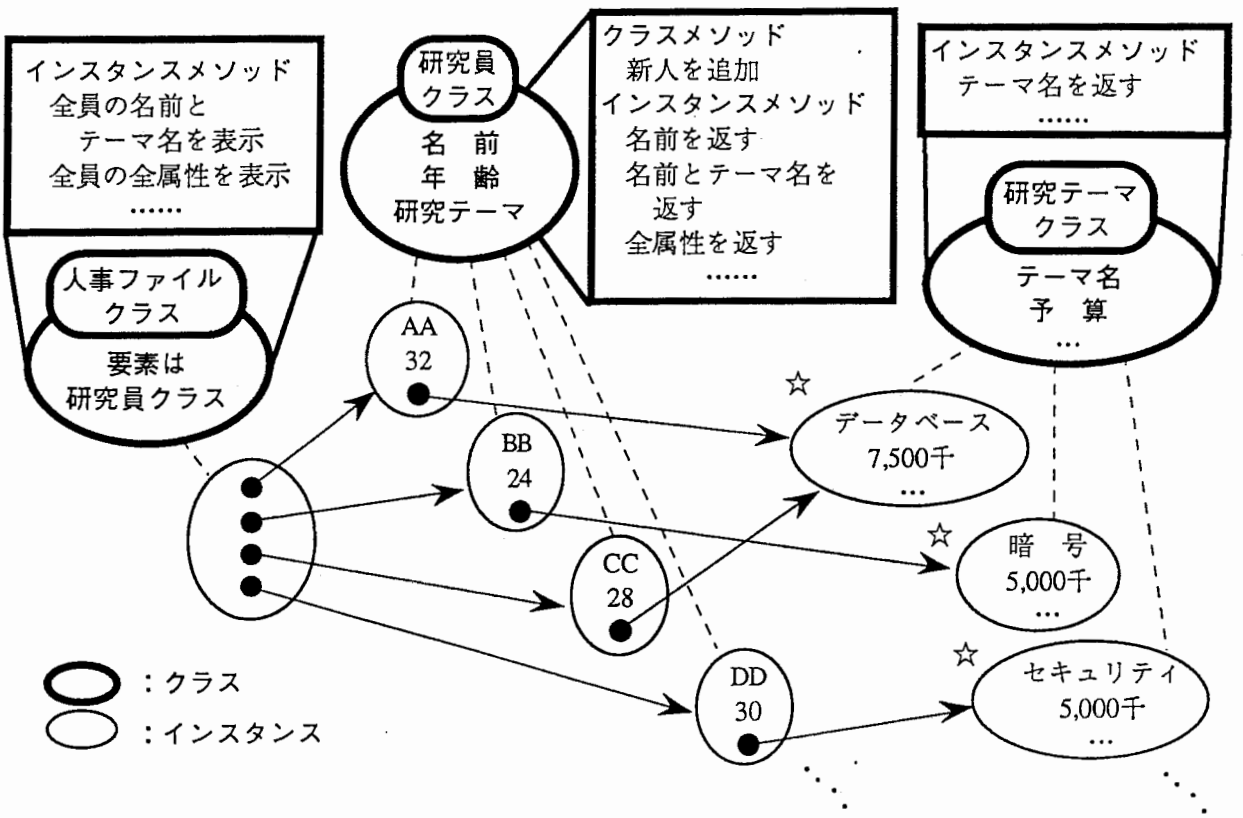



図2　OODBの例（研究グループ人事ファイル）

御）で実現するには次のようにする。まず、「データ（オブジェクト）のセキュリティレベルと等しいかそれ以上のレベルのユーザーのみがそのデータの情報を読み出せる」という一般的なアクセス制御規則を定め、これを満足するようにアクセス制御機構を設計する。そこで、“研究テーマクラス”のレベルをU1のレベルより高く設定する。これにより、U1は“研究テーマクラス”の情報を直接読み出すことはできなくなる。

しかし、これではまだ不十分である。もし、“研究テーマクラス”のインスタンス（図中☆印）のレベルがU1のレベルよりも低ければ、U1はその内部情報を読み出すことができる。インスタンスはクラスの具体的な実現値であるから、その情報により、結果的に“研究テーマクラス”の情報（例えば、“テーマ名”や“予算”という変数（属性）が存在する、というような情報）を類推できてしまう。このような不合理をなくすためには、「インスタンスのレベルはその型を定義しているクラスのレベル以上でなければならない。」というレベル設定に関する制約を予め定めておく必要がある。この制約に従ってレベル設定すれば、あるクラスの情報を読み出せないユーザーは、常にそのインスタンスの情報も読み出せなくなる。

アクセス制御機構の提案においては、以上述べたような“アクセス制御規則”や“レベル設定制約”を一般的に明らかにすることが主要な目的である。

## 2.3. OODBのセキュリティ設計支援の概要

前述の「ユーザーU1に対して“研究テーマクラス”に関する情報を秘密にしたい。」という機密漏洩防止要求（機密要求）に加え、「U1は全研究員の“名前”と“研究テーマ名”を知りたい。」というデータアクセス要求（アクセス要求）があるとする。“研究テーマ名”は“研究テーマクラス”に関する情報なので、これらの2つの要求は矛盾する。機密要求を優先する場合は、U1に“人事ファイルクラス”上で定義された次のメソッド、“全員の名前とテーマ名を表示”と“全員の全属性を表示”、の実行権を与えてはならない。そのためには、上記2つのメソッドのレベルをU1のレベルより高く設定すればよい。しかし、このままではU1は研究員の名前も知ることができなくなる。そこで、“人事ファイルクラス”上で、“全員の名前（のみ）を表示する”というメソッドを新たに定義し、それが“研究員クラス”上の“名前を返す”というメソッドを内部的に実行するようにし、この新メソッドのレベルをU1のレベル以下に設定する。これらにより、上記のアクセス要求の一部と機密要求が実現できる。なお、レベル設定においては、2.2. 節で述べた“レベル設定制約”も同時に満足せねばならない。

セキュリティ設計支援とは、データベース設計者がこのような、アクセス・機密要求の入力、要求間の矛盾判定、アクセス要求変更や適切なメソッド再定義による矛盾解消、そして、矛盾の解消した要求を実現するためのセキュリティレベル設定、という一連の処理を行う際、これを機械的に支援することである。本研究では、これを設計者と支援システムとの対話的な処理で実現するための手法を提案する。なお、2.2. 節の“アクセス制御規則”を実現するような基本的なアクセス制御機構は既に出来ていると仮定する。ここでは、具体的なアクセス要求や機密要求を実現するための設計支援手法を提案するのが目的である。

## 3. OODBの特徴とセキュリティ上の問題点

### 3.1. OODBの特徴

OODBに関しては、まだ、広くコンセンサスを得られるような厳密な定義は存在しない。しかし、OODBの備えるべき基本的な特徴についてはある程度の共通の認識が出来ている（文献[2]）。例えば、Atkinson他はOODBに必要な特徴を文献[3]で論じている。それによると、OODBには次の8つの特徴が必要である。それは、①複合オブジェクト、②オブジェクトの同一性、③カプセル化、④型とクラス、⑤クラスまたは型階層、⑥ポリモルフィズム、⑦計算完備性、⑧拡張性、である。これらを次のように整理し直す。

[A] データとメソッド（操作）の一体化、及び、データアクセスの機構

これは、上記の③, ⑥に対応する。即ち、個々のデータに対し、そのデータに対してなされるメソッド（操作）がそれぞれ定義されており、これらが一体となってデータベース中に格納されている（③）。オブジェクト内部へのデータアクセスは、全てそこで定義されているメソッドを起動することによってなされる。アクセスされたオブジェクトが、更に二次的に他のオブジェクトのデータにアクセスする場合もある。また、異なるデータ型（クラス）に対して定義された同種のメソッドに共通のメソッド名をつけることが出来る（⑥）。例えば、“表示”というメソッド名を、“ウィンドウを表示”、“ビットマップを表示”、“グラフを表示”等のメソッドに共通して使うことが出来る。

[B] クラス—インスタンス関係

これは、上記の④に対応する。クラスは、データ構造の“型”やメソッドの機能を定義しているオブジェクトであり、インスタンスはそれを具体化した個々のデータそのもののオブジェクトである。クラス全体で共通に定義される変数をクラス変数、インスタンス毎に異なる値を持つ変数をインスタンス変数という。

[C] クラスの汎化階層

これは、上記の⑤に対応する。あるクラスで定義された型やメソッドに新しい条件や制約を加えるなどしてより具体化・特殊化したクラスを定義した場合、そのクラスを元のクラスのサブクラス（下位クラス）という。サブクラスに対して、元のクラスをスーパークラス（上位クラス）という。例えば、クラス“日本人”は、スーパークラス“人”のサブクラスである。サブクラスはスーパークラスで定義された諸性質を基本的に継承（インヘリタンス）する。従って、スーパークラスで定義した変数やメソッドは、サブクラスで改めて定義しなくてもそのまま有効である。このようなクラス間の階層関係を“is-a関係”という。

[D] 複合オブジェクトとオブジェクトの同一性

これは、上記の①, ②に対応する。オブジェクトの構成要素がまた他のオブジェクトであるような入れ子構造をなすオブジェクトを、複合オブジェクトという（①）。このような入れ子構造を“part-of関係”という。本研究では、複合オブジェクトとして、変数とその値の組からなるような“組オブジェクト”と、オブジェクトの集合を一つのオブジェクトと見なす“集合オブジェクト”の2つを考える[8]（リストオブジェクト、や配列オブジェクトも集合オブジェクトとして扱う）。また、各オブジェクトは固有のオブジェクト識別子で識別される（②）。変数値や集合の要素として複合オブジェクトを構成しているオブジェクトは、その識別子により参照される（この場合、識別子は一種のポインタと考えられる）。

あと、⑦計算完備性、⑧拡張性、が残っているが、“計算完備性”は本研究にはあまり関係ない。また、“拡張性”はユーザーが新クラスを定義出来るということであるが、ここでは新クラスの定義は考慮しない。

以上の特徴は、大きく、データアクセスに関する特徴（[A]）と、構造に関する特徴（[B], [C], [D]）に分けられると考えられる。

### 3.2. 具体例

図2と同じ例を図3に再掲して説明する。ここでは、“人事ファイル”、“研究員”、“研究テーマ”という3つのクラスが定義されている。人事ファイルクラスは、研究員クラスのオブジェクトを要素として持つ集合オブジェクトのクラスである。また、“研究員クラス”のインスタンス変数“研究テーマ”は、研究テーマクラスのオブジェクトを値として持つ。なお、この図には“is-a関係”の例は記されていない。

ユーザーが、“人事ファイルクラス”のインスタンス（図中★印）上で「全員の名前とテーマ名を表示」というメソッド（メソッド1）を起動すると、それは★内の各ポインタによって参照されている“研究員クラス”の各インスタンス（☆印）上の「名前とテーマ名を返す」というメソッド（メソッド2）を二次的に起動する。メソッド2は更に、“研究テーマクラス”の「テーマ名を返す」というメソッドを二次的に起動する。最終的に、メソッド1は返ってきた値を基に、図4に示すような表を画面に表示する。

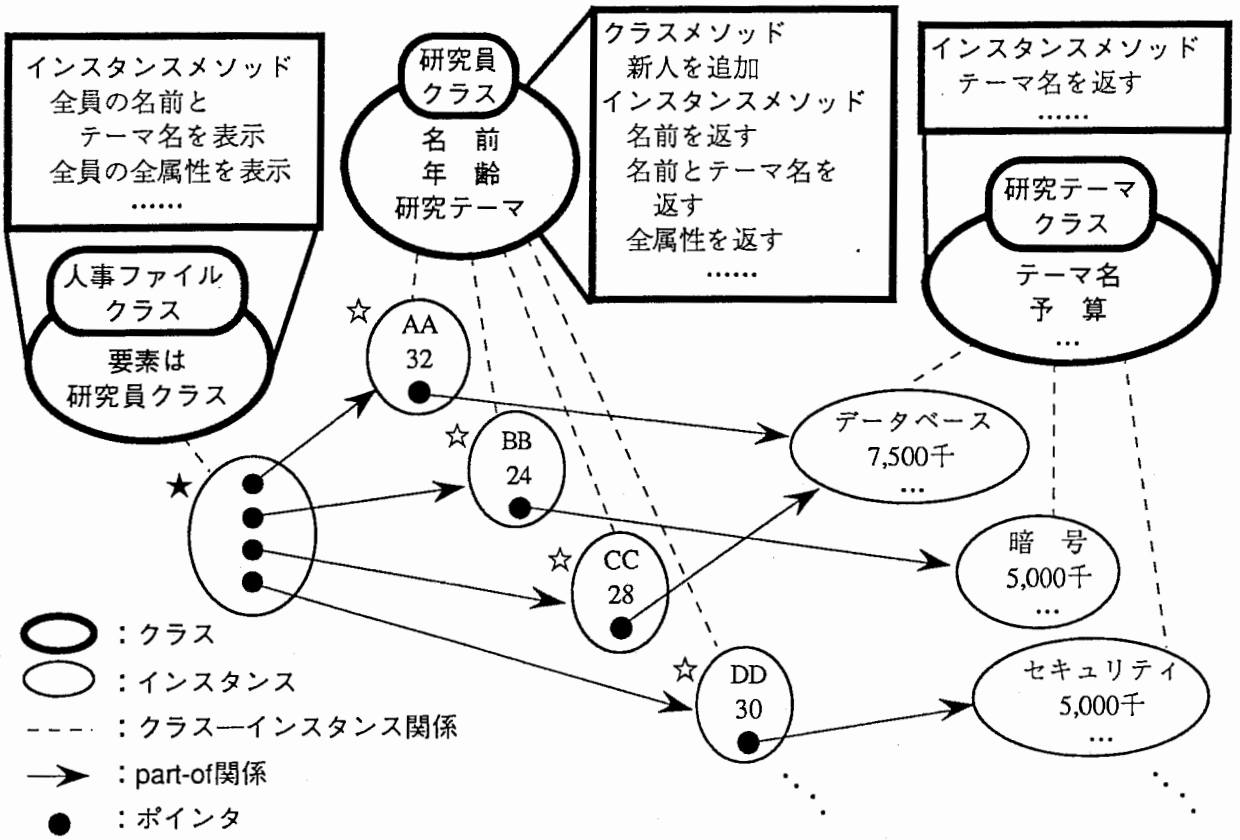



図3　OODBの例（研究グループ人事ファイル）

| 名前 | 年齢 | テーマ名 | 予算 | …… |
|---|---|---|---|---|
| AA | 32 | データベース | 7,500千 | …… |
| BB | 24 | 暗号 | 5,000千 | …… |
| CC | 28 | データベース | 7,500千 | …… |
| DD | 30 | セキュリティ | 5,000千 | …… |
| …… | | | | |

図4　メソッド実行結果の例

### 3.3. セキュリティ上の問題点

3.1. 節で説明したOODBの特徴[A]-[D]に対応して、その問題点を考察する。

[A] データへのアクセスは全てメソッドを通してなされるので、メソッドの存在によりクラスや変数の存在が漏洩する場合がある。メソッドが他のオブジェクトのメソッドを二次的に起動することによって、間接的な情報漏洩が起こる可能性がある。

例えば、図3の例において、あるユーザーに“研究員クラス”の全インスタンスの“名前”情報を秘密にしたい、というセキュリティ上の要求があったとする。そのためには、“名前”という変数の存在自体を秘密にするのが望ましい。よって、単に個々のインスタンスにおける“名前”へのアクセスを禁止するだけでは不十分である。なぜなら、「名前

を表示」というメソッドが見えれば、“名前”という変数の存在を知ることが出来てしまうからである。

[B] クラス（変数）とインスタンス（変数値）の機密度に差をつける必要がある。

例えば、AAの研究テーマは知られても良いが、BBとCCの研究テーマは秘密にしたい、というような要求の場合、“研究員クラス”の“研究テーマ”という変数の存在は知られても構わないが、個々のインスタンス毎に“研究テーマ”へのアクセス許可を判定する必要がある。

[C] サブクラスはスーパークラスの特別な場合なので、サブクラスの情報を知っているユーザーはスーパークラスの情報を類推できる。

[D] オブジェクト単独での機密度と、それが他の複合オブジェクトの変数値や要素であるときの機密度に差をつける必要がある。

例えば、暗号という研究テーマがあることは知られても良いが、それがBBの担当であることは知られたくない、等。

# 4. アクセス制御機構

## 4.1. 提案する制御機構の基本的アクセス制御ポリシーと従来手法の問題点

アクセス制御には、強制アクセス制御（Mandatory Access Control (MAC)、セキュリティレベルによる制御）と任意アクセス制御（Discretionary Access Control (DAC)、アクセス制御リストによる制御）という2つの主要な方法がある。従来のOODBのアクセス制御の研究についてみると、文献[4]-[10]はMACを、[4], [11]-[14]はDACを、それぞれ用いている。本研究ではMACを採用する。MACに基づく最も典型的な抽象的セキュリティモデルは“Bell and LaPadulaモデル（BLPモデル）（文献[18]）”である。BLPモデルの基本的な制御ポリシーは、「下位レベル→上位レベル、又は、同レベル間、の情報フローのみ許可する。」である。なお、代表的なセキュリティモデルについては付録1を参照されたい。

文献[4]-[10]で提案された手法の問題点は次のとおりである。これらは、オブジェクト単独での機密度と、それが他の複合オブジェクトの変数値や要素であるときの機密度を区別していない。よって、3.3. 節の問題点[D]を解決できない。また、文献[8]を除いて、クラスとインスタンスの機密度を区別する機構が明確でない。[8]においても、変数と変数値の機密度の差については考慮していない。よって、3.3. 節の問題点[B]を解決できない。文献[6], [7]においては、メソッドへのレベル設定とその実行制御について提案しているが、メソッドの存在によりクラスや変数の存在が漏洩する場合やメソッドが他のオブジェクトのメソッドを二次的に起動することによって間接的な情報漏洩が起こる場合を考慮していない。よって、3.3. 節の問題点[A]を解決できない。

以上より、従来手法では3.3. 節の問題点を完全には解決できない。本研究では、これらの問題点を解決する新しいアクセス制御機構を以下の節で提案する。そこでは、OODBの構成要素間の“レベル設定制約”と、“アクセス制御規則”を検討する。

## 4.2. 仮定

ここでは以下の仮定をおく。

- レベルの順序関係は半順序関係とする
- レベルの変更は考慮しない
- 新クラスの定義・クラスの変更は考慮しない
- クラスの多重継承（複数のクラスをスーパークラスとして持つ）は考慮しない

## 4.3. レベル設定に関する考察

レベル設定の制約を述べる前に、その基本となる考え方について触れる。

3.3. 節の問題点[A]を解決するには、クラスや変数だけでなく、メソッドにもレベルを設定して、そのレベル以上のユーザーしかこれらの存在を知られないようにするのが有効

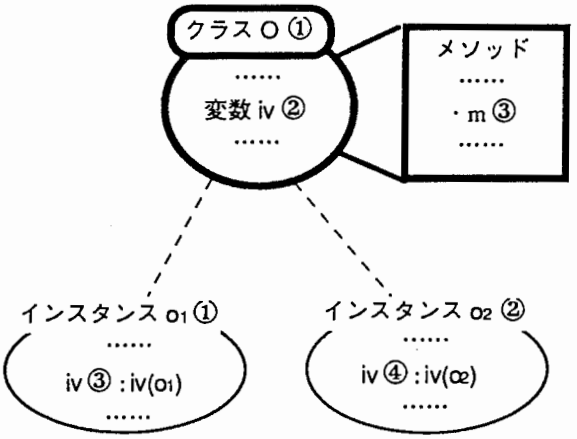



図5　レベル設定の例

である。特にメソッドのレベル設定については、間接的な情報漏洩を防ぐために、それがアクセスする変数や二次的に起動する他オブジェクトのメソッドのレベル以上にする必要がある。しかし、もし、メソッドから変数或は他メソッドへの情報フローがある場合は(メソッドが変数へ書き込むような場合)、そのメソッドのレベルと変数或は他メソッドのレベルを等しくする。このようなメソッドを“書き込み系メソッド”という。

しかし一方、これも2.3. 節で述べたように、クラスや変数の存在は知られてもよいが、具体的なインスタンスや変数値の機密度がインスタンス毎に異なる場合もある（問題点[B]）。そこで、インスタンスのレベルやインスタンス毎の変数のレベルも必要となる。これらについては、「クラスや変数の存在は知っている」ことが前提となるので、インスタンスのレベルはクラス以上、インスタンスにおける変数のレベルはクラスにおけるレベル以上である必要がある（図5）。

また、たとえメソッドを起動できても、それがアクセスするインスタンスのレベルや変数のインスタンスにおけるレベルが高ければ、結果的に実行不可能となる。メソッドのレベルは、それがアクセスする変数のクラスにおけるレベルよりは常に高いが、インスタンスにおけるレベルよりも高いとは限らない（図5）。

以上の議論は、主として組オブジェクトの場合を述べたが、集合オブジェクトにおいても基本的な考え方は同様である。

問題点[C]については、クラスのレベルはそのスーパークラスのレベル以上、そのクラス内のスーパークラスから継承された変数やメソッドのそのクラスにおけるレベルは、それらのスーパークラスにおけるレベル以上とすることで解決する。

問題点[D]については、オブジェクト単独でのレベルと、それが他の複合オブジェクトの変数値や要素であるときのレベルを区別し、後者を前者より高くする。後者は、ポインタへのレベル設定と考えることが出来る（3.1. 節[D]参照）。

更に、変数、メソッド、（集合オブジェクトの）要素のレベルは、それを定義している

（含んでいる）オブジェクトのレベル以上とする。次節以降で、以上の制約を記号を用いて厳密に記述する。

### 4.4. 記号の定義

ここでは、レベル設定制約を記述するための記号についてまとめておく。なお、記号には、必要に応じて適宜上下にインデックスを付けることがある。

O：クラス
o：インスタンス
class(o)：インスタンスoの含まれるクラス
sup(O)：クラスOのスーパークラス

cv：組オブジェクトのクラス変数
cv(O)：クラスOにおけるクラス変数cvの値
class(cv)：cvの値となるオブジェクトの含まれるクラス（=class(cv(O))）
inh(O, cv)：クラスOにおいて定義されているクラス変数cvがsup(O)から継承されたものなら“1”、クラスOで新たに定義されたものなら“0”を返す関数

iv：組オブジェクトのインスタンス変数
iv(o)：インスタンスoにおけるインスタンス変数ivの値
class(iv)：ivの値となるオブジェクトの含まれるクラス（=class(iv(o))）
inh(O, iv)：クラスOにおいて定義されているインスタンス変数ivがsup(O)から継承されたものなら“1”、クラスOで新たに定義されたものなら“0”を返す関数

EO：集合オブジェクトの要素クラス（集合オブジェクトは、クラスにおいて、その要素となるオブジェクトの含まれるクラス（要素クラス）を定義する、とする。例えば、図3の人事ファイルクラスは研究員クラスを要素クラスとする集合オブジェクトのクラスである。また、集合オブジェクトにはis-a関係は存在しないと考える）

m：メソッド
Vac(O, m)：組オブジェクトのクラスOのメソッドmによってアクセスされるO内の変数の集合
Eac(O, m)：集合オブジェクトのクラスOのメソッドmによってアクセスされるO内の要素クラスの集合

Mac(O, m)：クラスOのメソッドmによって二次的にアクセスされるクラスとメソッドの直積の集合

inh(O, m)：クラスOにおいて定義されているメソッドmがsup(O)から継承されたものなら“1”、クラスOで新たに定義されたものなら“0”を返す関数

write((O, m), (O, cv))：クラスOにおいて定義されているメソッドmからOで定義されているクラス変数cvへ情報フローがあれば“1”、そうでなければ“0”を返す関数（cvの代わりにiv, EO, m'（他メソッド）であっても同様）

level(O)：クラスOのレベル

level(o)：インスタンスoのレベル

level(O, cv)：組オブジェクトのクラスOでのクラス変数cvのレベル

level(O, iv)：組オブジェクトのクラスOでのインスタンス変数ivのレベル

level(o, iv)：組オブジェクトのインスタンスoでのインスタンス変数ivのレベル

level(O', O)：集合オブジェクトのクラスO'内部での要素クラスOのレベル

level(o', o)：集合オブジェクトのインスタンスo'内部での要素oのレベル

level(O, m)：クラスOでのメソッドmのレベル

## 4.5. レベル設定制約

### 4.5.1. 基本制約

クラス—インスタンス間、及び、クラス—スーパークラス間のレベル設定に関する基本制約は次のとおりである。

$$level(o) \geqq level(O), \text{ if } O = class(o) \quad (1)$$

$$level(O) \geqq level(O'), \text{ if } O' = sup(O) \quad (2)$$

### 4.5.2. 組オブジェクトに関する制約

まず、組オブジェクトを次のように表現する。

$$O = \langle cv_1 : cv_1(O), cv_2 : cv_2(O), \cdots, cv_m : cv_m(O); iv_1, iv_2, \cdots, iv_n; m_1, m_2, \cdots, m_k \rangle$$

$$o = \langle iv_1 : iv_1(o), iv_2 : iv_2(o), \cdots, iv_n : iv_n(o) \rangle$$

provided　$O = class(o)$

メソッド$m_i$ $(1 \leqq i \leqq k)$のアクセスする変数や他オブジェクトは次のとおりとする。

$$Vac(O, m_i) = \{cv^i_1, cv^i_2, \cdots, cv^i_{p(i)}; iv^i_1, iv^i_2, \cdots, iv^i_{q(i)}\}$$

$$Mac(O, m_i) = \{(O^i_1, m^i_{11}), (O^i_1, m^i_{12}), \cdots, (O^i_1, m^i_{1h(i,1)}); (O^i_2, m^i_{21}), (O^i_2, m^i_{22}), \cdots,$$

$$(O^i_2, m^i_{2h(i,2)}); \cdots; (O^i_{j(i)}, m^i_{j(i)1}), (O^i_{j(i)}, m^i_{j(i)2}), \cdots, (O^i_{j(i)}, m^i_{j(i)h(i,j(i))})\}$$

provided $\{class(cv^i_1), class(cv^i_1), \cdots, class(cv^i_{p(i)}), class(iv^i_1), class(iv^i_2), \cdots,$

$class(iv^i_{p(i)})\} \subset \{O^i_1, O^i_2, \cdots, O^i_{j(i)}\}$

クラス変数に関する制約は次のとおりである。

For all i ($1 \leqq i \leqq m$)

$level(O, cv_i) \geqq level(O)$ (3)

$level(O, cv_i) \geqq level(cv_i(O))$ (4)

$level(O, cv_i) \geqq level(sup(O), cv_i)$, if $inh(O, cv_i) = 1$ (5)

インスタンス変数に関する制約は次のとおりである。

For all i ($1 \leqq i \leqq n$)

$level(O, iv_i) \geqq level(O)$ (6)

$level(O, iv_i) \geqq level(class(iv_i))$ (7)

$level(O, iv_i) \geqq level(sup(O), iv_i)$, if $inh(O, iv_i) = 1$ (8)

$level(o, iv_i) \geqq level(o)$ (9)

$level(o, iv_i) \geqq level(iv_i(o))$ (10)

$level(o, iv_i) \geqq level(O, iv_i)$ (11)

メソッドに関する制約は次のとおりである。

For all i, s, t, u and v ($1 \leqq i \leqq k$, $1 \leqq s \leqq p(i)$, $1 \leqq t \leqq q(i)$, $1 \leqq u \leqq j(i)$, $1 \leqq v \leqq h(i, j(i))$)

$level(O, m_i) \geqq level(O)$ (12)

$level(O, m_i) \geqq level(O, cv^i_s)$ (13)

$level(O, m_i) = level(O, cv^i_s)$, if $write((O, m_i), (O, cv^i_s)) = 1$ (14)

$level(O, m_i) \geqq level(O, iv^i_t)$ (15)

$level(O, m_i) = level(O, iv^i_t)$, if $write((O, m_i), (O, iv^i_t)) = 1$ (16)

$level(O, m_i) \geqq level(O^i_t, m^i_{uv})$ (17)

$level(O, m_i) = level(O^i_t, m^i_{uv})$, if $write((O, m_i), (O^i_t, m^i_{uv})) = 1$ (18)

$level(O, m_i) \geqq level(sup(O), m_i)$, if $inh(O, m_i) = 1$. (19)

### 4.5.3. 集合オブジェクトに関する制約

まず、集合オブジェクトを次のように表現する。

$O = \{EO_1; EO_2; \cdots; EO_m; m_1, m_2, \cdots, m_k\}$

$o=\{o_{11}, o_{12}, \cdots, o_{1n(1)}; o_{21}, o_{22}, \cdots, o_{2n(2)}; \cdots; o_{m1}, o_{m2}, \cdots, o_{mn(1)}\}$

provided $O=class(o)$ and

$EO_i=class(o_{i1}), class(o_{i2}), \cdots, class(o_{in(i)})$ for all i $(1 \leqq i \leqq m)$

メソッド$m_i$ $(1 \leqq i \leqq k)$のアクセスする要素クラスや他オブジェクトは次のとおりとする。

$Eac(O, m_i)=\{EO^i_1, EO^i_2, \cdots, EO^i_{p(i)}\}$

$Mac(O, m_i)=\{(O^i_1, m^i_{11}), (O^i_1, m^i_{12}), \cdots, (O^i_1, m^i_{1h(i,1)}); (O^i_2, m^i_{21}), (O^i_2, m^i_{22}), \cdots,$

$(O^i_2, m^i_{2h(i,2)}); \cdots; (O^i_{j(i)}, m^i_{j(i)1}), (O^i_{j(i)}, m^i_{j(i)2}), \cdots, (O^i_{j(i)}, m^i_{j(i)h(i,j(i))})\}$

provided $\{EO^i_1, EO^i_2, \cdots, EO^i_{p(i)}\} \subset \{O^i_1, O^i_2, \cdots, O^i_{j(i)}\}$

要素クラス、及び、要素に関する制約は次のとおりである。

For all i $(1 \leqq i \leqq m)$

$level(O, EO_i) \geqq level(O)$ (20)

$level(O, EO_i) \geqq level(EO_i))$ (21)

For all i, j $(1 \leqq i \leqq m, 1 \leqq j \leqq n(i))$

$level(o, o_{ij}) \geqq level(o)$ (22)

$level(o, o_{ij}) \geqq level(o_{ij})$ (23)

$level(o, o_{ij}) \geqq level(O, EO_i)$ (24)

メソッドに関する制約は次のとおりである。

For all i, s, u and v $(1 \leqq i \leqq k, 1 \leqq s \leqq p(i), 1 \leqq u \leqq j(i), 1 \leqq v \leqq h(i, j(i)))$

$level(O, m_i) \geqq level(O)$ (25)

$level(O, m_i) \geqq level(O, EO^i_s)$ (26)

$level(O, m_i)=level(O, EO^i_s)$, if $write((O, m_i), (O, EO^i_s))=1$ (27)

$level(O, m_i) \geqq level(O^i_u, m^i_{uv})$ (28)

$level(O, m_i)=level(O^i_u, m^i_{uv})$, if $write((O, m_i), (O^i_u, m^i_{uv}))=1$ (29)

### 4.6. アクセス制御規則

アクセス制御機構は機能は、ユーザーのレベルとオブジェクト（クラス、インスタンス、変数、集合オブジェクトの要素）やメソッドのレベルを比較して、表示や実行の許可判定を下すことである。基本的には、ユーザーのレベル以下のオブジェクトやメソッドは表示可であるとする。

データを読み出すメソッドに関しては、制御機構は、そのメソッドがアクセスするオブ

ジェクトのレベルがユーザーのレベル以下であれば実行許可、そうでなければ実行不許可と判定する。データを変更するメソッドに関しては、制御機構は、そのメソッドがアクセスするオブジェクトのレベルがユーザーのレベルに等しければ実行許可、そうでなければ実行不許可と判定する。

データの追加メソッドは、“インスタンスの追加メソッド”、と、“集合オブジェクトにおける要素の追加メソッド”、の2つが考えられる。まず、インスタンスの追加メソッドは全て、そのオブジェクトの属するクラスのクラスメソッドとして定義されている（追加メソッドはそのクラス内部の全インスタンス変数にアクセスすると考える）。この追加メソッドのレベルより上位のユーザーはこれを常に実行できる。制御機構は、追加したインスタンス及びインスタンス内部でのインスタンス変数のレベルをユーザーのレベルに設定して、Diskに書き込む。また、集合オブジェクトにおける要素の追加メソッドは全て、その集合オブジェクトのインスタンスメソッドとして定義されている。その追加メソッドのレベルより上位のユーザーはこれを常に実行できる。制御機構は、追加した要素の集合オブジェクト内部でのレベルをユーザーのレベルに設定して、Diskに書き込む。

この追加（append）に関しては、通常のBLPモデルでの規則とやや異なっているが、その理由は、自分がその存在を知らないようなオブジェクトに対してappendすることは意味がないと考えられるからである。それでも、追加したオブジェクトのレベルは全て追加したユーザーのレベルに等しくなるので、「上位→下位に情報が流れない」というポリシーは満たされている。

また、起動されたメソッドが、更に他のオブジェクトにアクセスする場合、例えば図6のように2通りの場合がある（この例では$iv(o_{11})=o_{21}$となっている）。1つは直接アクセスする場合（ア）、もう1つはそのメソッドが定義されたオブジェクトの変数値または要素としてアクセスする場合である（イ）。メソッドmを起動したユーザーuのレベルをlevel(u)とすると（$level(u) \geqq level(O_1, m)$）、mが$o_{21}$に直接アクセスする場合は、アクセス制

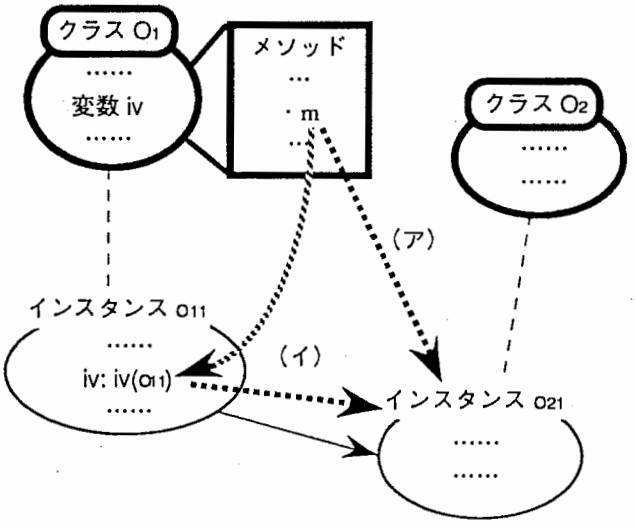



図6　メソッドのアクセス

御機構はlevel(u)≧level($o_{21}$)のときのみ実行を許可する。$o_{11}$の変数値としてアクセスする場合は、level(u)≧level($o_{11}$, iv)ときのみ実行を許可する。実行許可の場合は、iv($o_{11}$)＝$o_{21}$と（10）式より常にlevel(u)≧level($o_{21}$)となるので$o_{21}$へのアクセスは常に許可される（$o_{21}$へのアクセス許可判定の必要はない）。いずれも、$o_{21}$へのアクセスが許可されれば、mは$O_2$上で定義されたメソッドのどれかを二次的に起動するが、level(u)≧level($O_1$, m)と（19）式によりこれも常に許可される（起動許可判定の必要はない）。

# 5. セキュリティ設計支援

## 5.1. セキュリティ設計支援の目的

基本的なアクセス制御機構を設計したら、次に、具体的なセキュリティ上の要求を満足するようにユーザーやデータにアクセス権やセキュリティレベルを設定せねばならない。しかし、多種多量のデータを格納したり、メソッド間の内部実行関係が複雑なOODBの場合は、これを人手のみで行うには多大な時間や労力を必要とし、誤り発生の危険もある。よって、これを容易かつ正確に行うための機械的な“セキュリティ設計支援”が望まれる。しかし、3. 節で述べたOODBの特徴を反映した設計支援手法に関する研究はいまだなされていない。

本節では、OODBに対して主として“データへのアクセス要求（アクセス要求）”と“機密漏洩防止要求（機密要求）”が与えられたときに、これをアクセス制御で実現する場合のセキュリティ設計支援について述べる（メソッドに関する機密要求は考慮しない）。そこでは、4. 節で述べた基本的なアクセス制御機構、即ち、ユーザーと内部データのレベルを比較してアクセス許可・不許可を判定する機能は既に実現されていると仮定する。

本節で提案する手法は、この基本的機能の上で具体的なアクセス・機密要求を満足する（両者が矛盾する場合は機密要求を優先し、それに反しない範囲でアクセス要求も部分的に実現する）ようにユーザーやOODB内のデータやメソッドにセキュリティレベルを設定し、また、適切なメソッドの再定義を支援する手法である。その際、4.5. 節の“レベル設定制約”を満足するようにレベル設定を行う必要がある。

以下の節において、このようなセキュリティ設計支援を、OODB設計者と支援システムの対話的な環境で実現するための手法を提案する。

## 5.2. セキュリティ設計支援手法

### 5.2.1. 概要

提案する設計支援手法は、OODBに対するアクセス・機密要求を実現するためのメソッド再定義やセキュリティレベル設定を、OODB設計者と支援システムの対話形式で実現するものであり、その概要は次のとおりである。

まず、設計者がアクセス・機密要求を入力し、それをシステムがグラフ化する。その際、4.5. 節のレベル設定制約を一緒にグラフ化する。次に、グラフ探索による要求矛盾判定と矛盾解消・メソッド再定義を設計者とシステムとの対話処理で行い、矛盾解消後にシステムはセキュリティレベルの設定を行う（図7）。以下、これらの各ステップを詳述する。なお、ユーザーと、クラス・メソッド等のOODB構成要素をまとめて“エンティティ”と呼ぶことにする。

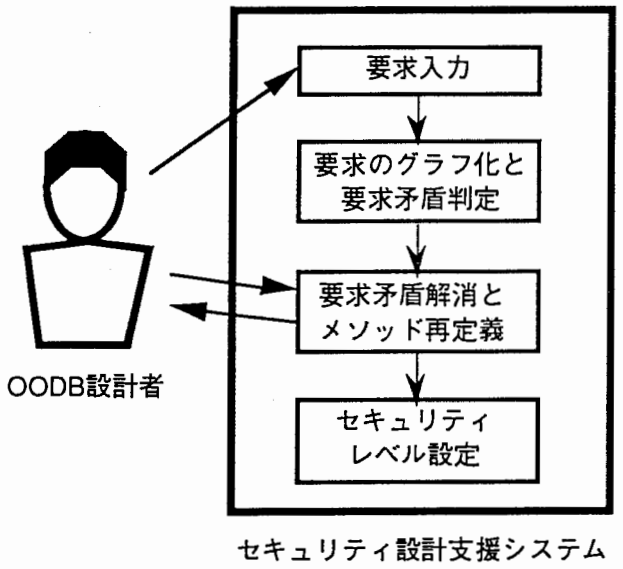



図7　設計支援手法の概要

### 5.2.2. 要求入力

要求入力は、ユーザーのアクセス・機密要求を設計支援システムが処理可能な形で入力するプロセスであり、データベース設計者がユーザーの要求を聞いて行うものとする。

OODBでは、データへのアクセスはメソッドの実行によってなされることから、アクセス要求は全てメソッドの実行要求に変換する。例えば、図3の例で「ユーザーU1は全研究員の名前を知りたい。」というアクセス要求は、「U1は“人事ファイルクラス”の“全員の名前とテーマ名を表示”を実行したい。」に変換する。

アクセス要求・機密要求ともに、システムが処理可能なように行列形式で入力する。“アクセス要求行列（arm）”は、

arm(m, U)＝1　（「Uがmを実行したい。」とき）
　　＝0　（それ以外）、

“機密要求行列（srm）”は、

srm(E, U)＝1　（「EをUに知られたくない（但しEはメソッドではない）。」とき）
　　＝0　（それ以外）、

と、それぞれ入力する。

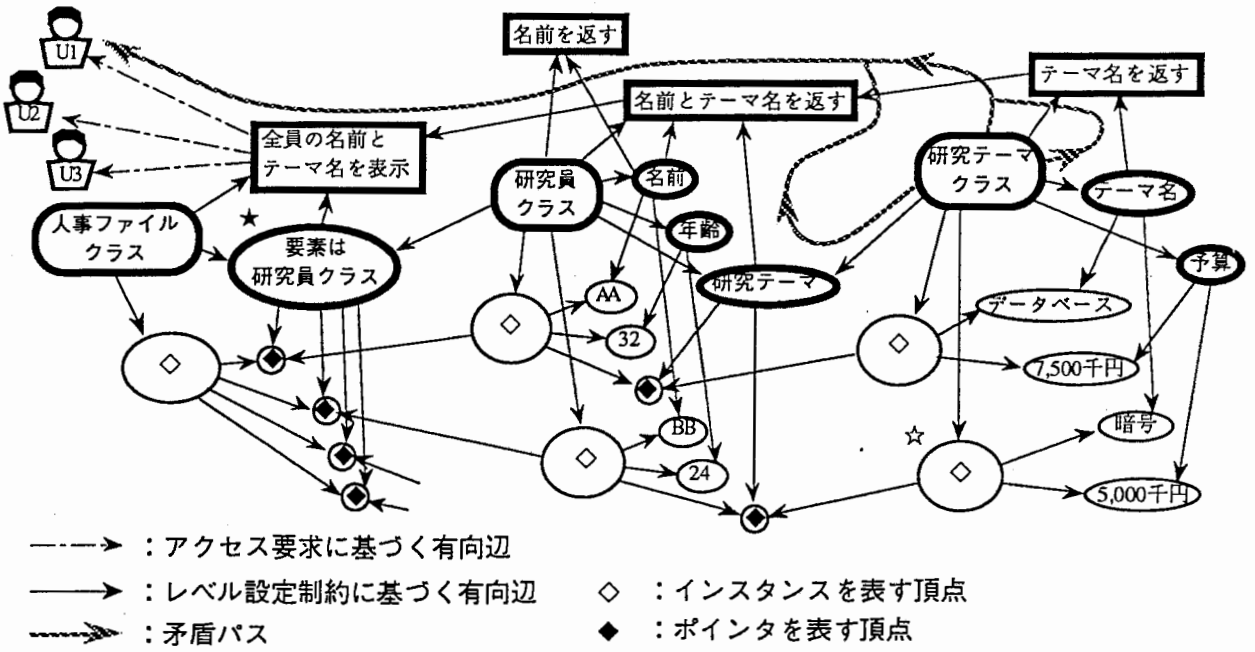



図8　要求のグラフ化と要求矛盾判定

### 5.2.3. 要求のグラフ化と要求矛盾判定

この処理は、要求をグラフ化して機密要求に反する情報フローを算出するプロセスで、全て支援システム内部で行われる。

まず、システムはアクセス要求行列とレベル設定制約を基に、“処理要求行列（prm)”を作成する。処理要求行列は、行と列にOODB内の全エンティティが対応する正方行列であり、次のように作成する。

prm(E1, E2)＝1　(E1＝E2)
　　＝1　(arm(E1, E2)＝1、又は、レベル設定制約でE1≦E2)
　　＝0　(それ以外)

次に、システムは各エンティティをグラフの頂点に各々対応させ、prm(E1, E2)＝1に対応してE1→E2の向きに有向辺(E1, E2)を引く（注1）。このグラフを“処理要求グラフ”といい、その有向辺はアクセス要求とデータベース内部で起こり得る情報フローの向きを表している。図8は、図3のOODBに基づく処理要求グラフの部分である。この例では、3人のユーザー（U1, U2, U3）がおり、皆「“人事ファイルクラス”の“全員の名前とテーマ名を表示”を実行したい。」というアクセス要求を持っている。

次に、要求矛盾判定を行う。そのため、システムは処理要求行列をブール演算でn－1乗して（行列のサイズをnとする）、データ→ユーザー間の全ての直接及び間接的な情報フローを算出する。この行列を“到達行列（rbm）”といい、rbm(E1, E2)＝1のとき、E1→E2への情報フロー（有向パス）が存在する（詳しくは付録2を参照されたい）。そこで、

srm(E, U)＝1、かつ、rbm(E, U)＝1なる(E, U)成分が存在するとき、“要求が矛盾している”と判定する（文献[19], [20]）。このとき、EからUへのグラフ上のパスを“矛盾パス”といい、その終端辺(m, U)を機密要求(E, U)に“矛盾するアクセス要求”という。

図8では、「U1には“研究テーマ”に関する情報を知られたくない。」という機密要求がある場合を考える。その矛盾パスは図中に示すとおりである。

（注1）mが書き込み系メソッドで、m→Eの情報フローがある場合は、prm(m, E)＝prm(E, m)＝1とし、m—E間の双方向に有向辺を引く。このとき、辺(m, E)を“書き込み辺”という。ただし、Eもまたメソッドでmを内部的に実行するときは(m, E)を書き込み辺とは見なさない。

### 5.2.4. 要求矛盾解消とメソッド再定義

この処理は、適切なメソッド再定義とアクセス要求の変更を行いつつ要求矛盾を解消するプロセスであり、支援システムと設計者の対話形式で行われる。

要求矛盾解消は、基本的にアクセス要求の削除により行う。ただし、他のメソッド(“代替メソッド”）の利用や新メソッドの定義によって機密要求に反しない範囲でアクセス要求が部分的に実現可能な場合は、システムはそのための支援をする。この処理は、矛盾パスの終点からパスを有向辺と反対の向きに探索し、要求の変更に伴ってグラフを書き換えながら行う。以下、その手法の主な点を述べる。詳細は付録3の疑似パスカル記法を参照されたい。

まず、システムは矛盾するアクセス要求の1つ(m, U)と対応する機密要求(E, U)を表示し、(m, U)をグラフから削除し、mの代替メソッド候補を探す。代替メソッド候補は元のメソッドの機能を部分的に実現する可能性のあるメソッドであり、厳密には、

{ms ¦ ((msはmに対する新メソッド)∨(∀v(prm(v, m)＝0 ⇒ prm(v, ms)＝0)∧
∃v(prm(v, m)＝1∧prm(v, ms)＝1∧((vは変数)∨(vは要素クラス)))))∧
(∀v(srm(v, U)＝1 ⇒ rbm(v, ms)＝0))}

なるmsである。なお、“mに対する新メソッド”は後述の処理によって追加されるメソッドのことであり、最初は存在しないが、2度目以降にmに対する代替メソッドを探索する際にはその候補となる。システムは代替メソッド候補の一覧を表示し、「処理要求(m, U)を諦める。」か、「mの代替メソッド候補の何れかを選択する。」か、「新メソッドを作成する。」を設計者に問う。設計者が「処理要求(m, U)を諦める。」を選んだ場合は(m, U)に関する矛盾は解消して終了。代替メソッド候補の何れかを選択すれば、システムはそのメソッドmsに関して新たな辺(ms, U)をグラフに追加して終了する。

もし処理要求を諦めず、かつ、適当なmの代替メソッド候補がない場合は、設計者は「新

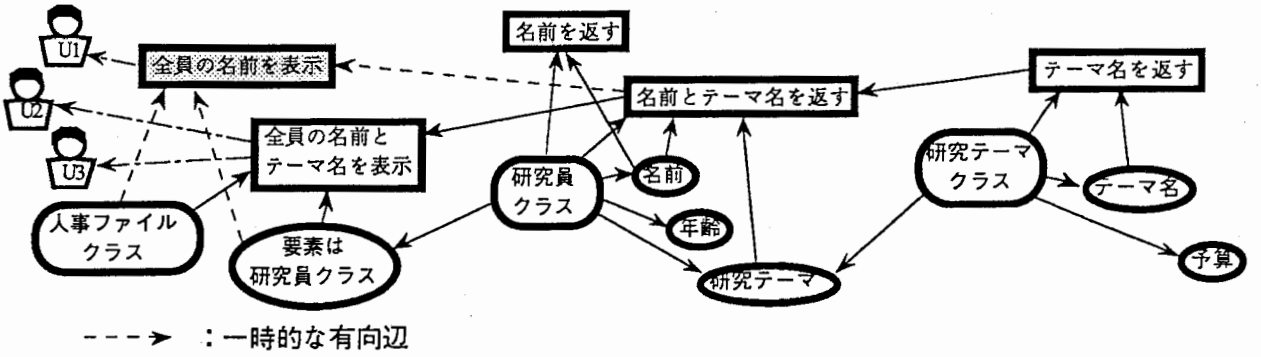



図9　要求矛盾解消とメソッド再定義

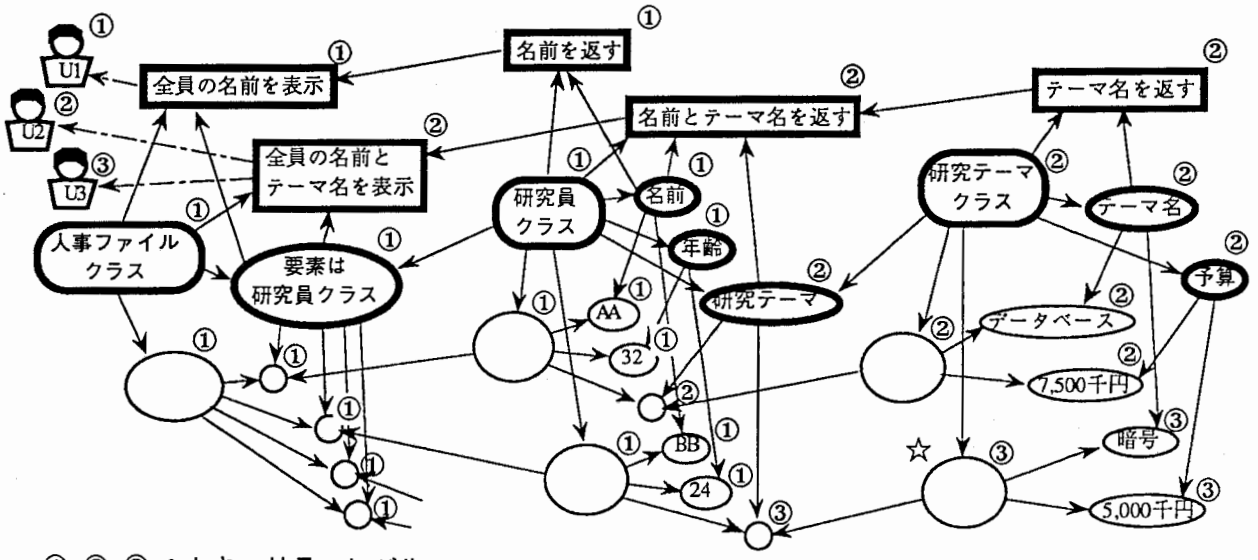



図10　セキュリティレベル設定

メソッドを作成する。」を選ぶ。するとシステムは設計者に新メソッドの名前の入力を促し、その新メソッドmnを新しい頂点として処理要求グラフのデータ構造に入れる。mnを“（元の）メソッドmに対する新メソッド”という。更に、システムはmに接続する全頂点vtを一時的にmnにも接続させる（辺(vt, mn)を加える）。これら全てのvtにつき、ここまでの処理を再帰的に行う。即ち、vtへ到達可能な頂点vの中にsrm(v, U)＝1となるものが存在しなければ(vt, mn)をそのままにする。そのようなvが存在する場合は、vtがメソッドでかつ代替メソッドや新メソッドによる改善の可能性があれば(vt, mn)に関して(m, U)のときと同様の矛盾解消処理を続行、改善不可能な場合（例えばsrm(vt, U)＝1のとき）は(vt, mn)を改めて削除する（注2）。

図9に、図8の例の矛盾を解消するために新メソッド“全員の名前を表示”を定義してU1に接続し、この新メソッドから元のメソッド“全員の名前とテーマ名を表示”に接続する頂点に一時的な辺を加えた状況を示す。更に、“研究員クラス”のメソッド“名前とテーマ名を返す”から新メソッドへの接続が機密要求に反するため、その代替メソッドとして同じクラスの“名前を返す”を選択した状況を図10に示す。

以上の処理を全ての矛盾するアクセス要求について行う。なお、一旦定義したメソッドを後で破棄することもできる。

（注2）元メソッドmが書き込み系メソッドで書き込み辺(m, vt)がある場合、新メソッドからも書き込み辺(mn, vt)を引く。このとき、mnへはvtの代替メソッドを接続できないとする。もし接続すると、mnから代替メソッドへ向かう新たな書き込み辺が加わり、元々矛盾のない部分にも矛盾が生じる可能性があるため。

### 5.2.5. セキュリティレベル設定

この処理は、処理要求グラフの半順序関係を満足するように、各頂点（エンティティ）に半順序レベルを設定するプロセスで、全て支援システム内部で行われる。。

まず前処理として、システムは機密要求に反しない範囲で、処理要求グラフの各頂点vから各ユーザーUへの新たな辺(v, U)を加える。更に、機密要求に反しない範囲で任意の頂点間に新たな辺を最大限加える。こうして出来たグラフに対応する行列を“極大アクセス許可行列”という。これらの処理は、レベル数をなるべく減らし、かつ、機密要求に反していないのにメソッドの実行結果が全く表示されないような無意味なレベル設定を防止するために行う。

この後、半順序レベル設定を行う。半順序は“自然数のn個組”で表現する。これは、n個の自然数をN1, N2, …, Nnとしたとき、(N1(a), N2(a), …, Nn(a))と (N1(b), N2(b), …, Nn(b))の2つのn個組に関し、N1(a)≦N1(b)かつN2(a)≦N2(b)かつ…かつNn(a)≦Nn(b)となる場合に、(N1(a), N2(a), …, Nn(a)) ≦ (N1(b), N2(b), …, Nn(b))と順序関係を定義する方法である。“自然数n個組”設定アルゴリズムの詳細は文献[20]を参照されたい。

図10には図8, 9の例にレベル設定を施した結果も示している。ここでは更に、「U2には“テーマ名”が“暗号”であるインスタンス（☆印）を知られたくない。」という機密要求がある場合も同時に考慮したレベル設定の例を示している。この結果により、U1は“全員の名前とテーマ名を表示”を実行できない。また、U2とU3は全メソッドを実行できるが、U2はインスタンス☆に関する情報は読み出せない。

# 6. 提案手法の評価と今後の展開

## 6.1. 提案したアクセス制御機構の評価

3.3. 節で述べた問題点は、4. 節で提案したアクセス制御機構により、それぞれ次のようにして解決される。

問題点[A]は、メソッドにレベルを設定し、(13)-(18)式及び(26)-(29)式のレベル設定制約に従い、そして、メソッドのレベルより上位レベルのユーザーにのみそのメソッドを示すことにより解決できる。問題点[B]は、変数のクラスにおけるレベルとインスタンスにおけるレベルを（要素クラスのレベルと具体的な要素のレベルを）区別し、(1), (11)式((24)式）の制約を用い、そして、4.6. 節のアクセス制御規則に従えば解決できる。問題点[C]は、(2), (5), (8), (19)式により解決できる。問題点[D]は、オブジェクト単独でのレベルと、それが他の複合オブジェクトの変数値や要素であるときのレベルを区別し、(4), (7), (10), (21), (23)式の制約を用い、そして、4.6. 節のアクセス制御規則に従えば解決できる。

以上より、提案したアクセス制御機構により、強制アクセス制御のポリシー（情報フローを、下位レベル→上位レベル、及び、同レベル間に限定する）が実現出来る。

## 6.2. 提案したセキュリティ設計支援手法の評価

5. 節で提案した設計支援手法では、アクセス要求をメソッドの実行要求に置き換えてレベル設定制約とともにグラフ化し、起こり得る直接及び間接的な情報フローを全て算出し、機密要求に矛盾するアクセス要求を検出している。ここで、5.2.4. 節と付録3の“要求矛盾解消・メソッド再定義プロセス（Resolving Inconsistencies and Method Redifinition (RIR)）”によってすべての矛盾が解消出来、かつ、新たな矛盾が生じないことの大まかな証明を行う。以下、まだRIRに入力されていないグラフを“初期グラフ”、初期グラフ上の頂点を“初期頂点”、初期グラフにおいて生じている矛盾を“初期矛盾”と呼ぶ。

RIRにおいては、全ての矛盾パスの終端ユーザーに1つずつ着目し、初期矛盾の原因となるアクセス要求を全て削除している。従って、新メソッドによって生じる新たな情報フローのみを考慮し、かつ、それが新たな矛盾を起こさないことを確かめればよい。

図9おいてユーザーUに着目し、Uに関する矛盾を解消すると仮定する。また、元メソッドv1の代わりに新メソッドv8を作成したとする（図11）。初期頂点に新たに入ってくる有向辺は常に書き込み辺である（例えば(v8, v3)）。しかも、そのような辺の終点（v3）は常に新メソッド（v8）の元メソッド（v1）に接続しており、かつ、元メソッド（v1）から終点（v3）へは常に書き込み辺がある。なお、新メソッドから代替メソッドへは決して書き込み辺がないことに注意（注2）。

さて、新メソッドのみを通ってUへ流れ込むような矛盾する情報フローは生じない。なぜなら、Uに関する矛盾解消プロセスにおいては、Uに関する機密要求に反する情報フローは全て削除されるからである。残る可能性は、新メソッドから初期頂点を通ってU以外

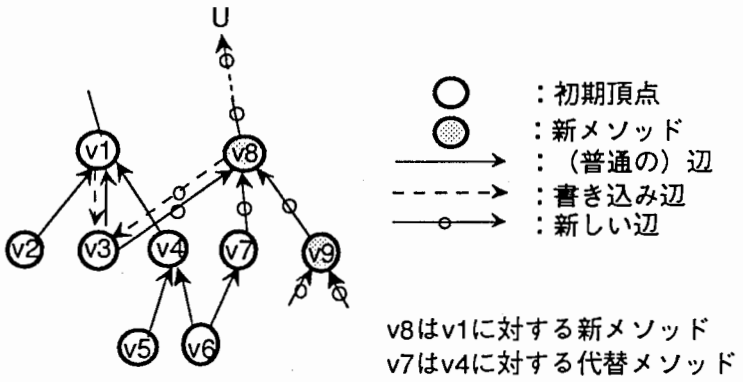



図11　新メソッドによって生じる新しい情報フロー

のユーザーへ流れ込むような矛盾する情報フローが発生することである。このようなフローが生じるとしたら、代替メソッドから新メソッドへ向かう有向辺（例えば(v7, v8)）と書き込み辺（(v8, v3)）の組による以外にない。しかし、5.2.4. 節の代替メソッドの定義より次が成り立つ（Orbmは初期グラフの到達行列の値を表す）。

$$\forall v(rbm(v, v8)=1 \wedge (v\text{はメソッドに対応する頂点でない})) \Rightarrow Orbm(v, v3)=1)$$

これは、v8を通る新しい情報フローが、メソッドの情報以外の新たな情報を初期頂点に与えることはない、ということを意味している。この議論は、全てのユーザーと全ての新メソッドに適用できるので、新たな情報フローが新たな矛盾を発生させることはない。

RIRを行った後、グラフの半順序関係を満足するようにグラフの各頂点(エンティティ)にレベルを設定する。よって、4.6. 節のアクセス制御規則に従って制御すれば、情報フローは“下位レベル→上位レベル、或は、同レベルのエンティティ間”に限定されるので、グラフの有向パス以外の情報フローは生じない。以上のことから、提案したセキュリティ設計支援手法によって、機密要求の実現と機密要求に反しない範囲でのアクセス要求の部分実現が可能になる。

### 6.3. 今後の展開

今後の課題としては、次のようなことが挙げられる。

- メソッド実行禁止要求やデータの改竄防止要求への対応
- 変数の結合情報に関する機密要求への対応
- メソッド処理の条件分岐やレベルの時間変化の考慮
- BLPモデル以外のモデルに基づくアクセス制御機構やセキュリティ設計支援手法の確立
- 推論による情報フローの考慮

・通常のOODB設計との結合

この中で、“変数の結合情報に関する機密要求への対応”については付録4で若干の議論をしているのでそちらを参照されたい。

また、1. 節でも触れたように、最近ではネットワーク管理をオブジェクト指向的手法を用いて行うというのが主流になりつつあり、CCITTから勧告草案が既に出ている（文献[21]）。本研究での提案手法のネットワーク管理への応用が期待される。

## 7. まとめ

本研究では、オブジェクト指向データベース（OODB）におけるセキュリティ上の問題点を明らかにし、それを強制アクセス制御で解決するためのアクセス制御機構を提案した。そこでは、セキュリティレベル設定制約やアクセス制御規則を明らかにした。

そこでは、メソッドにもレベルを設定し、上位レベルのユーザーに対してのみそのメソッドを表示することにより、クラスや変数の存在を機密にすることが可能になった。また、クラス（変数）と具体的な個々のインスタンス（変数値）間や、オブジェクト単独とそれが他の複合オブジェクトの変数値や要素であるときの間に機密度の差がある場合の対処法についても考察した。これらにより、強制アクセス制御のポリシーを実現出来た。

また、OODBのセキュリティ設計支援手法を提案した。そこでは、OODB設計者と支援システムの対話的環境の下で、ユーザーのデータへのアクセス要求と機密漏洩防止要求が与えられた場合に、後者を優先的に実現し、また、後者に反しない範囲でアクセス要求を部分的に実現するためのアルゴリズムを提案した。具体的には、要求をグラフ化して要求間の矛盾検出・解消・適切なメソッドの再定義を行い、ユーザーとデータにセキュリティレベルを設定する。これにより、多量のデータや複雑なメソッドをもつOODBのセキュリティ設計が容易に行え、かつ、見落としや誤りの発生が防止できる。

今後の課題としては、データの改竄防止要求や変数の結合情報に関する機密要求にも対応できる支援アルゴリズムの構築や、通常のOODB設計との融合等が挙げられる。また、ネットワーク管理への提案手法の応用が期待できる。

# 参考文献


[1] 増永良文, "次世代データベースシステムとしてのオブジェクト指向データベースシステム," *情報処理*, vol. 32, no. 5, pp. 490-499, May 1991.

[2] 牧之内顕文, "オブジェクト指向データベース," *情報処理*, vol. 32, no. 9, pp. 1032-1040, Sep. 1991.

[3] Atkinson, M., Bancilhon, F., Witt, D. D., Dittrich, K., Maier, D. and Zdonik, S., "The Object-Oriented Database System Manifesto," *The First International Conference on Deductive and Object-Oriented Databases*, pp. 40-57, Dec. 1989.

[4] Hsiao, D. K., "An Object-Oriented Approach to Security Policies and their Access Controls for Database Management," *The Second Far-East Workshop on Future Database Systems*, pp. 8-25, Apr. 1992.

[5] Keefe, T. F., Tsai, W. T. and Thuraisingham, M. B., "SODA: A Secure Object-Oriented Database System," *Computers and Security*, vol. 8, no. 6, pp. 517-533, 1989.

[6] Lunt, T. F., "Multilevel Security for Object-Oriented Database Systems," *Proc. of the Third IFIP WG 11.3 Workshop on Database Security*, Sep. 1989.

[7] Thuraisingham, M. B., "A Multilevel Secure Object-Oriented Data Model," *12th NCSC*, pp. 579-590, 1989.

[8] Jajodia, S. and Kogan, B., "Integrating an Object-Oriented Data Model with Multilevel Security," *1990 IEEE Symp. on Security and Privacy*, pp. 76-85, 1990.

[9] 上林弥彦, "オブジェクト指向によるセキュリティモデル," *信学技報*, COMP89-112, Feb. 1990.

[10] Varadharajan, V. and Black, S., "Multilevel Secrity in a Distributed Object-Oriented System," *Computers and Security*, vol. 10, no. 1, pp. 51-68, 1991.

[11] Fernandez, E. B., Gudes, E. and Song, H., "A Security Model for Object-Oriented Databases," *1989 IEEE Symp. on Security and Privacy*, pp. 110-115, 1989.

[12] Lunt, T. F., "Discretionary Security for Object-Oriented Database Systems," *SRI International Final Report*, Oct. 1990.

[13] Kelter, U., "Discretionary Access Controls in a High-Performance Object Management System," *1991 IEEE Symp. on Security and Privacy*, pp. 288-299, 1991.

[14] Millen, J. K. and Lunt, T. F., "Security for Object-Oriented Database Systems," *1992 IEEE Symp. on Security and Privacy*, pp. 260-272, 1992.

[15] Araki, T., Chikaraishi, T., Hardjono, T., Ohta, T. and Terashima, N., "An Access Control Mechanism for Object-Oriented Database Systems," *IEICE Trans. Fundamentals*, vol. E76-A, no. 1, Jan. 1993.

[16] 荒木禎史, 力石徹也, Thomas Hardjono, 太田理, "オブジェクト指向データベースのセキ

ュリティ設計支援手法," *信学会 1993年暗号と情報セキュリティシンポジウム(SCIS93)*, SCIS93-14A, Jan. 1993.
[17] ドン・パーカー（日本情報処理開発協会監訳）, "コンピュータ・セキュリティ(邦訳)", 企画センター, 1982.
[18] Bell D. E. and LaPadula L. J., "Secure Computer System: Unified Exposition and Multics Interpretation", *MITRE Corp.*, MTR-2997, 1976 (available as NTIS AD-A023588).
[19] Araki, T., Morizumi, T., Nagase, H., Takenaka, T. and Yamashita, K., "Security Level Assignment by Graph Analysis," *IEICE Trans.*, vol. E74, no. 8, Aug. 1991.
[20] 荒木禎史, 永瀬宏, 竹中豊文, 山下紘一, "アクセス制御による情報処理システムのセキュリティ実現法," *信学技報*, ISEC91-9, Jul. 1991.
[21] CCITT Recommendation M.3010, "Principals for a Telecommunications Management Network," 1992.

付録1

# 代表的なセキュリティモデル

## 1. Bell and LaPadulaモデル

Bell and LaPadulaモデル（BLPモデル）（文献[1]）は“強制アクセス制御（mandatory access control）”の考えに基づく最も代表的なセキュリティモデルの一つであり、1976年に米国MITRE社より提案された。これは主として軍用システムや政府調達用システム向けに考案されたモデルで、“機密の漏洩防止（control of disclosure）”に重点をおいている。

強制アクセス制御とは、情報処理に関わる全ての主体・客体にレベルを設定し、その値によってセキュリティ核がアクセス制御を行う、というものである。

BLPモデルでは、システムのセキュリティ状態を、（アクセス集合b、アクセス許可行列M、レベルf、階層構造H）の組で表現しているが、ここでは、アクセス集合とレベルを取り上げる。

アクセス集合の各要素は、（主体、客体、アクセス属性（access attribute））からなり、アクセス属性にはread, append, write, executeが入る（writeは読みと変更、appendは追加のみ、executeは読みも変更も追加も無し）。例えば(S, O, read)は、主体Sが客体Oをreadする、というアクセスを表す。

レベルは半順序構造をなし、（等級（clearance）、カテゴリー（category））という2つの要素からなる。等級は各主・客体の地位や機密度に対応し、カテゴリーは所属組織等に対応すると考えられる。そこでの順序関係≦は、等級をC、カテゴリーをSとすると、C1≦C2かつS1⊆S2のときに(C1, S1)≦(C2, S2)と定義される。

さて、BLPモデルの基本的なセキュリティポリシーは、次の2つの特性で述べられている。

1）単純セキュリティ特性（simple security property）

主体が客体の情報を読むときは、主体のレベルは客体のレベル以上でなければならない。

2）*特性（*-property）

（単純セキュリティ特性が満足されていて、かつ、）主体が客体1の情報を読み、かつ、客体2の情報を変更するときは、客体1のレベルは客体2のレベル以下でなければならない。

即ち、情報フローを下位レベル→上位レベル、または、同レベル間に限定する、ということである。従って、システムのアクセス集合の要素(S, O, attribute)はいかなる状態でも次の各条件を満足せねばならない（levelは主体・客体のレベルを表す）（図1）。

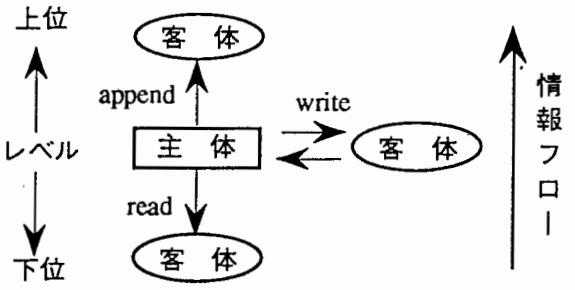



図1　BLPモデルの＊特性

level(O)≧level(S)　　if attribute is append
level(O)＝level(S)　　if attribute is write
level(O)≦level(S)　　if attribute is read

BLPモデルでは、アクセス集合への要素の追加、削除（新アクセスの追加・削除）により状態遷移が起こるとし、いかなる状態でも上記の特性（*特性）が満足されるための状態遷移の条件を“基本セキュリティ定理（basic security theorem）”として証明している。その概要は次のとおりである。

1）初期状態は*特性を満たす
2）追加した新アクセスは次状態で*特性を満たす
3）次状態で*特性を満たさなくなったアクセスは削除される

BLPモデルでは、更に、これらの特性や条件を実現するための具体的なアクセス制御手法（セキュリティ核の動作規則）も述べられている。例えば、主体Sが客体Oのreadアクセス要求を出したら、(S, O, read)がアクセス集合にあるか、或は、level(S)≧level(O)であるかをチェックし、これらが満足されていれば、アクセス要求を受け入れて次状態に遷移する。そうでなければ要求を拒否して状態遷移は行わない、等である。

前述のように、BLPモデルは“機密漏洩の防止”に重点をおいたモデルであり、実際、上記のメカニズムに従えば、上位レベルから下位レベルへの情報フローは生じない。

しかし、“改竄の防止（writeの防止）”まできめ細かく制御するのは不可能な場合もある。例えば、主体S1が客体O1を、S2がO2を、それぞれread, writeでき、かつ、S1はO2をreadできるがwriteできない、S2はO1をreadできるがwriteできない、というような条件を満たすレベルの設定は不可能である。

## 2. Take-Grantモデル

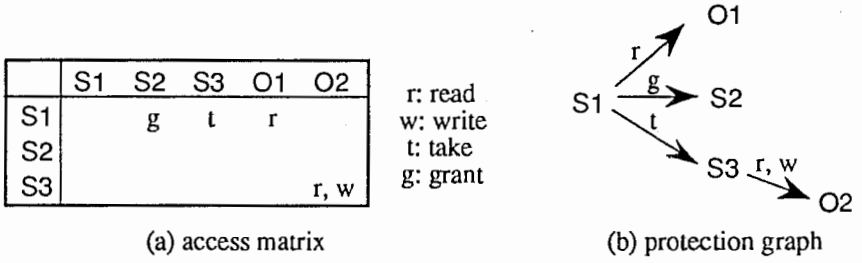
Take-Grantモデル（文献[2]）は、“任意アクセス制御（discretionary access control）”の考えに基づくモデルであり、L. Snyderによって提案された。任意アクセス制御とは、各主体が客体に対して、或は、主体間毎にそれぞれ持つ権限を定義し、その権限を記述したアクセス許可行列（access matrix）に従ってセキュリティ核がアクセス制御を行う、というものである。

Take-Grantモデルでは、read, write, append, execute, take, grantの6種類の権限を扱っている。このうち、take, grantを除く残りのものはほぼBLPモデルと同様の意味である（take, grantの意味は後述）。これらの権限を用いて記述したアクセス許可行列の例を図2(a)、及び、それをグラフ化して記述したもの（protection graph）を図2(b)にそれぞれ示す。

モデルでは、更に、グラフの変形に関する以下の4つの規則を定めている。

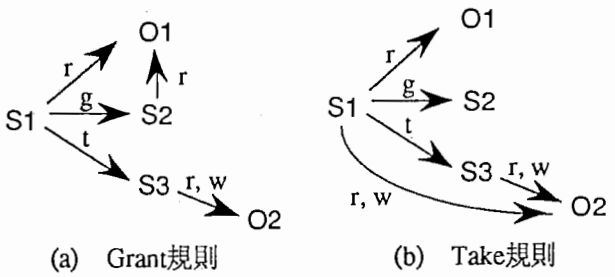
Grant：主体xがyに対してgrant権を持つとき、xがzに対して持っている権限をyに譲渡（grant）できる。
Create：主体は新たな客体を生成し、それに対して適当な権限を持てる。
Take：主体xがyに対してtake権を持っているとき、yがzに対して持っている権限を得る（take）ことができる。
Remove：主体は自分が他の主・客体に対して持っている権限を放棄できる。

上記のうち、図2の例についてGrant, Takeを施した結果のprotection graphを図3(a), (b)にそれぞれ示す。




(a) access matrix　　(b) protection graph

図2　access matrixとprotection graph




(a)　Grant規則　　(b)　Take規則

図3　Grant規則とTake規則

Take-Grantモデルの適用に当たっては、情報のフロー解析が重要である。例えば、図2のprotection graphでは、S2はO1のread権がないのでこのままではO1→S2の情報フローは生じない。しかし、Grant規則により図3(a)のようにS1からS2へO1のread権が譲渡されると、O1→S2と情報フローする。これ以外にも、主体・客体間でのread, write, appendの組み合せにより、間接的な情報フローの起こる可能性がある。

従って、Take-Grantモデル上で情報漏洩についての安全性を保証するには、権限の初期設定だけでなく、そこから起こり得る全ての間接的な情報フローを解析する必要がある(ここが、レベルを設定すれば情報フローが下位→上位、及び、同レベル間のみに限定されることが保証されるBLPモデルと大きく異なる点である)。ところが、権限の初期設定だけでは、どのような間接的情報フローは良く、また、どのようなフローは認められないかの判断規準が分からない。このことは、Take-Grantモデルでは、“どのようなセキュリティを保証したいのか”が明確でない、即ち、セキュリティポリシーが明確でない、ということでもある。

ただし、Take-Grantモデルでは、read, write, append権をそれぞれ独立に設定可能なので、改竄の防止に関しては、BLPモデルよりきめの細かい制御ができる。

## 3. Clark-Wilsonモデル

Clark-Wilsonモデル(文献[3])は、商業用システム向けに考案されたセキュリティモデルで、BLPモデルやTake-Grantモデルが機密の漏洩防止に重点をおいているのに対し、“情報の完全性の維持(providing integrity)”を主要なセキュリティポリシーとしている点が特徴である。

このポリシーを実現するための基本的なメカニズムは次の2つである。

1) well-formed transaction

データには、そのintegrityを保つことができるとみなされた適当な手段によってのみ、アクセスできる。即ち、各データには決められたプログラムのみアクセスできる。

2) separation of duty

異なるユーザーが異なる処理を行う。即ち、各ユーザーは決められたプログラムのみ実行できる。

1)は、システム内部での処理の一貫性、2)は外部での一貫性、をそれぞれ保証するものであるといえる。

Clark-Wilsonモデルでは、これらのポリシーやメカニズムを次のような構成要素と規則を用いてモデル化した。

まず、モデルの構成要素は次のとおりである。

CDI（constraind data item）：integrityが保たれているデータ
UDI（unconstraind data item）：integrityが保たれていないデータ
IVP（integrity verification procedure）：全てのCDIのintegrityが保たれていることを検証するプロセス
TP（transformation procedure）：CDI, UDIを入力として、CDIを出力するプロセス

規則には、システム外でセキュリティ担当者（security officer）やシステム管理者（system owner）が人手で行うための規則（certification）と、システム内の動作規則（enforcement）の2種類がある。certificationをC、enforcementをEとして、以下、規則の概略を示す。

C1：IVPは、全てのCDIのintegrityが保たれていることを保証できねばならない。
C2：TPは、それに入力されたCDIのintegrityが保たれるような処理でなければならない。更に、各TPがアクセスできるCDIのリスト(TP, (CDIa, CDIb, …))を作成せねばならない。
E1：システムはC2でのリストを保持し、リスト内のアクセスのみ許可せねばならない。
E2：システムは各ユーザーが実行できるTPのリスト(UserID, TP, (CDIa, CDIb, …))を保持し、リスト内の実行のみ許可せねばならない。
C3：E2でのリストは、separation of dutyの要求に合致せねばならない。
E3：システムはユーザーIDを認証せねばならない。
C4：TPは、ログ記録用の追加専用CDIに処理過程を記録せねばならない。
C5：UDIを入力とするTPは、UDIをCDIに変換して出力するか、或は、UDIの入力を拒否して処理を行わない、の何れかでなければならない。
E4：システムは、entityをcertifyしたユーザーにしかE2のリストの変更を認めてはならない。更に、そのユーザーには、そのentityに関与するプログラムを実行させてはならない。

Clark-Wilsonモデルの特徴は、まず、前述のように情報の完全性の維持を目標としている点である。これは、機密の漏洩防止より不当な改竄防止に重点をおいているということでもある。従って、制御メカニズムも、単に客体へのアクセス制御だけではなく、transaction（TP）の実行制御や実行履歴の記録、更に、IVPによる一種の監視等が含まれている。

加えて、規則の中にはsystem owner等の人手の作業に関する規則（certification）があることも特徴といえる。しかし、どのIVPをどのUDIに作用させるとintegrityが保証されるかの判断は、この人手の作業にまかされることになっており、なんら形式的、一般的な条件はない。このことは、Clark-Wilsonモデルが形式的モデルとしてはまだ不十分であることを示す、とも考えられる。

## 4. Chinese Wallモデル

Chinese Wallモデル（文献[4]）も、商業用システム向けに考案されたセキュリティモデルであり、互いに競合関係にある企業間にまたがるような情報とその処理を対象としている。その基本的なポリシーは、ユーザーは自分が既に持っている情報と競合しないような情報にのみアクセスできる、ということである。従って、その実現メカニズムにおいては、各ユーザーのデータへのアクセス履歴の記録が重要な意味を持つ。

Chinese Wallモデルでは、まず客体を3階層に分ける（図4）。最下層は個々のファイル等の情報を指し、objectという。中間層は各企業に相当し、company datasetという。最上層は企業群に相当し、conflict of interest classという。全てのobjectはcompany datasetの何れかに属し、全てのcompany datasetはconflict of interest classの何れかに属す。同一のconflict of interest classに属する企業は、互いに競合関係にある。

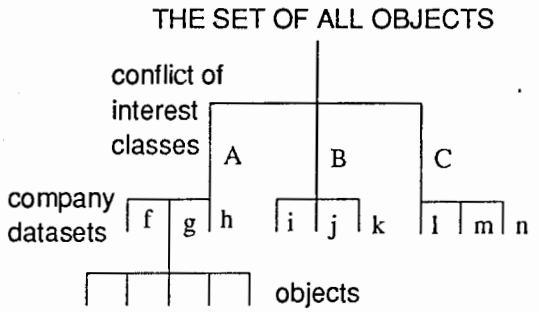



図4　Chinese Wallモデルでの客体の階層構造

Chinese Wallの意味は次の通りである。ユーザーは、最初は（どのデータにもアクセスしていない状態では）全てのデータにアクセスが許可される。しかし、一旦あるデータ（object）にアクセスすると、そのobjectの属するdatasetの回りにChinese Wallが生成される。そして、そのdatasetの含まれるconflict of interest classに属す他のdatasetを、“the wrong side of this wall”という。

さて、Chinese Wallモデルでは、アクセス制御ルールの記述に際し、BLPモデルとの対照のために単純セキュリティ特性（simple security property）や*特性（*-propety）の用語を使用している（ただし、こちらはセキュリティポリシーというより、むしろ実現メカニズムにおけるルールと見なせる）。それぞれの内容を次に示す。

1）単純セキュリティ特性

objectへのアクセス要求は、次の場合にのみ許可される。

a）そのobjectが、アクセスを要求した主体が既にアクセスしたobjectと同じcompany datasetに属する。

或は、

b）そのobjectが全く別のconflict of interest classに属する。

2）*特性

objectへの書き込み要求（write access）は次の場合にのみ許可される。

a）そのobjectへのアクセスが単純セキュリティ性質で許可されている。

かつ、

b）書き込みを要求した主体が、そのobjectと異なるcompany datasetに属し、かつ、unsanitized informationを含むような、いかなる他のobjectも読めない（sanitized informationとは、競合他社に流れてもよい情報。unsanitized informationは逆に機密を要する情報）。

単純セキュリティ特性は、ある企業の情報にアクセスした主体は、競合他社の情報にはアクセスできないこと。*特性は、書き込みによって競合他社の情報が間接的に漏洩するのを防ぐこと、をそれぞれ目的としている。

さて、Chinese WallモデルもClark-Wilsonモデルと同様に商業用システム向けに考案されたものであることは既に述べたが、Clark-Wilsonモデルが情報の完全性の維持を目標としていたのに対し、こちらはむしろ機密の漏洩防止に重点をおいている。また、主体のアクセス権限が変化するという点は、BLPモデルよりTake-Grantモデルに近いともいえる。

しかし、こちらではTake-Grantモデルと異なりセキュリティポリシーは明確である。それは、同一主体に競合する企業の情報がフローしてはならない、ということである。

## 参考文献

[1] Bell D. E. and LaPadula L. J., "Secure Computer System: Unified Exposition and Multics Interpretation", *MITRE Corp.*, MTR-2997, 1976 (available as NTIS AD-A023588).

[2] Snyder L., "Formal Models of Capability-Based Protection Systems," *IEEE Trans. on Computers*, C30, 3, pp.172-181, Mar. 1981.

[3] Clark D. D. and Wilson D. R., "A Comparison of Commercial and Military Computer Security Policies," *IEEE Symp. on Security and Privacy*, pp.184-194, 1987.

[4] Brewer D. F. C. and Nash M. J., "The Chinese Wall Security Policy," *IEEE Symp. on Security and Privacy*, pp.206-214, 1989.

付録2

## 行列を用いた情報フロー解析

有向グラフにおいて、一般に、両端点を含まずに、m個の頂点が載っているような有向パスを“長さm＋1の有向パス”と呼ぶことにする（ただし、ここではm個の頂点が全て異なる場合のみを考える）。例えば、有向パスa→b→c→dは長さ3の有向パスである。

さて、処理要求行列（prm）をブール演算（注1）で2乗すると、処理要求グラフに含まれる全ての長さ2の有向パスが算出される。つまり、一般に処理要求行列のk乗を“$prm^k$”で表すと、prm(a, b)＝1かつprm(b, c)＝1なら、常に$prm^2$(a, c)＝1となる（無論、$prm^2$(a, b)＝1かつ$prm^2$(b, c)＝1でもある）。同様に、処理要求行列を3乗すると、全ての長さ3の有向パスが算出される。つまり、$prm^2$(a, c)＝1かつprm(c, d)＝1なら、常に$prm^3$(a, d)＝1となる。一般に、処理要求行列をk乗すると、処理要求グラフに含まれる全ての長さkの有向パスが算出される。

頂点数がn個の場合、即ち処理要求行列のサイズがnの場合、長さがn以上の有向パスはありえない。長さn－1の有向パスが最長である。よって、$prm^{n-1}$、即ち、到達行列(rbm)を算出することにより、処理要求グラフに含まれる任意の2頂点間の有向パスを全て求めることができる。従って、到達行列には、データ→ユーザー間の全ての直接及び間接的な情報フローが表されているといえる。

（注1）ブール演算とは、

| | |
|---|---|
| 0＋0＝0 | 0×0＝0 |
| 0＋1＝1 | 0×1＝0 |
| 1＋0＝1 | 1×0＝0 |
| 1＋1＝1 | 1×1＝1 |

なる演算規則を持つ演算である。

付録3

# OODBセキュリティ設計支援における
# 要求矛盾解消・メソッド再定義プロセス（RIR）

入力：処理要求行列、機密要求行列、到達行列
出力：新メソッド追加後の矛盾解消された処理要求行列

```
procedure RIR:
begin
    for 各ユーザーvu do
        RIR1(vu, vu);
        到達行列を計算し直す
end

procedure RIR1(vu, va):
begin
    for prm(vt, va)＝1である全ての頂点vt do
        begin
            if vtがメソッドでない、かつ、「rbm(v, vt)＝1、かつ、srm(v, vu)＝1」である頂点vが存在する
            then
                begin
                    prm(vt, va)←0;
                    prm(va, vt)←0
                end
            if vtがメソッド、かつ、srm(vt, vu)＝1、又は、「rbm(v, defcl(vt)*)＝1、かつ、srm(v, vu)＝1」で
            ある頂点vが存在する then
                begin
                    prm(vt, va)←0;
                    prm(va, vt)←0
                end
            else  if vtがメソッド、かつ、「rbm(v, vt)＝1、かつ、srm(v, vu)＝1」である頂点vが存在する
            then
                begin
                    if prm(va, vt)＝1 then
                        vtに関して設計者に「処理要求を諦める」か「新メソッド作成」を選択させる
                    else
                        begin
                            代替メソッド候補を探す;
                            if 代替メソッド候補が存在する then
                                設計者に、「処理要求を諦める」か「各代替メソッド候補のいずれか」か「新
                                メソッド作成」を選択させる;
                            else
                                設計者に、「処理要求を諦める」か「新メソッド作成」を選択させる
                        end;
                    if 設計者が代替メソッド候補の 1つを選択 then
                        選択した代替メソッド候補をvsとし、prm(vs, va)←1;
                    else  if 設計者が「新メソッド作成」を選択 then
                        begin
                            設計者に新メソッド名を入力させ、それに対応する頂点をvnとしてデータ構造に
                            追加する;
                            prm(vn, vn)←1;
```

```
                    for prm(v, vt)＝1、かつ、(v, vt)が書き込み辺でない全てのv do
                        begin
                            prm(v, vn)←1;
                            if prm(vt, v)＝1、かつ、(vt, v)が書き込み辺 then
                                prm(vn, v)←1、かつ、(vn, v)を書き込み辺とする
                        end;
                    RIR1(vu, vn);
                    if vnに“破棄”の印がある then
                        vn、及び、vnより後に追加した全ての頂点vをデータ構造から削除する
                    else
                        begin
                            prm(vn, va)←1;
                            if prm(va, vt)＝1、かつ、(va, vt)が書き込み辺 then
                                prm(va, vn)←1、かつ、(va, vn)を書き込み辺とする
                        end
                end;
            prm(vt, va)←0;
            prm(va, vt)←0
        end
    end;
prm(v, va)＝1であるvを表示し、設計者にvaを破棄するかどうかを選択させる;
if vaを破棄する then
    vaに“破棄”の印を付ける
end
```

*) defcl(vt)は、vtを定義しているクラスを表す

付録4

# 情報セキュリティにおける属性結合問題に関する一考察

## 1. はじめに

情報セキュリティにおける属性結合問題（Aggregation Problem）とは、アクセス対象である客体の複数の属性の結合情報の機密保持をいかに実現するか、という問題である。この問題に関しては関係データベースやオブジェクト指向データベースの分野で既に研究され、解決法が提案されている（文献[1], [2]）。しかしそれらの手法にはなお不十分な点があり、完全な機密漏洩防止が実現できない。そこで本稿では新たな解決法を提案する。そこでは、客体の各構成要素へのアクセス権限設定法や、権限の相互矛盾を防止するための権限設定制約を明確にする。この手法は、一般に属性とその値の組から構成されるような客体へのアクセス制御手法として広く応用できる。

## 2. 情報セキュリティにおける属性結合問題

一般に、情報セキュリティにおける属性結合問題とは、複数の属性とその値の組から構成される客体において、ある単独の属性データの情報より、それらが複数個結合した情報の機密度を高くしたいとき、それをアクセス制御でいかに実現するか、という問題である。

例として、図1のような構造を持つデータベースを考える。あるユーザーUに対し、名前のみの一覧や職業のみの一覧を見せてよいが、名前と職業の結合データの一覧を見せてはならないような状況を考える（つまり、どの職業に何人の人が従事しているかという統計データは知られてもよいが、誰がどの職業に就いているかという情報は知られたくないような状況を考える）。このセキュリティ上の要求を実現するためにこれまで提案された代表的な手法は、Uに対して、“名前”と“職業”の単独の属性だけでなく、“名前—職業”の結合属性にもアクセス権（Read権）を設定し、単独属性についてはアクセス権を与え結合属性については与えない、ということである（文献[1], [2]）。

| 識別子 | 名前 | 職業 |
|---|---|---|
| 1 | XX | 研究者 |
| 2 | YY | 警官 |
| 3 | ZZ | 教師 |
| … | … | … |

図1　データベースの例

しかし、この手法は属性値の一覧を見るような操作に対しては有効であるが、識別子ごとに属性を指定して属性値を見るような操作に対しては不十分である（識別子とは、個々の客体を一意に識別するための一種の属性のことである）。なぜなら、Uが各識別子ごと

に、各属性に個別にアクセスすることにより、結果的に属性の結合情報が知られてしまうからである。図1の例では、Uは識別子1のデータの“名前”を指定して“XX”を得、更に、同じく識別子1の“職業”を指定して“研究者”を得て、結果的に“XXは研究者である”という情報を得ることができてしまう。これは、識別子情報を介して結合情報が漏洩したためと考えられる。

このような問題点を解決する手法はまだ提案されていない。

## 3. 解決法

本節では、2.節で述べた問題点を解決するためのアクセス権限設定法や、権限を矛盾なく設定するための権限設定制約を述べる。

まず、ユーザーUに対し、属性の一覧に関するアクセス権と、識別子ごとの属性値に関するアクセス権を区別し、次のように表す。

R(U, att)：属性attの一覧へのUのアクセス権
R(U, att1-att2)：属性att1とatt2の結合情報の一覧へのUのアクセス権
$R_i$(U, att)：識別子iの客体の属性attの値へのUのアクセス権

これらは全て{0, 1}のいずれかの値をとる関数で、“0”ならばUはアクセス（Read）可能、“1”なら不可能、と定義する。

次に、これらのアクセス権限の設定に関する制約を考える。2.節の問題を防止するには、R(U, att1-att2)＝1のとき、$R_i$(U, att1)と$R_i$(U, att2)の少なくとも一方を“1”にせねばならない。しかも、推論による情報漏洩を防ぐため、全てのiに関して同じ属性のアクセス権が“1”でなければならない。やや極端な例であるが、図1の例で、研究者を職業とする人がXXだけしかいないとする。そこで、R(U, 名前)＝0, R(U, 職業)＝0, R(U, 名前-職業)＝1、かつ、$R_1$(U, 名前)＝0, $R_1$(U, 職業)＝1、かつ、1以外の全てのiに関して$R_i$(U, 名前)＝1, $R_i$(U, 職業)＝0、というアクセス権を設定すると、Uは容易に“XXは研究者である”ことを類推できてしまう。全てのiに関して$R_i$(U, 名前)＝1, $R_i$(U, 職業)＝0、とすればこの問題はなくなる。以上をまとめると次の制約が必要となる。

R(U, att1-att2)＝1ならば、

$$\forall i,\ R_i(U, att1)=1、または、\forall i,\ R_i(U, att2)=1 \qquad (1)$$

権限設定の相互矛盾を防ぐには、これ以外にも次のような制約が必要となる（詳細な説明は省略）。

R(U, att1)＝1、または、R(U, att2)＝1ならば、

$$R(U, att1\text{-}att2)=1 \qquad (2)$$

R(U, att)＝1ならば、

$$\forall i,\ R_i(U, att)=1 \qquad (3)$$

しかし、まだ次のような問題点がある。R(U, att)＝0かつ$R_i$(U, att)＝1なる権限設定の下では、Uは識別子iを指定して属性attの値を知ることは不可能だが、attの一覧は見ることができる。よって、iに関する属性attの値の存在を（それがiに対応することは分からないにしても）常に知ることができてしまう。図1の例によれば、R(U, 名前)＝0かつ$R_2$(U, 名前)＝1のとき、Uは、“識別子2に対応する名前はYYである”ことは知れないが、“YYと言う名前の人がいる”ことは常に知ることができる。特定のiに関する属性attの値の存在をもUに知られないようにするには、上記の方法ではR(U, att)＝1とするしかないが、今度は全てのiに関するattの値を知ることができなくなる。この問題点を解決するために、次のようにする。

$R_i$(U, att)を{0, 1, 2}の3値をとる関数に変更する。値が“0”、“1”であるときの定義は前記と同様である。“2”のときは、Uは識別子iを指定する場合も、また、属性の一覧を見る場合も、常にiの属性attの値にはアクセス不可能、と定義する。図1の例においてR(U, 名前)＝0かつ$R_2$(U, 名前)＝2のときは、Uが名前一覧を見る操作を行っても、YYは一覧の中には含まれない。

この変更にともない、制約(1), (3)を次のように改める。

R(U, att1-att2)＝1ならば、

$$\forall i,\ R_i(U, att1)\geqq 1、または、\forall i,\ R_i(U, att2)\geqq 1 \qquad (1)'$$

R(U, att)＝1ならば、

$$\forall i,\ R_i(U, att)\geqq 1 \qquad (3)'$$

以上述べた手法により、2.節で述べた問題点を解決することができる。また、この手法は、関係データベースやオブジェクト指向データベースをはじめ、一般に属性とその値の組から構成されるような客体へのアクセス制御手法として広く応用できる。

## 4. まとめ

情報セキュリティにおける属性結合問題の一解決法を提案した。今後は、この手法をオブジェクト指向データベースのアクセス制御機構やセキュリティ設計支援に応用していく予定である。

## 参考文献

[1] チャットウィチェンチャイ・ソムチャイ, 上林彌彦, "多段階関係データベースにおけるデータ保護問題," *情処学研報*89-DBS-69-1, Jan. 1989.

[2] Millen, J. K. and Lunt, T. F., "Security for Object-Oriented Database Systems," *1992 IEEE Symp. on Security and Privacy*, pp. 260-272, 1992.