データに基づくオブジェクト設計：
データ構造と参照関係を用いた責務の配置
Object Design based on Data:
Responsibility Allocation using Data Object and Navigability

佐藤英人
Hideto Sato
東京国際大学 商学部 情報システム学科
Department of Information Systems, Faculty of Commerce, Tokyo International Univ.

要旨
情報システムはデータとプログラムからなる。データについては、正規化設計法があり、誰がやってもほぼ同じ概念設計結果が得られ、データの重複が避けられる。プログラムについては、そのような基準がないために、概念設計結果が設計者によってまちまちになり、プログラムの重複を防止することが難しい。本論文では、正規化されたデータオブジェクト間の参照可能性に基づいて、オブジェクト間に責務を機械的に配分する一つの手順を提案する。

1. はじめに

データベースの概念設計の場合、アプローチとしてトップダウン・ボトムアップのいずれの方法を使ったとしても、誰がやってもほぼ同じ設計結果になる。これは正規形という形式的な判定基準があるからである[1]。

オブジェクトはデータとプログラムで構成される。オブジェクトの責務とは、オブジェクトの果たすべき義務のことであり、オブジェクトがもつメソッド（プログラム）によって実装される。

この責務の配置はオブジェクト指向設計の中心課題であり、その指針の代表的なものとして GRASP パターンがある[2]。これは責務の配置に関する知見をパターンとしてまとめたもので、多くのオブジェクト指向設計がこれに依拠している。しかしながら、この GRASP パターンを構成する各パターンは必ずしも独立ではなく、同一の問題について相反する結果をもたらす場合も少なくない。どこにどのパターンを適用するかは設計者の判断に委ねられており、結果として、設計者の経験や力量によって異なる設計結果が生み出されることになる。

筆者はかつてプロジェクトマネジャークラスの人達の集まりで、図１を示して、「期限を越えて未払いの請求を抱えている顧客は誰か？」という問に答えるメソッドをどこに置くのが良いか尋ねたことがある。その結果、(1)顧客オブジェクト、(2)請求クラス、(3)債務者といった役割オブジェクト、(4)請求ハンドラといった人工オブジェクトという４通りの答が返ってきた。ここではこれらの長所・欠点は論じない。重要なことは、経験を積んだ設計者間でもこのように意見が分かれるということである。

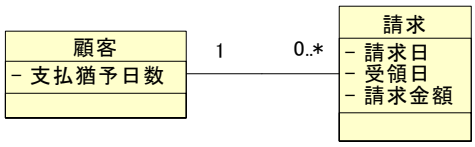

図１ 未払い顧客は誰か？

本論文では、責務の配置を一意に決定できる手順の試案を提示する。これを以下では RADON 法（本論文英文副題の頭文字）と呼ぶ。RADON 法の基本アイデアは１９８８年に発表された堀内の DLCP と同様である[3]。本研究は DLCP の概念用語を見直し、責務一意配置の手順としてまとめたものである。

なお、本研究は、J2EE ライクの４層アーキテクチャを前提とし、そのドメイン層（EJB 層）のオブジェクト（ドメインオブジェクト）の概念設計（実装方式に依存しない設計）を対象としている。プレゼンテーション層オブジェクトや永続化サービスオブジェクトの設計などは対象としていない。

2. 正規化設計の活用

RADON 法は、簡単に言えば、以下の３つの要素で構成される。

(1) クラスの一意な識別
(2) 参照方向の一意な識別
(3) 参照方向に基づく責務の一意な配置

これらのうちのはじめの２つは、データの正規化設計によってもたらされるものである。本章では、

これらについて説明する。

2.1.クラスの一意な識別

対象領域で必要とされるデータ項目が列挙できれば、正規化することで正規形テーブルが得られる。正規形テーブルの各行はオブジェクトの属性値を集めたものであり、これをオブジェクトインスタンスとみることで、クラス（静的モデル）を一意に識別できる。サブクラス関係を含むクラスのデータによる識別については、Smith & Smith の(3, 3) Normal Form を利用することができる[4]。

近年オブジェクト指向分析の世界でも、正規化設計に近い方法が取り入れられてきている。例えば、Ambler は分析クラスの識別に ORM（Object-Role model）を使用しているが、これは Nijssen らが提唱してきた２項モデルと同様であり、そこから導き出される分析クラス図は正規化設計結果と同じである[5]。

2.2. 外部キーによるクラス間リンクと参照方向の一意な識別

正規化設計は、正規形テーブルだけでなく、外部キーによるテーブル間の参照方向も一意に識別する。外部キーによる参照はオブジェクト間の関連の一つの表現である。われわれは外部キーによる参照関係をオブジェクト間のリンク関係と同一視する。

2.3. 外部キー参照と Navigability

オブジェクト指向の世界には、参照関係を表現する Navigability という概念がある。これは「関連先のクラスのオブジェクトまたはオブジェクトの集合を渡り歩ける(traverse)かどうか」を示すもので、「相手が提供するサービスを、他に情報がなくても、利用できる能力」を示している[6]。Navigability は誘導可能性と訳されることが多いが、この直訳は誤解を招きやすいので、ここでは原語を使用する。

われわれは外部キーの参照方向と Navigability を同一視する。外部キーをオブジェクト参照に置き換えて、リンクの意味を考えれば、この同一視は自然である。ただし、この扱いが正しいか否かはここでは論じない。これは仮定である。この仮定を置くことで、Navigability を一意に決定できる。

2.4. 例示

ここで１つ例を示そう。図２は注文受付から配送までを扱う販売システムの例である[7]。

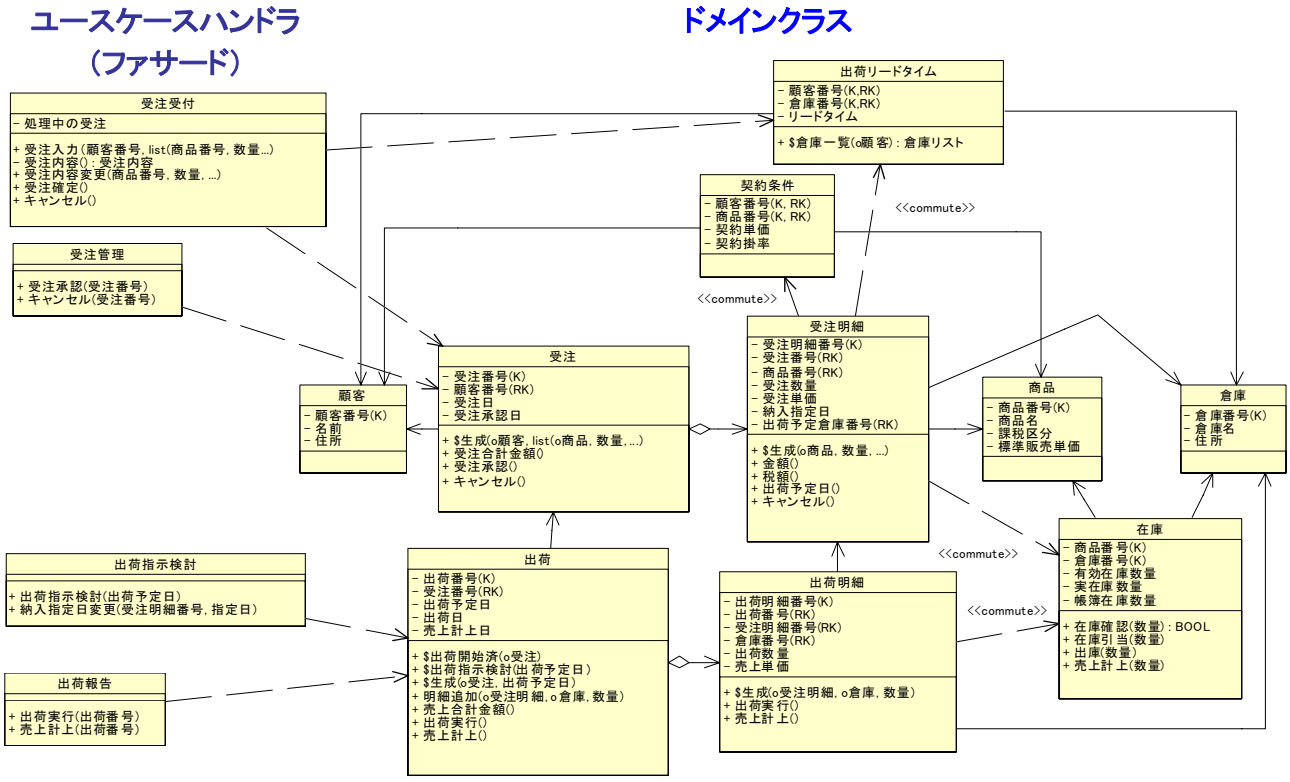

図２　販売システムのクラス図

紙面の都合でこの図は責務の配置が終わった最終的なクラス図を示している。しかし、この図のドメインクラス、その属性、クラス間のリンク線（矢線）は、正規化設計だけを用いて導き出されたものである。属性の脇に書かれた(K)は主キーを、(RK)は外部キーを表している。

受注-受注明細、および、出荷-出荷明細の◇付リンク線は集約関連である。これについては後述する。また、<<commute>>とかかれた破線は、他のリンクから可換性（commutative）を用いて導出できる多対１リンクである。なお、リンクの矢印は外部キーからその参照先に向いている。これは UML における Navigability の記法に合わせたもので、ＤＢ分野の通常の記法とは逆で、「多→１」の向きである。

3. RADON 法の前提

上で述べたように正規化設計によってクラスとクラス間の参照方向を機械的に決定することができる。しかし、それだけでは機械的な責務の配置は実現できない。本章では、責務配置の前提として、われわれが追加した仮定を説明する。

3.1. 関連と参照方向

外部キーによる参照方向は関数従属性（多対１関連）という形式的な判定基準に基づくものである。ここでは、その意味を考察する。外部キーによる参照関係は、関連とその参加者の関係である。外部キーの参照方向に Navigability を限定するということは、「関連オブジェクトは、参加オブジェクトを参照できるが、参加オブジェクトは自分がどの関連オブジェクトに参加しているか知らない」という仮定を置くことに他ならない。

この仮定は、正しいか否かは別として、reasonable である。この仮定に基づいて設計された参加オブジェクトは、関連オブジェクトが存在しない世界でも通用する一般性を持つからである。例えば、受注という関連オブジェクトが存在しない店頭販売の世界でも、顧客や商品というオブジェクトは存在する。

3.2. 集約関連

上述のように Navigability を考えるとき、外部キーによる参照方向と Navigability を同一視することが不適切である関連が存在する。それは集約関連である。外部キーによる参照方向では、集約は「構成要素→集約」になる。他方、集約オブジェクトは構成要素を参加者とする関連であるから、関連は参加者を参照できると仮定すると、「集約→構成要素」でなければならない。

以上の理由から、集約関連は例外として双方向に参照可能であると仮定する。集約関連は他の関連と区別できるので、この例外の導入は参照方向の一意性を阻害しない。

［仮定1］集約関連は双方向参照である。

3.3. 集約関連以外の双方向参照

現実のクラス図では、集約関連以外にも双方向参照が発生する。これは１対１リンクが縮退した結果発生するもので、もとの関連に戻せば、双方向参照をなくすことができる。

3.4. 参照階層

上述のように参照方向を定めたクラス図は、他に依存しないオブジェクト（エンティティ/リソースと呼ばれることがある）から、関連を通じて順次積み上がってきたものになる。従って、参照方向のグラフはクラスをノードとする非循環グラフ(acyclic graph)になる。これを参照階層と呼ぶことにする。

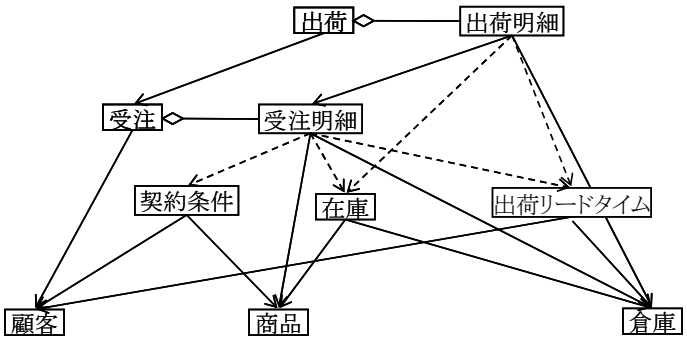

図３　参照階層

集約関連は双方向参照であるので、このグラフは厳密には非循環とはいえないが、双方向参照は局所化されているので、大局的には非循環である。

3.5. クラスに関する仮定

Navigability はインスタンス間の参照関係である。インスタンスの集まりを参照するために、以下の仮定をおく。

［仮定2］クラスはそこに属するすべてのインスタンスを参照できる。

［仮定3］インスタンスは自身が属するクラスを参照できる。

仮定２により、クラスはいつでもある条件を満足するインスタンスの集まりを検索できることになる。

3.6. クラス間の参照方向

クラスはグローバルであり、どこからでも参照できると仮定することが多い。しかし、それでは次に述べる責務の配置に任意性が発生する。そこでわれわれは以下の仮定をおく。

［仮定4］クラス間の参照方向は、属するオブジェクト間の参照方向と同じである。

4. 責務の配置原則

以上の準備のもとに、RADON 法では以下のようにして、責務を機械的に配置する。

(1) 責務毎にその実行に必要となるデータが決まる。このデータの一部を自身の固有データとしてもち、かつ、それ以外のデータすべてを参照できるオブジェクトあるいはクラスに責務を配置する。

(2) 上記条件を満足するオブジェクト/クラスが存在しないときは、必要なデータを参照できる人工クラスを用意し、そこに配置する。

(3) 同じクラスを参照できる人工クラスを１つにまとめることにより、人工クラスの重複を排除する。

これらに加えて、局所的な双方向参照である「集約-構成要素」と「クラス-インスタンス」間の責務の配分原則も必要である。これらは自明に近いものであるので、ここではその詳細を省略する。

図２のクラス図に書かれたメソッドは、ユースケースハンドラが外界から受け取ったイベントの処理を上記基準に従って分解・配置していったものである。ここでは、人工クラスを必要としていない。

5. むすび

本論文では、データベース分野の正規化設計をベースに４つの仮定を追加することで、責務の配置を機械的に行えることを示した。４章で示した責務配置原則の(1)は GRASP の Expert パターン、(2)は Controller パターンに相当する。RADON 法の GRASP との大きな違いは以下の２点である。

(1) 事前に Navigability を決定している。

(2) クラスの責務として「インスタンスの検索」を考慮している。

(2)により、オブジェクト間の複雑な依存関係を排除し、(1)により、責務の一意配置を可能にしている。

RADON 法は責務の一意配置を目的とし、他の評価軸を考慮していない。しかし、その結果は図２をみていただくと分かるように、比較的疎結合かつ高凝集なクラス群が設計できている。少なくとも、概念設計の初期モデルとして十分使用に耐えられるものと思われる。ただし、機械的配置は時として人間にとって理解しにくい結果を生み出す。例えば「出荷済なら受注はキャンセルできない」という条件付の受注キャンセルは、RADON 法では受注オブジェクトではなく出荷オブジェクトの責務になる。恣意性を排除しつつ、人間にとっても理解しやすい配置を実現する方法は今後の課題として残されている。

RADON 法は一例に過ぎないが、このような恣意性の少ない設計手順をベースに個別事情に基づく改良パターンを議論していくことが、ソフトウェアを資産として蓄積していく上で重要であると思われる。RADON 法がそのようなアプローチへの一石となれば幸いである。

参考文献

[1] Fleming, C.C. and Halle, B., *Handbook of Relational Database Design*, Addison-Wesley, 1989.

[2] Larman, C., *Applying UML and patterns: an introduction to object-oriented analysis and design* (依田光江訳、実践 UML: パターンによるオブジェクト指向開発ガイド、プレンティスホール、1998）, Prentice Hall, 1998.

[3] 堀内一, *データ中心システム設計*, オーム社, 1988.

[4] Smith, J.M. and Smith, D.C.P., "Database Abstractions: Aggregation and Generalization", *TODS Volume*.2 *No*.2, June 1977, pp.105-133.

[5] Ambler, S.W., *Object Primer, 3rd edition: Agile Model-Driven Development With UML 2.0* (越智典子訳 : オブジェクト開発の神髄、日経 BP 社、2005), Cambridge University Press, 2004.

[6] Rumbaugh, J., Jacobson, I. and Booch, G., *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1999.

[7] CBOP フレームワーク開発部会, “BO 基本パターンを用いたフレームワークの開発 -販売システムを例として-“、*１９９９年度 CBOP 定例総会報告*, ビジネスオブジェクト推進協議会, 1999.4.12.