3X−6

オブジェクト指向データベースにおける部分スキーマによるアクセス制御

田島 敬史
神戸大学情報知能工学科

1 はじめに

データベースにおけるアクセス管理のための機構として従来から用いられいている物にビューの機構がある。ビューを用いてアクセス管理を行なう場合は、まずデータベースのスキーマ中に、ビューとして仮想的に定義されるデータに関するスキーマを追加し、そして各ユーザ毎にこのビューを含むスキーマの一部分のみを見せる事によってアクセス管理を実現する。これが部分スキーマである。オブジェクト指向データベースの場合、スキーマはクラス階層であり、これに仮想的なデータを定義するクラスを加え、これらのクラスを含むクラス階層の一部分のみを部分スキーマとして各ユーザに見せる事になる。

しかし、オブジェクト指向データベースにおいては各クラス間に参照を通した依存関係が存在するために、単純にクラス階層の一部分を取り出すわけにはいかない。そこで、この論文では、オブジェクト指向データベースにおいて部分スキーマを提供する場合に、必要十分な「部分」を決定する作業を支援する枠組みについて提案する。

2 オブジェクト指向データベースにおける部分スキーマ

オブジェクト指向データベースにおいては、クラス間に、あるクラスの属性が他のクラスのインスタンスを参照する参照や、あるクラスのメソッドが他のクラスのメソッドを参照する参照による、依存関係が存在する。

そのため、オブジェクト指向データベースにおいて部分スキーマを考える場合、部分スキーマ内から部分スキーマ外への参照が問題になる。例えば次のような例を考えてみる。

```
Class Customer = [name:str, debt:int, maxdebt:int];
Method Customer.name():str = return name;
Method Customer.maxdebt():int = return maxdebt;
Method Customer.owe(amount:int):bool =
  if (debt + amount <= maxdebt) then
    debt := debt + amount; return true
  else return false;
Class Account = [customer:Customer, balance:int];
Method Account.draw(amount:int):bool =
  if balance >= amount then
    balance := balance - amount; return true
  else if balance >= 0 then
    if customer.owe(amount - balance) = true then
      balance := balance - amount; return true;
  else if customer.owe(amount) then
    balance := balance - amount; return true;
  return false;
```

クラス Customer は銀行の顧客を表し、属性として、名前 name、所有する各口座の負債額の総計 debt、この顧客に許されている最大負債額 maxdebt を持つ。クラス Account は銀行の口座を表し、所有者である Customer を参照する customer と現在の残高 balance という属性を持つ。この例では、Account の属性 customer が Customer のインスタンスを参照しており、このインスタンスにメッセージを送る形で、Account のメソッド draw が Customer のメソッド owe を参照している。

ここで、あるユーザに対して Customer から属性 debt を取り除いた projection view にあたるクラス Customer2 を以下のように定義したとする。

```
Class Customer2 = [name:str, maxdebt:int];
Method Customer2.name():str = return name;
Method Customer2.maxdebt():int = return maxdebt;
```

そして、あるユーザに対しては Customer2 のみを見せ Customer は隠蔽するとする。この場合、このユーザに対しては、Customer のインスタンスは Customer2 のインスタンスとして見せられるというのが一般的なアプローチである[3]。

しかしその場合、このユーザは Account クラスにアクセスが許されているにもかかわらず、そのメソッド draw はその中でクラス Customer のメソッド owe を参

Access Control using Subschema in Object-Oriented Databases
by Keishi Tajima (tajima@in.kobe-u.ac.jp)
Dept. of Comp. and Sys. Eng., Kobe University,
Rokkodai, Nada, Kobe 657 Japan

照しているため、draw が実行できないことになる。

そこで考えられるのは、このユーザが直接 owe を実行することは許さないが、他のクラスからの参照を通して間接的に owe を実行することは許すという考え方である[1, 2]。しかし、このような間接実行を許すと、ユーザが隠蔽されるべき情報を得てしまう危険性がある。例えば、上の場合、このユーザは Customer2 の maxdebt の値がアクセス可能なので、draw メソッドを使って、いくら以上引き出そうとすると false が返されるか調べれば、現時点での debt の値がわかってしまう。

このように、オブジェクト指向データベースにおいては、クラス間に依存関係があるため、ユーザに部分スキーマを提供する場合には、ユーザに見えているメソッドの実行を保証することと、ユーザが知ってはいけない情報を隠蔽することとの間にトレードオフが生じる。

3　安全な部分スキーマの自動生成

実は上の例では、外部スキーマの基本的な目的が「Account.draw へのアクセスは許すが、Customer.debt へのアクセスは許さない」という物であったとしたら、Customer.maxdebt を外部スキーマから取り除いてやれば、Account.draw の実行を保証しつつ、debt の値は隠蔽することが出来る。このように、外部スキーマに何を含めるかを全て手作業で記述するやり方では、どのメソッドを含めてどのメソッドを含めなければ、目的の外部スキーマを実現できるかを非常に注意深く決めなければならない。

そこで、この研究では外部スキーマの「目的」すなわち、その外部スキーマのユーザに対して特に実行を保証したいメソッドと、逆に隠蔽したい情報に関する記述だけ与えてやり、あとはその条件を満たす部分スキーマを自動生成することを考える。例えば、上の例で考えると、まず以下のような記述を行う。

提供メソッド　＝　{Account.draw, Customer.name}
隠蔽属性　＝　{Customer の debt}

そして以下の外部スキーマの自動生成を考える。

```
Class Customer2 = [name:str];
Method Customer2.name():str = return name;
Class Account = [customer:Customer, balance:int];
Method Account.draw(amount:int):bool =
        ⋮
 return false;
```

単純に考えれば、全てのメソッドの集合の部分集合のうちで、(1) 隠蔽属性と指定された属性の情報を漏洩せず、(2) かつ最大の集合を選べば良い。このうち、与えられたメソッドの集合が、ある属性の情報をユーザに与えうるかどうかを判定する枠組みについては、既にこれまでに提案している[4]。しかし、最大の物は一意に決まるとは限らない。例えば、以下の条件を考える。

提供メソッド　＝　{Customer.name}
隠蔽属性　＝　{Customer の debt}

この場合、外部スキーマとしては、{Customer.name, Customer.maxdebt} と {Customer.name, Account.draw} が考えられる。

このような場合、各候補はその共通部分 (上の例では Customer.name) と、「これらが全て揃うとある属性の情報が漏洩する」という集合 (上の例では {Customer.maxdebt, Account.draw}) からなっている。後者のことを「危険組合わせ」と呼ぶことにする。隠蔽すべき属性が複数ある場合には、危険組み合わせが複数組存在しうる。そこで、候補集合をこれらの集合に分解し、各危険組み合わせを順にユーザに提示し、各組み合わせ中のどれか一つを指定させて、それを外部スキーマから除外していくことで、安全な外部スキーマを容易に作成できるように支援することができるものと考える。

4　まとめ

安全な外部スキーマの生成を支援する枠組みの概略について提案した。今後、各手順の詳細について検討する予定である。例えば、各危険組合わせを提示する時に、もっとも多くの危険組合わせに繰り返し登場するメソッドから順に提示してやることで、より効率よく安全な外部スキーマに到達するようにするなどの工夫が考えられる。

参考文献

[1] Rafiul Ahad, James Davis, and Stefan Gower. Supporting access control in an object-oriented database language. In *Proc. of EDBT*, volume 580 of *LNCS*, pages 184–200. Springer-Verlag, Mar. 1992.

[2] Elisa Bertino. Data hiding and security in object-oriented databases. In *Proc. of IEEE ICDE*, pages 338–347, Feb. 1992.

[3] Elke A. Rundensteiner. Multiview: A methodology for supporting multiple views in object-oriented databases. In *Proc. of VLDB*, pages 187–198, Aug. 1992.

[4] Keishi Tajima. Static detection of security flaws in object-oriented databases. In *Proc. of ACM SIGMOD*, pages 341–352, Jun. 1996.