エンティティの状態の正規化に関する考察
Considering entity states normalization

ナヌ　アレキシ† 　　佐藤　英人‡
Nanou Alexis† 　　Hideto Sato‡
† 東京国際大学大学院　商学研究科（2年）
‡ 東京国際大学　商学部
† Graduate School of Business and Commerce, Tokyo International Univ.
‡ Faculty of Commerce, Tokyo International Univ.

要旨
業務システムは大きく分けるとビジネスロジックとデータから成り立っている。データについては正規化の概念があり、属人性のない分析結果を得ることができる。他方、ビジネスロジックについては、客観的な方法がなく、分析結果は人によって異なる結果となる。そこで本論文ではビジネスロジック記述の基礎として、エンティティの状態図に焦点を当て、誰が描いても同じ結果に至る方法を考察した。

1. はじめに

これまで業務システムの上流分析工程では、エンティティとプロセスの整合性をチェックする手段として、CRUD 図が広く使われてきた。しかし、CRUD 図は厳密ではない。例えば、U(Update)と書いてあっても、エンティティの何がどのように更新されるかは分からない。そこで、より厳密な仕様記述としてエンティティの状態図が使われるようになってきている[1]。しかし、状態図の書き方には属人性があり、描く人によって異なる場合が少なくない。

ビデオレンタル業務を例にみてみよう。そこには図１のように会員とテープの間に貸出という関連がある。このとき貸出業務への参加者であるエンティティの状態として、図２のような状態図を書くこともできるし、図３のように貸出という関連の状態図で記述することもできる[2]。

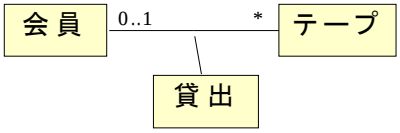

図１　ビデオレンタルのモデル

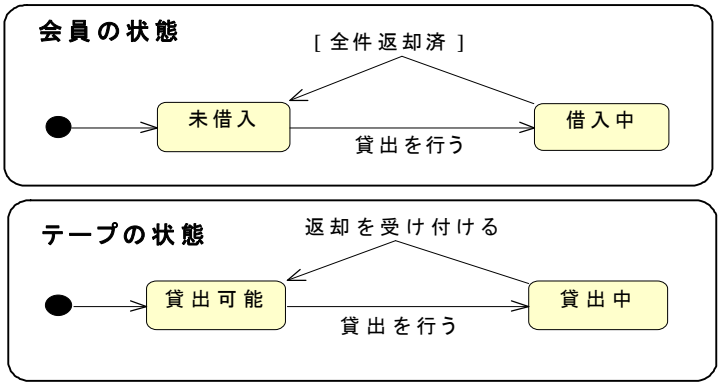

図２　関連参加者の状態図

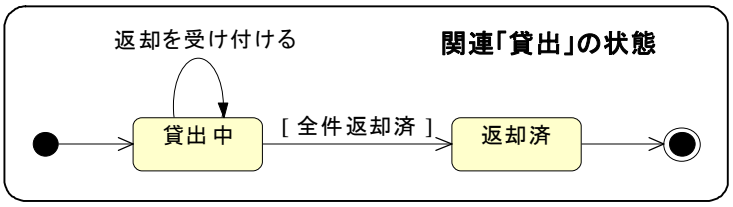

図３　関連の状態図

このように複数の状態図の書き方が可能である背景には、他のエンティティの状態から演繹できるような派生状態があるからである。これは正規化する前のデータとよく似ている。データの正規化では「テープＡを借りている顧客の住所はＢである」というデータを「テープＡを顧客Ｃが借りている」「顧客Ｃの住所はＢである」という２つの独立なデータに分解することにより、データの重複記述を排除する。これと同様に、状態をエンティティに固有な状態と派生状態に分けることによって、状態記述の重複を排除し、状態のとらえ方を属人性のないものにできるのではないかと考えられる。

2. 状態の概念

2.1. 状態の定義

状態の定義には、イベント（事象）をもとに定義する方法と属性値やリンクの存在をもとに定義する方法の２通りのものがある。

前者は有限状態マシンの考え方によるものである。それは「有限状態マシンというのは、仮定の機構であり、いろいろな状態をとりうる中で、一時点ではそのうちのある１個の状態をとるというものである。ある事象が生ずると、この状態の切り換えが起こる。ある１つの処理というものは、有限状態マシンをいくつか集めた形で表すことができる。これを利用して複雑な処理を精密に書いたり概念として表現したり、可能な状態の推移がすべて考え尽されたかどうかをチェックしたりすることができる」[3]

これは与えられた要求仕様に整合的な一組の状態図の満たすべき条件を述べているが、そのような状態図が１つに限られるという保証を与えるものではない。

いまひとつの定義は、「状態とは、１つのオブジェクトの属性値および他のオブジェクトとのリンクを抽象化したものである」[4]というものである。同様の属性値に基づく状態の定義は、データベース分野で広く用いられている。

この属性値やリンクの存在をもとにする状態定義は、属人性を排して状態を定義しようというわれわれの目的に適っている。正規化されたデータ構造（エンティティ）を前提とし、外部キーの存在をリンクと同一視することで、状態定義の恣意性を排除することができるからである。以降これを「エンティティのデータに基づく状態」と呼ぶ。このデータに基づく状態は、状態フラグのような人為的な属性によって識別される状態は含まないものとする。

2.2. 把握するべき状態

データに基づく状態では、属性値の組み合わせ一つ一つを異なる状態と考えると無数の状態がありうることになる。アプリケーションの挙動を記述する上で必要な状態は、そのうちの一部に過ぎない。これを把握するには、イベントに基づく状態定義を併用する必要がある。

ユースケースは、システムの外界とのインタラクションの集まりで意味がある最小のものである[5]。同様のものとして、IE の基本プロセスの概念がある[6]。われわれはこのユースケース（あるいは基本プロセス）を一組のイベントと考え、その事前条件と事後条件を記述する上で必要最小限の状態を把握すべき状態と考えることにする。

2.3. 固有状態と派生状態

この把握すべき状態として識別された状態は、必ずしも上で述べたデータに基づく状態として解釈できるとは限らない。関連先のエンティティの状態から演繹されるような状態もあるからである。

そこで把握すべき状態のうち、当該エンティティの属性値や外部キーで表現できる状態をそのエンティティの「固有状態」と呼び、関連先のエンティティの固有状態から演繹される状態を「派生状態」と呼ぶことにする。

3. 状態の正規化：事例による検討

3.1. 前提とする情報

上で述べた固有状態、派生状態の概念を用いて、与えられた外部仕様に対応するエンティティの状態図を導出する手順をビデオレンタル業務を例として述べる。ここでは、要求仕様に登場するデータ項目の正規化を通じて、図４のエンティティ関連図が事前に得られているものとする。

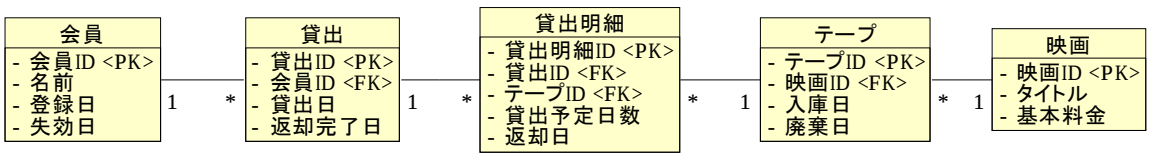

図４ 前提とするエンティティ関連図

また、このアプリケーションには多くのユースケースが含まれているが、ここでは紙面の都合上、「貸出を行う」「返却を受け付ける」という２つのユースケースを例として取り上げる。この２つのユースケースの事前条件、事後条件は、表１の通りである。

ユースケース	貸し出しを行う
事前条件	借り手の会員は有効会員でなければならない。
	会員は返却を延滞しているテープが3本以上ない。
	貸出希望のテープは貸出待ちの状態である。
事後条件	貸出エンティティが生成されその状態は契約中である。
	貸出テープの状態は貸出中である。
プロセス	返却を受け付ける
事前条件	返却テープの状態は貸出中である。
事後条件	返却されたテープの状態は貸出待ち
	未返却テープがあるときは契約エンティティの状態は契約中である
	最後の一本が返される時は契約エンティティの状態は契約完了である。

表１ユースケースの事前条件と事後条件

会員	有効
	無効
	3本以上の延滞あり
	3本以上の延滞なし
テープ	貸出待ち
	貸出中
貸出	契約中
	契約完了
	未返却テープあり
	未返却テープなし

表２　抽出された状態

3.2. ユースケースから抽出される状態

表１の事前条件と事後条件から、表２にある状態が把握すべき状態として抽出される。

3.3. 固有状態と派生状態の識別

上で抽出された状態の１つ１つについて、エンティティのもつ固有属性の組合せで記述できるかどうかを検討する。例えば、会員の有効、無効の状態は、図５の会員の状態のように、登録日と失効日の組合せで表現できる。

他方、テープが貸出待ちであるか否かはテープがもつ属性では表現できない。テープが参加する貸出明細の状態を見ることで、テープが貸出待ちであるか否かを決定できる。すなわち、テープの貸出待ち、貸出中の状態は、表３にあるような派生状態である。

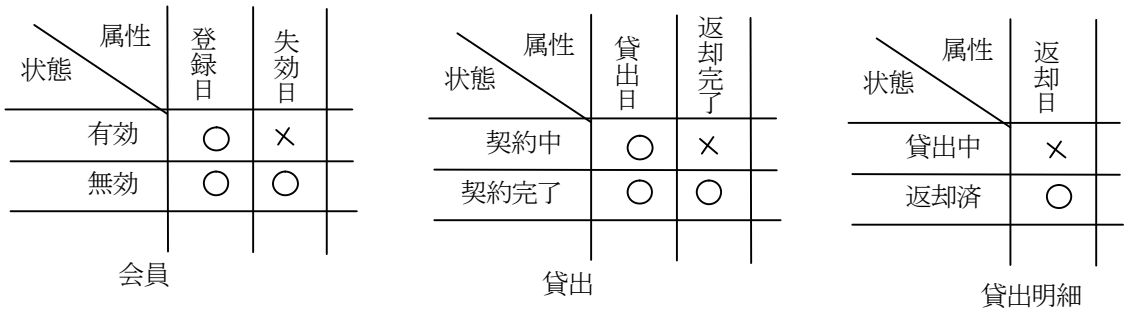

図５ エンティティの固有状態

エンティティ	状態	導出方法
会員	3本以上延滞あり	会員が参加する貸出がもつ貸出明細で延滞中のものが3以上
	3本以上延滞なし	会員が参加する貸出がもつ貸出明細で延滞中のものが3未満
貸出明細	延滞中	貸出中で、本日-貸出．貸出日-貸出予定日数＞0
テープ	貸出待ち	参加貸出明細がないか、または、参加貸出明細が貸出中でない
	貸出中	参加貸出明細が貸出中である

表３ エンティティの派生状態

以上の分析結果の要点をまとめると図６のようになる。

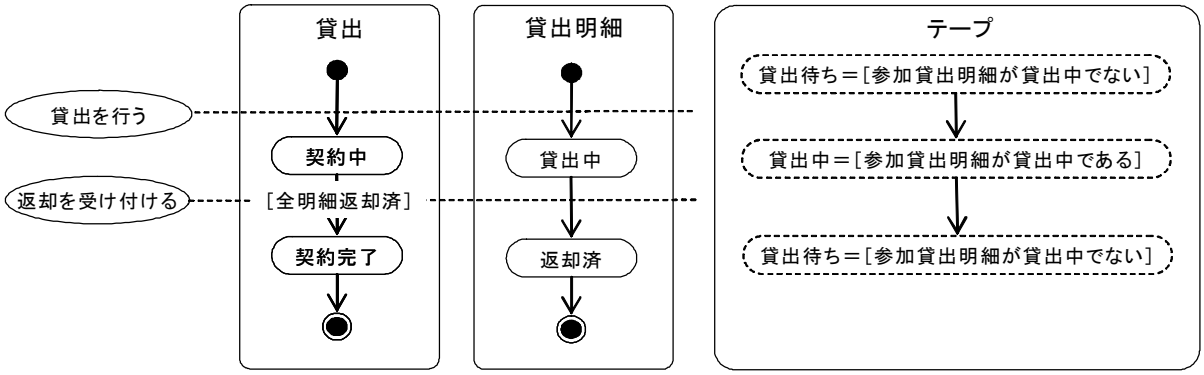

図６ ユースケースとエンティティの状態遷移

3.4. 残された課題

以上の手順を踏むことで、与えられたデータ構造とユースケースの事前・事後条件から、半ば機械的に状態図を導き出すことができる。しかし、いくつか問題点あるいは判断を迷う点が残されている。

１つは固有状態と派生状態の切り分けについてである。上の例では、貸出の状態「契約完了」を固有状態としている。これは返却完了日という固有属性があるからである。しかし、この属性がなくても「その貸出に属する貸出明細のすべてが返却済」という派生状態として「契約完了」を定義することもできる。このように返却完了日という属性がない場合は、「契約中」という状態も、固有属性のみでは判定できなくなり、「契約済で契約完了前」といった派生状態として定義する必要がでてくる。

現実には例え計算で求まる属性でも、契約完了日のような重要な属性は、固有属性として記述する場合が多いであろう。どこまで固有属性として考えるべきか、判断を迷う例である。

上述のこととも関連するが、状態図に派生状態を含むべきか否かという問題もある。状態図を書く目的は、与えられたユースケースのもとでシステムの取り得る状態が尽くされているか否かをチェックすることである。システムの取り得る状態は永続化されるエンティティの固有属性の組合せに外ならないので、固有状態のみの状態図で十分であると考えることができる。しかし、固有状態のみの状態図では、ユースケースの事前・事後状態を直接読み取ることはできず、トレーサビリティが低くなる。状態図をどのように書くのが良いか、今後更なる検討が必要である。

4. まとめ

本稿では、エンティティの固有状態と派生状態の概念を導入し、分析者の恣意性に依存しないエンティティ状態図の導出手順を説明した。前提として正規化されたデータ構造が与えられているという仮定をおいているが、現実に多くの業務システム開発でプロセスとは独立にデータ分析が行われており、強すぎる仮定とはいえないであろう。今後、この手順をより大規模な業務システムに適用し、その有用性を確認していきたいと考えている。

なお、本研究は DOODA（データ指向とオブジェクト指向の融合化研究）プロジェクトでの議論をもとに、筆者等の考えを加えてまとめたものである。関係者各位に御礼申し上げます。

参考文献

[1] 石井孝, 仲谷元 , ３層クライアント/サーバ設計技法, 共立出版, 2000.
[2] 佐藤英人, "JDMF/M-92 の振る舞いモデル機能拡張", データベースシステム 103-6, 情報処理学会, 1995, pp.41-48.
[3] J. Martin, ソフトウエア構造化技法, 近代科学社, 2001.
[4] J. Rumbaugh et al, Object-Oriented Modeling and Design, Prentice-Hall, 1991.
[5] J. Rumbaugh, I.Jacobson, G.Booch: The Unified Modeling Language Reference Manual, Addison-Wesley, 1999.
[6] Texas Instruments, A Guide to Information Engineering Using the IEF , Texas Instruments, 1990.