オブジェクトのライフタイムに基づくクラス図の設計

Design of Class Diagram based on Object Lifetime

金田 重郎† 井田 明男†
Shigeo Kaneda† Akio Ida†
†同志社大学大学院・理工学研究科・情報工学専攻
† The Graduate School of Science and Engineering, Doshisha University

要旨
クラス図の関連を RDB の関数従属性と同等な関数関係と仮定し，オブジェクト間の関係をオブジェクトのライムタイムの視点から分析した．その結果，関数従属的な関連は，あるオブジェクトから，当該オブジェクトを包含するライフタイムを有する(ライフタイムがより長い)オブジェクトに向かう場合に限定されることが示された．この場合，オブジェクトにタイムスタンプ属性や有効期間を示す属性を導入すれば，少なくともライフタイムが包含関係にある場合には，関連の一方の多重度が 1（そして 1 に限る）の「関数従属的な関連」に変換できる．そして，それ以外の関連は意味的に存在することは少ないと推定される．本分析から，クラス図設計におけるライフタイムの重要性が確認されるとともに，関連の片方に，多重度 1（そして 1 に限る）を用いている，存在従属クラス図，及び，渡辺幸三によるモデリング手法の正当性・汎用性が傍証される．

1. はじめに

UML 静的モデルのクラス図（図 1）では，「関連」が重要な要素として登場する．「関連(Association)」とは何であろうか．UML の仕様では，「意味的な関係がある場所に引く」と言った表現で定義されている[1]．しかし，これでは，エンジニアは自信をもって関連を引くことはできない．そこで，本稿では，「関連(Association)」を「こちらのインスタンスが決まれば，関連先のインスタンスが一意に決まる」RDB の関数従属と等価な関数関係として，関連と仮定する．そこでは，非推移的関数従属であることも併せて含意する．

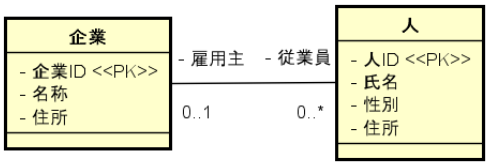

図 1　クラス図の例

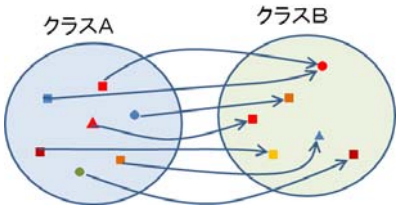

図 2 関数従属性

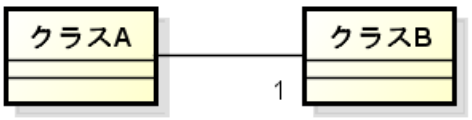

図 3　関数従属的なクラス間の関連

上記条件のもとで，関連で接続された 2 つのインスタンスのライフタイムの関係に注目する．その分析から，ライフタイムの 2 者関係は，「同時」，「すれ違い」，「包含」，「非会合」の 4 通りしかなく，1) 関数従属的な関連は，一方のライフタイムが，他方を包含する場合にのみ限定されること，2) 属性としてタイムスタンプや期間属性（開始タイミングと終了タイミング）をクラスに与えることにより，通常の関連は，片方の多重度が「1（かつ，その時に限る)」を容易に実現できること，を示す．この結果は，井田による存在従属クラス図[2][3]，あるいは，渡辺幸三のデータモデル[4][5][6]が，特定業務だけに適用されるものではなく，広く汎用性を持つことを傍証する[1]．

2. 関数従属性としての関連（Association）

2.1. 関数従属的な関連

図 2 には，ＲＤＢの関数従属性の概念イメージを示す．クラスＡのインスタンスが一つ決まれば，クラスＢのイ

[1] 本稿の内容は，文献[7][8]の発表内容を当該文献の著者自身がまとめ直したものである．ただし，文献[7][8]の発行元の著作権に配慮し，図および記述文は，すべて全面的に書き改められている．

ンスタンスがユニークに決まる．ただし，クラスBの同一インスタンスが，複数のクラス A のインスタンスに関数従属することはある．図 2 はオブジェクト図とも読み取れる．クラス図化した場合には，図 3 となる．なお，本稿では，多対多の関連は扱わない．図 3 では，関数の出力として指定される側のクラスの多重度は「1(でかつそれに限る)」である．　他方の多重度は，例えば，「0..*」「1..*」等がある．

2.2. ライフサイクルタイムから見たインスタンス間の関係性

ここで，2 つのインスタンスがそれぞれ持っているライフサイクルの関係に着目する．2 つのインスタンス間のライフサイクルの関係は 4 通りしかない(図 4)．関連が引き得るのは，2 つのインスタンスが同一タイミングでこの世に存在している期間（一部あるいは全部）である．以下，順番に説明する．

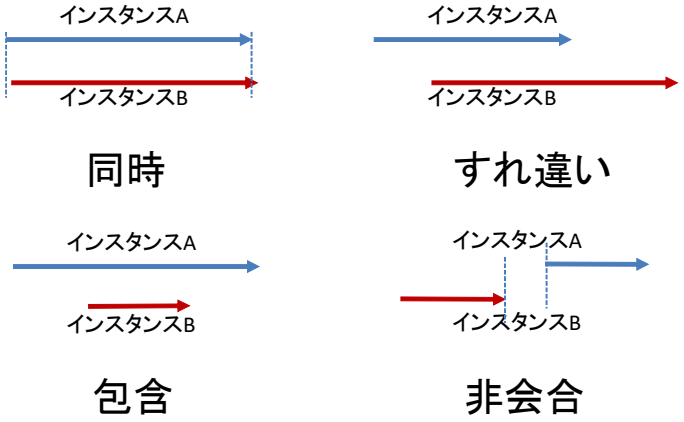

図 4　インスタンス間の関係

【同時】
2 つのインスタンスの生成と消滅が全く同一の場合である．本来は，2 つのインスタンスは一つにまとめるべきであろう．Has-a のコンポジションの可能性もある．尚，コンポジションでは，片方のクラスの多重度は 1 であり，関数従属的な関連である．

【すれ違い】
2 つのインスタンスが同時に存在する期間はあるが，それぞれが単独で存在する期間もある．この場合には，片方の多重度を 1 限定にすることは難しい．ただし，すれ違いは，2 つのクラスに関連性が薄いことを含意する．この場合に関連が引かれるか否かは疑わしい．

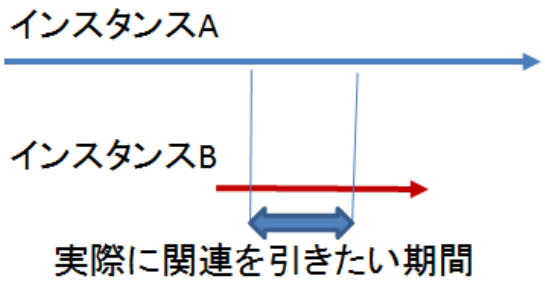

図 5　関連を引きたい期間

【非会合】
2 つのインスタンスが，この世に同時には存在し得ないケースである．関連など引けるはずもない．

【包含】
片方の（ライフサイクルの長い）インスタンスのライフサイクルが，他方のライフサイクルを完全に包含するケースである．では，この場合，どちら側のインスタンスが決まれば，他方が決まるのであろうか．回答は，明らかに「インスタンス B が決まれば，インスタンス A が決まる」である．なぜなら，もし，「インスタンス A が決まれば，インスタンス B が決まる」となると，関数入力側のインスタンス A が存在するにもかかわらず，関数値に相当するインスタンス B が存在しない期間があり不合理であるっためである．

2.3. タイムスタンプ導入による関数従属性の実現

ただし，図 4 の「包含」においても，図 3 に示す様な，多重度=1（そして，それに限る）とはならないケースが想定さえる．「関連」は，インスタンス A とインスタンス B の双方がこの世に存在する期間の一部に対して張られる可能性があるからである．その様子を図 5 に示す．ここでは，特定の期間のみ関連が引かれるので，インスタンス A 側の多重度は「0..1」である．このまま無理に多重度を「1」とするには，図 5 の例では，インスタンス B が出現してから，インスタンス A とともにインスタンス B が消えるまで，常に関連が張られている必要がある．

図 5 に示す様に，インスタンス A とインスタンス B の間に関連が張られるのは，一定の期間のみと思われる．た

だし，実際には，その期間には，極端に短く一瞬のタイミングの場合も含むと思われる．従って，次の 2 つの対策により，インスタンス A 側の多重度を常に「1」としながら，実質的に，インスタンス A とインスタンス B の間の関連を特定期間のみ有効化できる．これらの手法は，現場でプログラマによって従来から使われている手法（小細工気分の多少のうしろめたさを伴って）であるが，以上の論議から，単なる便法ではなく，技術的な必然性がある．

【タイムスタンプの導入】（図 6）

「もの―こと―もの」パタンに現れる，イベント型のインスタンスについて考える．イベント型の例としては，「発注（する）」「受注（する）」などがある．イベント型では，イベントの発生した時点の情報を保持している傾向が強い．最近はメモリが大容量化し，HDD も安価となっているので，一度作成したインスタンスを無理に消去する必要はない．

【有効期間を表示する属性の導入】（図 7）

関連が一定の期間を持つ場合には，有効期限の最初と最後を表現する属性を導入すれば，インスタンス A 側の多重度を常に「1」としながら，実質的に，インスタンス A とインスタンス B の間の関連を特定期間のみ有効化できる．有効期間が開始されたが，まだ，終了していない場合には，終了を表す属性は NULL 値となる．

3. 存在従属クラス図との関係性

井田[2]は，ER 図の Chen が提案した「存在従属[3]」をクラス図に取り入れることの重要性を指摘している．存在従属クラス図では，クラスは「独立クラス」と「従属クラス」に分かれている．独立クラスとは，他のクラスのインスタンスの存在に無関係に，当該クラスのインスタンスを生成可能なクラスである．従属クラスのインスタンスは，別のクラス（一個とは限らない）のインスタンスが存在していないと生成できない．例えば，商品を製造メーカに発注する場合を考えよう．この場合，「商品（種別を表す）」「製造メーカ」そして「発注」の 3 つのクラスが存在する（図 8）．商品と製造メーカが独立クラス，発注は従属クラスである．商品のインスタンスと，製造メーカのインスタンスが指定されないと，発注クラスのインスタンスは生成できない．

図 8 からもわかる様に，独立クラスには，オブジェクト ID がプライマリーキーとして定義されている．しかし，従属クラスでは，このクラスのインスタンス

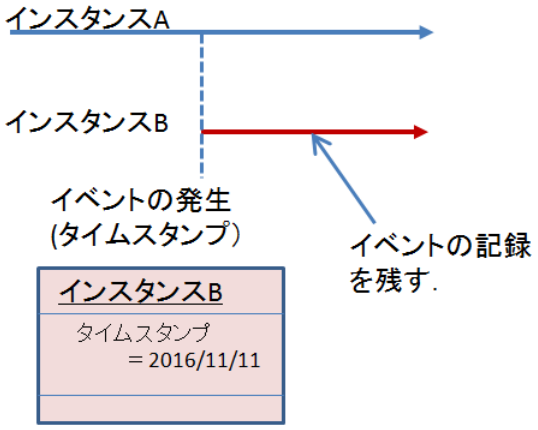

図 6 タイムスタンプの導入

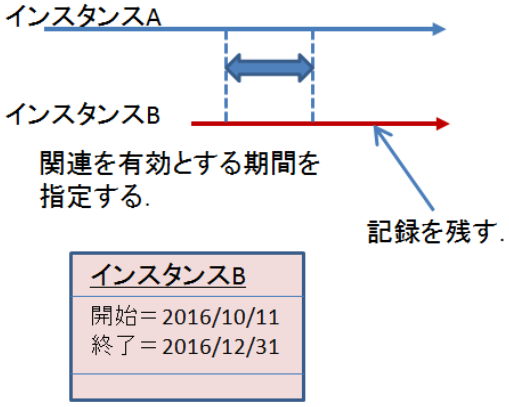

図 7　有効期間を表す属性の導入

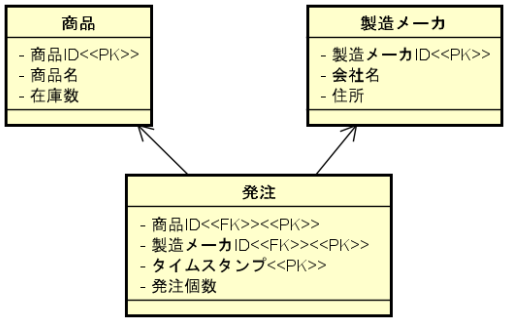

図 8　存在従属クラス図の例 1

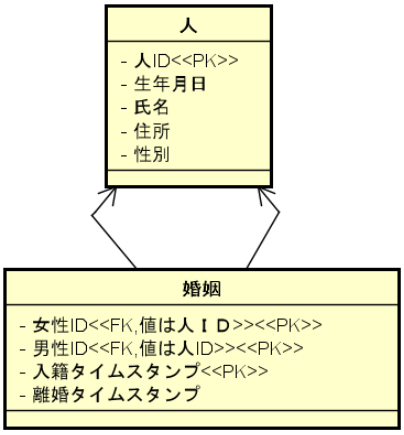

図 9　存在従属クラス図の例 2

が存在を前提とするインスタンスのオブジェクト ID をプライマリーキーとして利用できる．ただし，同じ商品と製造メーカとの組み合わせでも，何度も発注は行われるから，ここでは，タイムスタンプを用いて，（同一の商品と製造メーカを有する）発注インスタンスを区別している．

関連には，UML の誘導可能性の表示を流用して，矢印を与えている．鏃（やじり）の側が上流である．上流側のクラスのインスタンスが存在しなければ，下流側のクラスのインスタンスは存在し得ない．矢印の先の多重度は，「1（そして，1 に限る)」である．すなわち，存在従属の関連は，関数従属的な関連に他ならない．関数従属になっているのは，図 6 に示した，タイムスタンプが導入されているためである．

同じく図 9 は，別の存在従属クラス図の例である．ここでは，女性と男性を含む「人」が独立クラスであり，この両者の間に婚姻関係を定義している．婚姻インスタンスの有効期間は，その開始が「入籍タイムスタンプ」により表現される（これをプライマリーキーに含めている)．一方，万が一離婚すれば離婚タイムスタンプ（単なる属性）が書き込まれる．結果として，図 9 の上流側クラスの多重度は「1（そして 1 に限る)」であり，関連は，関数従属性的な関連（関数）となる．

前述（2.2 節）の議論から，従来のクラス図のすべての関連を存在従属関連のみで表現できるとは言えない．しかし，「非会合」のケースでは，関連が引かれるはずもなく，「すれ違い」のケースでは，意味的な関連の弱さが示唆されており，関連が必要となるチャンスは少ないと推定される．ライフタイムが同一の「同時」では，Has-a のコンポジションの場合と，本来は一つのクラスで表現するべきケースが存在する．したがって，現実の関連のほとんどは，図 8～9 に例示した，存在従属の関連で表現できると思われる．言い換えると，井田の存在従属クラス図は，どのような分野のモデリングにも適用できる広い汎用性を有している．

尚，渡辺幸三は，ER 図ではあるが，片方の多重度が「1（に限る)」モデリング手法を提案している．以上の議論から，渡辺らのモデリング手法も，業務システムなど一部のドメインにしか適用できない特殊な手法ではなく，広く適用できる汎用性を持っていると傍証される．

4. まとめ

2 つのクラス間の関連を RDB の関数従属性と同等の「片方のクラスのインスタンスが決まれば，他方のクラスのインスタンスがユニークに指定される関数関係」とした際のクラス図について分析した．関連が張られる 2 つのインスタンス間のライフタイム分析から，大半の関連を関数従属的な関連で表現でき，その際には，プログラマが多用する，タイムスタンプや，有効期間を示す属性が重要なことが示された．関連に関する本稿の理解は，クラス図作成における，関連作成のガイドラインであり，併せて，存在従属クラス図，および渡辺幸三のデータモデリング手法の汎用性を傍証している．

参考文献

[1] OMG Management Group, “OMG Unified Modeling Language™ (OMG UML), Superstructure Version 2.4.1”, p.36, OMG Management Group (2011)
[2] 井田明男，金田重郎，熊谷聡志，藤本明莉，“存在従属性に着目した論理要件ロバストなドメインモデルの作成 -ドメインクラス図をユビキタス言語として用いるために-”，情報処理学会論文誌，Vol.56，No.5，pp.1340-1350，(2015)
[3] P.Chen, “The Entity Relationship Approach to logical Database Design, QED, Information Science, Wellesley, (1979)
[4] 椿正明, “名人椿正明が教えるデータモデリングの技”，翔泳社（2005）
[5] 渡辺幸三, “販売管理で学ぶモデリング講座”，翔泳社（2008）
[6] 渡辺幸三, “業務別データベース設計のためのデータモデリング入門”，日本実業出版社（2001)
[7] 金田重郎・井田明男，“オブジェクトのライフタイムに基づくクラス図（ER 図）の理解”，電子情報通信学会・知能ソフトウェア研究会，2016 年 11 月
[8] 金田重郎・井田明男，“オブジェクトのライフタイムに基づくクラス図の理解”，情報処理学会・データベースシステム研究会，2016 年 9 月
[9] 金田重郎，井田明男，酒井孝真，熊谷聡志, “日本語仕様文からの概念モデリングガイドラインー行為文と関数従属性に基づくクラス図の作成ー”，電子情報通信学会論文誌 D，Vol.J98-D，No.7，pp.1068-1082, (2015)