データベースシステム 116−71
(1998. 7. 10)

依存関係を持つデータに関するアクセス権の管理

亀井 俊之* 依田 和也*†† 田島 敬史† 田中 克己‡

*神戸大学大学院自然科学研究科情報知能工学専攻
†神戸大学工学部情報知能工学科
‡神戸大学大学院自然科学研究科情報メディア科学専攻
†† 現在，日本アイ・ビー・エム(株)

本稿では，オブジェクトが持つ依存関係を考慮することで，予め与えられていたアクセスレベルをより適切なアクセスレベルに修正する手法について提案する．ここで，依存関係とは，例えば，あるオブジェクトを表示する場合に他のオブジェクトも表示する必要が生じる場合や，あるメソッドを実行する際に他のメソッド呼び出しが必要となる場合での関係を意味する．このような依存関係を全く考慮せずに与えられたアクセスレベルをそのまま利用した場合，依存しているオブジェクトを辿った後でそのアクセスが拒否されるようなケースが生じる可能性がある．そこで，最初に与えられたアクセスレベルを依存関係を考慮したより適切なアクセスレベルに予め修正しておくことで，効率のよいアクセス管理が可能になるものと考えられる．

Access Control for Data with Dependency Relationships

Toshiyuki Kamei* Kazuya Yoda*†† Keishi Tajima† Katsumi Tanaka‡

*Division of Computer and Systems Engineering,
Graduate School of Science and Technology, Kobe University

†Department of Computer and Systems Engineering,
Faculty of Engineering, Kobe University

‡Division of Information and Media Sciences,
Graduate School of Science and Technology, Kobe University

††IBM Japan, Ltd

This paper proposes a method for updating given access-levels to objects by taking account of their dependency relationships. An object is said to depend on another object if the latter object becomes necessary whenever the former is used. If we try to access an object based on original access-levels, access may be refused if no access rights are granted to its dependent objects. For that reason, updating suitable access-levels will make access administration effective and ultimately reduce the excessive checking of access rights at different levels.

1　はじめに

情報を提供するには一般に，様々な種類のデータが大量に必要となってくる．例えば，VRMLによってモデリングされた仮想3次元空間においてはその空間を形成するための多くのデータが存在している．これらのデータは3次元グラフィックスデータのみならず，それらに付加されるテクスチャ画像や音声データ，画像データやリンク情報など，様々なものから構成されている．そして一般には，これらのデータの間には何らかの依存関係が成り立っている．例えば，ある空間オブジェクトを表示しようとした場合に，その他のオブジェクトも表示させる必要がある場合などがそれに当たる．また，オブジェクト指向データベースにおけるあるメソッドの実行を考えた場合，このメソッドが他のオブジェクトの持つメソッドを実行するようなケースがあるが，この場合も空間データの場合と同様に依存関係が成り立っている．このように，他のデータに対して依存しているようなデータを扱う際には，これらの間に成り立っている依存関係を考慮する必要がある．

一方では，これらのデータの中には，あるユーザには参照や実行を許可するが，それ以外のユーザには許可してはいけないものが含まれる場合もあるため，データの機密性を保護する必要がある．このために，これまでに様々なアクセス権制御モデルが提案されているが[1, 2, 3]，これには大きくわけて2つのものが存在する．1つは，アクセスする主体(サブジェクト)がアクセスされる物体(オブジェクト)に対してどのような操作(メソッド)が実行可能(または実行禁止)かを組(tuple)で表現し，これを用いることで主体のアクセスを制御するモデルである．もう1つは，各オブジェクト，メソッド(以下では，これらをまとめてオブジェクトと呼ぶ)に対しそれぞれ1つのアクセスレベルを設け，これを参照することでアクセス制御を行なうモデルである．この論文においては，我々は後者のモデルを採用することにする．しかし，それらのオブジェクト間に成り立っている依存関係を全く考慮せずに各オブジェクトに対してアクセスレベルを与え，その値をそのまま利用してアクセス制御を行なうと，依存しているオブジェクトを辿った後でそのアクセスが拒否される可能性があり，結果的に効率の悪いアクセス管理になってしまう．

このような状況を避け，効率のよいアクセス管理を行なうために，本論文では，予め設定したオブジェクトのアクセスレベルを依存関係を考慮したより適切なアクセスレベルに修正し，無駄なアクセスを回避する手法を提案する．

2　オブジェクト間の依存関係およびアクセスレベル

ここでは基本的な概念である依存関係，およびアクセスレベルについて述べる．

2.1　依存関係

本稿では，それ以上分解不可能であるようなオブジェクトを基本オブジェクトと定義し，それらを表わすための変数として$a, b, \ldots$等を用いることにする．依存関係とは，「オブジェクト(メソッド) a を表示(実行)する際には必ずオブジェクト(メソッド) b も表示(実行)する必要がある」ことを意味する関係であり，

$$a \rightarrow b$$

で表現する．例えば，空間データを扱う場合には以下のような2つの依存関係が存在すると考えられる．

- **描画的依存関係**

  例えば，仮想3次元空間において，あるオブジェクトの一部分をクリックすると文書データが表示されるような場合，この文書データはそれ自身では仮想3次元空間に表示できず，表示させるためにはこのデータが付加されるべき形を持ったオブジェクトが少なくとも1つ存在しなければならない．このとき，この文書データは，このオブジェクトに依存している．このように，実際に表示する際に矛盾が起こらないための制約から描画的依存関係が生じる．

- **意味的依存関係**

  描画的な観点からはそれ単独での表示は可能だが，意味的に捕らえると，それ以外のオブジェクトと共に表示する必要がある場合にこ

の意味的依存関係が生じる．例えば，ある建物が全体の形状を表わすオブジェクトと窓などを表わす部品オブジェクトから構成されており，これらのオブジェクトを仮想3次元空間内において表示させる場合を考える．この時，この建物の形状を表わすオブジェクトはそれ単独で表示可能ではあるが，そのオブジェクトのみが表示されても，ユーザにとってそれが建物なのかただの箱状の物体なのか判断できない．この建物の形状を表わすオブジェクトと窓などを表現する部品オブジェクトが共に表示されて初めてこのオブジェクトは建物であることが分かる．このとき，これらのオブジェクト間には意味的な依存関係が成り立っている．

また，ある建造物を構成する部品オブジェクトp, q_1, q_2, q_3があるとする．このとき，部品オブジェクトpを表示する場合，ある条件が成り立つ場合にはq_1と共に表示させ，また他のある条件が成り立つ時にはq_2, q_3と共に表示する必要がある場合，依存関係は論理記号 $\wedge, \vee$ を用いて次のように表現される．

$$p \rightarrow q_1 \vee (q_2 \wedge q_3)$$

ここで，$q_1 \wedge q_2$のように基本オブジェクト$p, q, \ldots$と論理記号$\wedge, \vee$のみで表現されたオブジェクトを複合オブジェクトと呼び，P, Qなどの変数によって表現することにする．また，基本オブジェクトは複合オブジェクトの特殊なものであるといえる．このような論理式表現を用いることで任意のオブジェクトとそれらの間における依存関係が表現でき，任意の依存関係を表現すると

$$P \rightarrow Q \tag{1}$$

となる．この(1)における左辺，および右辺をそれぞれ$and-or, or-and$標準形に変換すると，

$$P_1 \vee \ldots \vee P_m \rightarrow Q_1 \wedge \ldots \wedge Q_n \tag{2}$$

のようになる．ここで，各$P_i (i = 1, \ldots, m)$は論理記号として$\wedge$のみを含み，また各$Q_j (j = 1, \ldots, n)$は論理記号として$\vee$のみを含んでいる．さらに，依存関係において

1. $P \rightarrow Q_1 \wedge Q_2 \ldots \wedge Q_n$

2. $P_1 \vee P_2 \vee \ldots \vee P_m \rightarrow Q$

はそれぞれ

1' $P \rightarrow Q_1, P \rightarrow Q_2, \ldots, P \rightarrow Q_n$

2' $P_1 \rightarrow Q, P_2 \rightarrow Q, \ldots, P_m \rightarrow Q$

に分解可能であるため，依存関係の一般形は(3)のようになる．

$$p_1 \wedge p_2 \wedge \ldots \wedge p_m \rightarrow q_1 \vee q_2 \ldots \vee q_n \tag{3}$$

また，依存関係は推移律を満たすので，任意の複合オブジェクトP, Q, Rについて次のことが言える．

$$P \rightarrow Q \text{ かつ } Q \rightarrow R \text{ ならば } P \rightarrow R \tag{4}$$

2.2 依存関係グラフ

2.1節で述べたように，依存関係の一般形は(3)のように表現されるが，この式における関係をグラフ表現した依存関係グラフは次の実線矢印のANDエッジ，点線矢印のORエッジ(図1)の2つの要素から構成される．

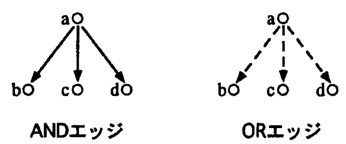

図 1: AND,OR edge

図1の左図では，$a \rightarrow b \wedge c \wedge d$，(つまり$a \rightarrow b, a \rightarrow c, a \rightarrow d$)を，また，右図では，$a \rightarrow b \vee c \vee d$を表わす．次にこの2つのエッジを利用して依存関係

$$a \wedge (b \vee c) \wedge d \rightarrow p \vee (q \wedge r)$$

を表現した依存関係グラフを考える．この依存関係を標準形に直すと，

$$a \wedge b \wedge d \rightarrow p \vee q$$
$$a \wedge b \wedge d \rightarrow p \vee r$$
$$a \wedge c \wedge d \rightarrow p \vee q$$
$$a \wedge c \wedge d \rightarrow p \vee r$$

となる．また，このように標準形に変形した結果，左辺，右辺を構成する複合オブジェクトについては，それらを構成する基本オブジェクトに対して論理記号$\wedge, \vee$に従ったエッジがでるように依存関係グラフを描くことにする．結果として得られる依存関係グラフを図 2に示す．

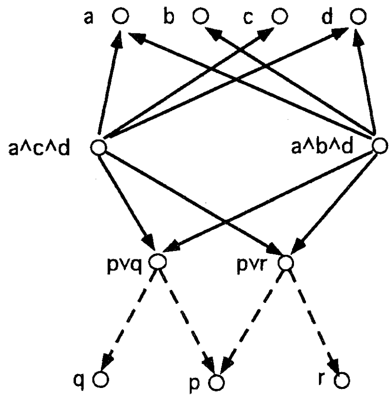

図 2: An example of dependency relationship graph

2.3 アクセスレベル

全てのアクセスレベル間で比較可能であるような全順序のアクセスレベルを扱う場合もあるが，話をより一般化するために，ここでは半順序のアクセスレベルを扱うことにする．

2.3.1 半順序アクセスレベル

ユーザは各オブジェクトに対して，1つのアクセスレベルを与えるとし，それらを$l^1, l^2, \ldots, l^n$で表わす．このように表記されたものを特に基本アクセスレベルと呼ぶことにする．そして，l^jがl^i(ただし$i \neq j$)とアクセスレベルが等しいか高い場合，これらの間の関係を$l^i \leq l^j$で表現する．

また，後に述べるアクセスレベルの自動修正により，各オブジェクトは，基本アクセスレベルと和と積の論理演算からなるアクセスレベルをもつため，このようなアクセスレベルの定義を，

$$L ::= l \mid L + L \mid L * L$$

とし，Lを単にアクセスレベルと呼ぶことにする．但し，$*, +$はそれぞれ論理演算の和，積に対応する記号であり，基本アクセスレベルはアクセスレベルの特殊なものといえる．さらに，$L = L^1 + L^2 + \ldots + L^n$とした場合，$L^1, L^2, \ldots, L^n$をアクセスレベルの構成要素と呼び，$L^i \lhd L$と表わす．$L = L^1 * L^2 * \ldots * L^n$の場合は$L^1 * L^2 * \ldots * L^n$が構成要素となる．

次に，このように拡張されたアクセスレベル間の半順序について定義する．今，アクセスレベルL^A, L^Bが与えられていたときに，L^AとL^B間の半順序を，

- $L_A = L^{a1} + L^{a2} + \ldots + L^{an}$の場合

$$L^A \leq L^B \equiv \exists L^{ai} \lhd L^A, \exists L^{bj} \lhd L^B . L^{ai} \leq L^{bj}$$

と定義し，

- $L_A = L^{a1} * L^{a2} * \ldots * L^{an}$の場合

$$L^A \leq L^B \equiv \forall L^{ai} \lhd L^A, \exists L^{bj} \lhd L^B . L^{ai} \leq L^{bj}$$

と定義する．そしてこの関係によって順序づけされた結果，極小，極大となるアクセスレベルを求め，それらの集合を求める演算をmin，また，それらを記号$+$で結合させたアクセスレベルを得る演算としてMinを定義する．つまり，アクセスレベルの要素の集合$S = \{L^1, L^2, \ldots, L^n\}$があり，半順序関係において極小となるアクセスレベルの要素の集合を$\{L'^1, L'^2, \ldots, L^m\}$とすると，

$$min(S) = \{L'^1, L'^2, \ldots, L'^m\}$$
$$Min(S) = L'^1 + L'^2 + \ldots + L'^m$$

となる．これと同様にして，集合Sの要素のうち，極大となるものの集合を$max(S)$，それらの要素を$+$で結合させたアクセスレベルを得る演算として，$Max(S)$を定義する．例えば，後に示す図4のように各基本アクセスレベルが半順序関係を満たしている場合，アクセスレベルの集合Sを$S = \{l^1 + l^2, l^4 + l^6, l^4 * l^2 + l^5\}$とした場合，

$$l^1 + l^2 \leq l^4 + l^6 \leq l^4 * l^2 + l^5$$

より，

$$min(S) = \{l^1 + l^2\}$$
$$Min(S) = l^1 + l^2$$

となる．また，$S=\{l^1+l^2, l^4+l^6, l^3\}$とした場合，

$$l^1+l^2 \leq l^4+l^6,\ l^3 \leq l^4+l^6$$

であるが，l^1+l^2とl^3の間の半順序は成立しないため，

$$min(S)=\{l^1+l^2, l^3\}$$
$$Min(S)=\{l^1+l^2+l^3\}$$

となる．

また，Sをアクセスレベルの集合とし，TをSの部分集合とした場合，Sの要素L^cがTのどの要素L^xに対しても，$L^x \leq L^c$となるとき，L^cはTの上界といい，Tの上界全体の集合を$ub(T)$で表現する．さらにこの上界集合$ub(T)$に最小元L^aがあれば，L^aはTの最小上界(least upper bound)といい，$lub(T)$で表わす．しかし，lubは常に存在するとは限らない．そこで，本稿では，このような場合におけるlubを以下のように定義する．

$$lub(T)=\begin{cases} Min(ub(T)) & if\ ub(T)\ exists \\ \{L_1 * L_2 * \ldots * L_m & \\ |L_1, L_2, \ldots, L_m \in max(T)\} & else \end{cases}$$

3 アクセスレベルの自動修正

ユーザが各オブジェクトに対して与えたアクセスレベルをオブジェクトの依存関係に基づいて自動的に修正するアルゴリズムについて述べる．

3.1 アクセスレベルの付与および自動修正

ユーザの付与することのできるアクセスレベルが$l^0, l^1, \ldots, l^{r-1}$の$r$種類あり，それらはある半順序関係が与えられているものとする．ユーザは各オブジェクトに対して，1つの基本アクセスレベルを付与する．但し，ここでは，各ユーザがアクセスレベルを付与する際には，オブジェクト間の依存関係を全く考慮せずに行なうものとする．また，アクセスレベルが更新される前の状態においては，各オブジェクトに対してただ1つの基本アクセスレベルしか与えられていないが，更新されてゆくにつれて基本アクセスレベルと記号$+, *$からなるアクセスレベルを持つようになるため，以下のアルゴリズムにおいては，各オブジェクトo_iに対してアクセスレベル$L(o_i)$が与えられているものとする．

アクセスレベルの自動修正アルゴリズム

1. **依存関係グラフから閉路を除去**

   オブジェクト間の依存関係を示す依存関係グラフにおいて閉路が生じている部分は強連結部分とみなし，それらをまとめて新しい1つのオブジェクトを生成し，以前のグラフにおける依存関係を崩さないように新しい依存関係グラフを生成する．また，この際に，新しく生成したオブジェクトに対するアクセスレベルは次のように定義することにする．閉路を構成していたオブジェクトo_1，o_2，…，o_jに対し与えられていた各アクセスレベルを$L(o_1)$，$L(o_2)$，…，$L(o_j)$とする．このとき，新しく生成された代理オブジェクトo_{New}に対するアクセスレベル$L(o_{New})$は，

   $$L(o_{New})=lub(\{L(o_1), L(o_2), \ldots, L(o_j)\})$$

   とする．ただし，lubは，以前に述べたように，最小上界(least upper bound)が存在するときはその値を，存在しない場合は，その上界集合の極小元を求める演算を意味する．また，閉路を除去された依存関係グラフは，いわゆるDAGとなる．

2. **オブジェクトに対して更新手続きを行なう**

   最初に，ルートとなるオブジェクトに対して更新手続きUpdateを適用する．Updateの手順は次のようになる．

   i. **子オブジェクトに対してUpdateを適用する**

      そのオブジェクトの各子オブジェクトに対してUpdateを再帰的に呼び出す．

   ii. **アクセスレベルを更新する**

      子オブジェクトがなければ，そのオブジェクトo_iの更新後のアクセスレベル$L'(o_j)$は更新前のアクセスレベル$L(o_j)$とする．もし，子オブジェクトが存在する場合は，親オブジェクトと子オブジェクトがどのような依存関係を持っているかを調べ，その依存関係に基づいてアクセスレベルの更新を行なう．その場合分けについては以下の通りである．

(a) **ANDエッジの場合**
その親オブジェクトのアクセスレベルとその子となるオブジェクトに付与されたアクセスレベルの構成要素からなる集合の*lub*を求め，得られた値を要素とする集合を求める．さらに，その集合に対して*Min*を適用し，その結果として得られたアクセスレベルをその親オブジェクトのアクセスレベルとして更新する．つまり，この親オブジェクトに与えられる更新後のアクセスレベルを$L'(o_p)$，その各子オブジェクトのもつアクセスレベルを

$$L'(o_{c_1}), L'(o_{c_2}), \ldots, L'(o_{c_n})$$

とすると，

$$L'(o_p) = Min(multi(L(o_p), L'(o_{c_1}), L'(o_{c_2}), \ldots, L'(o_{c_n})))$$

となる．ただし，ここで，*multi*は次のように，アクセスレベルの集合の*lub*を要素とする集合を求める演算を意味する．

$$\begin{aligned} multi(L(o_p), L'(o_{c_1}),& \\ L'(o_{c_2}), \ldots, L'(o_{c_n}))& \\ = \{lub(\{L^p, L^{c_1},& \\ L^{c_2}, \ldots, L^{c_n}\})& \\ \mid L^p \lhd L(o_p),& \\ L^{c_1} \lhd L'(o_{c_1}),& \\ L^{c_2} \lhd L'(o_{c_2}),& \\ \ldots,& \\ L^{c_n} \lhd L'(o_{c_n})\}& \end{aligned}$$

(b) **ORエッジの場合**
親オブジェクトo_pの子オブジェクトに付与されたアクセスレベルの和集合とその親オブジェクトのアクセスレベルの*lub*を取ったものとして変更する．つまり，この親オブジェクトo_pに与えられるアクセスレベル$L'(o_p)$は

$$\begin{aligned} L'(o_p) = \{L''_1 + \ldots + L''_n \mid& \\ L''_i \in multi(L(o_p),& \\ L'(o_{c_1}) + \ldots + L'(o_{c_n}))\}& \end{aligned}$$

3.2 例

このアルゴリズムを依存関係グラフを用いて説明する．ただし，依存関係グラフは図3のように既に閉路が除去されて与えられており，各オブジェクトに対してそれぞれ1つずつアクセスレベルが与えられているものとする．ここの例では，図4

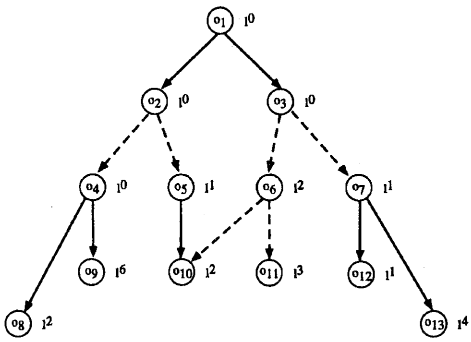

図 3: Before access-levels are changed

のように各アクセスレベル間に半順序が成り立つ場合について考える．

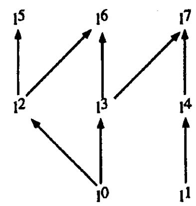

図 4: Partial order access-levels

まず，図3において，ルートとなっているオブジェクトo_1に対してアクセス更新手続きUpdateを適用する．o_1はo_2, o_3を子オブジェクトとして持っているのでo_2, o_3に対してUpdateが再帰的に適用される．まず，o_2にUpdateが適用された場合について考える．o_2の子オブジェクトo_4, o_5にUpdateが適用され，さらにo_4の子オブジェクトo_8, o_9，およびo_5の子オブジェクトo_{10}それぞれに再帰的にUpdateが適用される．子オブジェクトo_8, o_9, o_{10}は図3においてそれぞれ葉となっており，子オブジェクトを持っていないため，更新後のア

クセスレベルをそれぞれ

$$L'(o_8) = L(o_8) = l^2$$
$$L'(o_9) = L(o_9) = l^6$$
$$L'(o_{10}) = L(o_{10}) = l^2$$

に変更する．次にo_4に対して適用されたUpdateに操作が戻る．o_4は子オブジェクトo_8, o_9とANDエッジによって表現される依存関係を持っているので，このo_4のアクセスレベルは

$$\begin{aligned} L'(o_4) &= Min(multi(L(o_4), L'(o_8), L'(o_9))) \\ &= Min(multi(l^0, l^2, l^6)) \\ &= Min(\{lub(\{l^0, l^2, l^6\})\}) \\ &= Min(\{l^6\}) \\ &= l^6 \end{aligned}$$

となる．o_5に適用されたUpdateについても同様に演算すると，

$$\begin{aligned} L'(o_5) &= Min(multi(L(o_5), L'(o_{10}))) \\ &= Min(multi(l^1, l^2)) \\ &= Min(\{lub(\{l^1, l^2\})\}) \\ &= Min(\{l^1 * l^2\}) \\ &= l^1 * l^2 \end{aligned}$$

となる．次にo_2に適用されたUpdateに戻り，o_2のアクセスレベルを演算する．

$$\begin{aligned} multi(L(o_2), L'(o_4) + L'(o_5)) &= multi(l^0, \\ &\qquad l^6 + l^1 * l^2) \\ &= \{lub(\{l^0, l^6\}), lub(\{l^0, l^1 * l^2\})\} \\ &= \{l^6, l^0 * l^1 * l^2\} \\ &= \{l^6, l^1 * l^2\} \end{aligned}$$

なので，それらを＋で結合させて，

$$L'(o_2) = l^6 + l^1 * l^2$$

となる．o_3に適用されたUpdateも同じように考える．まず，葉となるオブジェクトにたどり着くと，そこでo_{11}, o_{12}, o_{13}のアクセスレベルはそれぞれ

$$L'(o_{11}) = L(o_{10}) = l^3$$
$$L'(o_{12}) = L(o_{12}) = l^1$$
$$L'(o_{13}) = L(o_{13}) = l^4$$

と更新される．またオブジェクトo_6については，

$$\begin{aligned} multi(L(o_6), L'(o_{10}) + L'(o_{11})) &= multi(l^2, \\ &\qquad l^2 + l^3) \\ &= \{lub(\{l^2, l^2\}), lub(\{l^2, l^3\})\} \\ &= \{l^2, l^6\} \end{aligned}$$

なので，

$$L'(o_6) = l^2 + l^6$$

オブジェクトo_7のアクセスレベルは

$$\begin{aligned} L'(o_7) &= Min(multi(L(o_7), L'(o_{12}), L'(o_{13}))) \\ &= Min(multi(l^1, l^1, l^4)) \\ &= Min(\{lub(\{l^1, l^1, l^4\})\}) \\ &= Min(\{l^4\}) \\ &= l^4 \end{aligned}$$

となる．そして，オブジェクトo_3については，

$$\begin{aligned} multi(L(o_3), L'(o_6) \cup L'(o_7)) &= multi(l^0, \\ &\qquad l^2 + l^4 + l^6) \\ &= \{lub(\{l^0, l^2\}), lub(\{l^0, l^4\}), lub(\{l^0, l^6\})\} \\ &= \{l^2, l^0 * l^4, l^6\} \end{aligned}$$

なので，

$$L'(o_2) = l^2 + l^0 * l^4 + l^6$$

最後に，根となっているo_1の更新後のアクセスレベル$L'(o_1)$は

$$\begin{aligned} L'(o_1) &= Min(multi(L(o_1), L'(o_2), L'(o_3))) \\ &= Min(multi(l^0, l^6 + l^1 * l^2, \\ &\qquad l^2 + l^0 * l^4 + l^6)) \\ &= Min(\{lub(\{l^0, l^6, l^2\}), \\ &\qquad lub(\{l^0, l^6, l^0 * l^4\}), \\ &\qquad lub(\{l^0, l^6, l^6\}), \\ &\qquad lub(\{l^0, l^1 * l^2, l^2\}), \\ &\qquad lub(\{l^0, l^1 * l^2, l^0 * l^4\}), \\ &\qquad lub(\{l^0, l^1 * l^2, l^6\})\}) \\ &= Min(\{l^6, l^6 * l^4, l^6, \\ &\qquad l^1 * l^2, l^2 * l^4, l^1 * l^6\}) \\ &= l^6 + l^1 * l^2 \end{aligned}$$

となる．このように，依存関係に基づいて更新された各アクセスレベルは図5のようになる．

4 おわりに

本研究では、ユーザが各オブジェクトに与えたアクセスレベルをオブジェクト間の依存関係を考慮することで，実行可能である最低のアクセスレベルを計算し，それらの値を新たなアクセスレベルとして再定義する手法の提案を行なった．

本研究の結果として、利点を挙げると次のものが挙げられる。

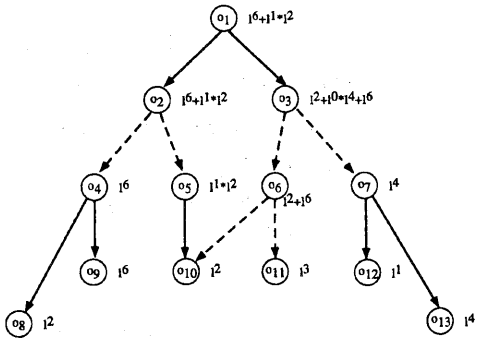

図 5: After access-levels are changed

- 適切なアクセスレベルへの自動修正が可能
- データの参照やメソッドの呼び出しによる無駄なアクセスの軽減

今後の課題として、

- オブジェクトの数が膨大な数になった場合，どのような手法で各オブジェクトに対してアクセスレベルを付与するか
- 実際に様々な例に対しこのアルゴリズムを適用することで，その有効性を実証する．

などが挙げられる。

謝辞

この研究は、一部、文部省科学研究費重点領域研究「高度データベースNo.275」(課題番号08244103)、および日本学術振興会未来開拓学術研究推進事業における研究プロジェクト「マルチメディア・コンテンツの高次処理の研究」による。ここに記して謝意を表す。

参考文献

[1] Rafiul Ahad and James Davis and Stefan Gower. *Supporting Access Control in an Object-Oriented Database Language.* In Proc. of EDBT,Vol.580 of LNCS,pp.184-200,Springer-Verlag,1992

[2] Teresa F. Lunt and Eduardo B. Fernández. *Database Security.* SIGMOD Record,Vol.19,No.4,pp.90-97, Dec.1990

[3] Sushil Jajodia and Ravi Sandhu. *Database Security: Current Status and Key Issues.* SIGMOD Record,Vol.19,No.4,pp.123-126, Dec.1990