解　説

3. オブジェクト指向データベースの技術的諸問題

3.3 オブジェクト指向データベース設計†

酒　井　博　敬††

1. はじめに

オブジェクト指向データベースないしデータベースのオブジェクト指向化は，知識ベース，マルチメディアデータ，エンジニアリングデータなどいわゆるデータベース技術の高度応用，および企業を中心とする情報システムの情報資源管理指向に支えられて急速に発展しつつある．

本来データベースはデータという資源を独立体として長期的に管理し，多様な目的のために共有するという考えから発展したもので，実現技術の観点からは持続性と共有性が核となる．オブジェクト指向データベース（以後 OODB と略す）では，「データ」が「オブジェクト」という資源に代わる．すなわち OODB には，多様な応用に共有され得るオブジェクト資源が維持されていることが必要条件となる．

オブジェクトは実体と実体に固有の操作を一体化したものである．**図-1** は従来のデータベース（a）と OODB（b）を対比したものである．（a）において「受注」と「在庫」は応用プログラムであり，「得意先」と「製品」はデータベースである．データベースに対して「売掛」，「引当」，「入庫」処理を行う操作モジュールは応用プログラムの中に組み込まれている．一方（b）では，これらのモジュールはそれぞれ「得意先」，「製品」に固有の操作として組み込まれ，オブジェクトを形成している．また「受注」，「在庫」はオブジェクトに適切な順序でメッセージを送ることによって，それらのもつ固有の操作を起動する機能をもったオブジェクトである．「受注」あるいは「在庫」と，「得意先」あるいは「製品」との関係は，（a）のようなプログラム対データではなく，利用するオブジェクト対利用されるオブジェクトの関係にある．そして多様な応用に共有されるオブジェクトの集まりが OODB を形成する．

さらに OODB システムの支援すべき機能的な要件には，複合オブジェクト，オブジェクトアイデンティティ，カプセル化などの実現，型またはクラスの定義，クラスの性質の継承などが含まれてくる[1),11)]．これらはオブジェクト資源の再利用性，拡張性を高めるための基本手段でもある．以上のことから，OODB の基本的な目標はオブジェクトの共有性，再利用性，拡張性，一貫性，および持続性の実現にあるといえる[3)]．

一方 OODB の設計は，たとえば企業環境におけるすべてのオブジェクトを識別し，相互に関連付け，データベースに編成し，外部とのインタフェースを設定するという意味で，情報システムの構築法にかかわってくる．この分野では伝統的な

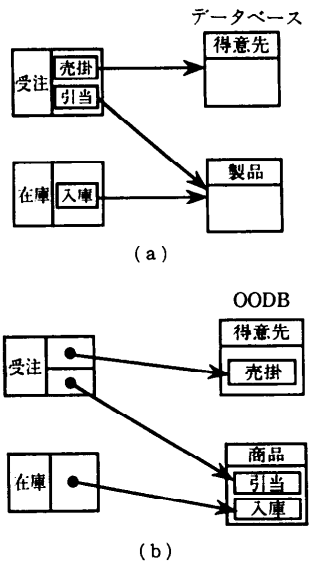

図-1 データベースと OODB

† Object Oriented Database Design by Hirotaka SAKAI (Department of Industrial and Systems Engineering, Chuo University).

†† 中央大学理工学部管理工学科

機能中心アプローチに代わって，データ中心アプローチがとりあげられるようになり，わが国でも多くの企業がこの考え方に沿って情報資源の整備を進めている．OODB 設計はこの構築法をさらに体系化するものと考えられる[18),23)]．

OODB 設計方法論は未開拓の分野であり，現状ではその体系的な解説は困難である．本稿では次のような問題を考慮しつつ，OODB の概念設計上基本的と思われる考え方についてまとめてみたい．

◆ オブジェクトとは何か，また何をオブジェクトとすべきか．

◆ オブジェクトの関連性とは何か．

◆ オブジェクトの振舞いおよび一貫性とは何か．

2. オブジェクトの設計とは何か

オブジェクトは，実世界に存在する実体のもつ構造，操作，および一貫性制約に関する諸性質が凝集された概念 (a coherent concept of the world[6)]) である．オブジェクトとは何か，また何をオブジェクトとすべきかを考える際，「オブジェクトはある応用環境において活動する実体の表現である」ことを認識してかかる必要がある．すなわち次のことが設計の基本となる．

◆ 実体を定義したとき，その実体はどのように振る舞うのか，すなわちどのような固有の操作を提供するのかを考えること．

◆ 一方応用における機能を定義したとき，その機能を実現するための操作を提供する実体は何かを考えること．

そのためには，実体の静的，動的な性質，およびそれらに関する一貫性制約を併せて認識しなければならない．これは実世界に関するオールラウンドな知識と問題の直観的な把握を要請する．それゆえに OODB 設計においては，適切な要求分析，定義技法が従来のデータベース設計に比べて一層重要になる．たとえば RML (Requirements Modeling Language)[6)] や Telos[8)] は先駆的な役割を果たす言語といえよう．

このような前提のもとで，OODB 設計手順は次のようになるであろう．

(1) 個々のオブジェクトを識別し，多様なオブジェクトが構成するオブジェクト世界の構造的な性質，すなわちオブジェクト間の構造的な関連性を明らかにする（構造的抽象化）．

(2) オブジェクトの振舞い，すなわち操作的な性質を明らかにする．

(3) ほかのオブジェクトの振舞いとの関連，すなわち振舞いについて，ほかのオブジェクトに何を要求し，またほかのオブジェクトから何を求められるかを明らかにする（振舞いの抽象化）．

なお一貫性制約はこれらの性質のすべてにかかわるものであるが，ここではとくに (2) および (3) と結び付けてとりあげる．

3. オブジェクト構造の抽象化

多様なオブジェクトはいくつかの抽象化概念によって関連付けられる．これは「個から類へ（類型化）」，「単体から複合体へ（集約化）」，あるいは「特殊から一般へ（汎化）」といった，オブジェクトの構造的な関連性を整理するための考え方を指す．抽象化概念によって実世界の秩序体系は自然な形でオブジェクト世界の秩序体系に写像されるため，オブジェクトの諸性質に関する冗長性と矛盾性が減少する．オブジェクト指向設計を取り入れて再構築したソフトウェアが軽量化するのも，オブジェクトの凝集性と自然な秩序付けによる．

オブジェクト設計は実体の構造的性質の解明を基本とする．その解明には抽象化概念が用いられる．実体の抽象化は意味データモデル[7),14)]において発展してきたが，必ずしも十分形式化されているわけではない．類型化，集約化，および汎化の概念も，実世界の様子を忠実に反映するようさらに細分されるようになった．構造的側面からのオブジェクトの抽象化は以下のように整理される[13)]．

[1] 類 型 化 (Classification)

オブジェクトの構造に関する性質を特性 (property) とよぶ．類型化とは共通の特性をもつオブジェクトの集合を考えることである．この集合をクラスとよぶ．クラスに属するオブジェクトの特性の集合を型，またクラスに属する各オブジェクトをクラスまたは型のインスタンスとよぶ．とくに印字可能なオブジェクト[14)]によって作られるクラス (integer, real, string, symbol, date など) を基本クラスとよぶ．

クラス E のオブジェクトの特性 p は，E のオブジェクト e をあるクラス E' (E とは必ずしも

異ならない）のオブジェクト e' に対応付ける2項関係である．p は E のすべてのオブジェクトが共通にもつ特性という意味で，E の特性ともよぶ．E' を E の特性 p の領域 (domain)，オブジェクト e' をオブジェクト e が特性 p によってとる値という．ただしこの値は時間とともに変わり得る．

オブジェクト間の構造的な関連性（以後単に関連とよぶ）は，次に述べる集約化および汎化・特化の概念によってさらに精密な分類基準を与えられる．

［2］ 集　約　化 (aggregation)

クラス E がそれぞれクラス $E_1, \cdots, E_n$ を領域とする特性 $p_1, \cdots, p_n$ をもつとし，E のオブジェクト e が特性 $p_1, \cdots, p_n$ によってとる値をそれぞれ $e_1, \cdots, e_n$ とする．このときオブジェクト e はオブジェクト $e_1, \cdots, e_n$ の集約化によって定義されているといい，e を集約オブジェクト，各 e_i を e の成分オブジェクトという．また E を成分クラス $E_1, \cdots, E_n$ をもつ集約クラスという．ここで集約オブジェクトと成分オブジェクトの存在依存関係に着目して，集約クラス E と各成分クラス E_i の関連を次のように分けることができる．

（1） has_constituent：E のオブジェクトの存在が E_i のオブジェクトの存在に依存する（たとえば「製品」が「受注」の成分クラスであるとして，「製品」が存在しなければ「受注」は存在しえない．すなわち「受注」は「製品」に存在依存する）とき，E と E_i の関連を has_constituent (E, E_i)，または constituent_of (E_i, E) で表す．このとき一般に成分クラス E_i の構造および振舞いに関する性質は，集約クラス E の性質に制約を与えることがある．

（2） has_component：E_i のオブジェクトの存在が E のオブジェクトの存在に依存する（たとえば「部屋」は「ホテル」の成分クラスであるとして，「ホテル」が存在しなければその「部屋」も存在しえない．すなわち「部屋」は「ホテル」に存在依存する）とき，E と E_i の関連を has_component (E, E_i)，または component_of (E_i, E) で表す．

（3） 汎　関　連 (general relationship)：p_i が E_i を領域とする E の特性であるとき，p_i によって対応付けられる E のオブジェクトと E_i のオブジェクトの存在が互いに相手の存在に依存しないとき，この特性 p_i を汎関連とよぶことにする．このとき一般に，E_i には E を領域とする p_i とは逆の特性 q_i が定義されている．

＜例1＞　ここで Kappel, Schrefl によるオブジェクト図[9),10)]（一部修正してある）を用いてオブジェクトの構造的な性質を図式的に表現しよう．**図-2** はホテルの予約管理環境におけるオブジェクト図である．長方形は一般のクラス，楕円は基本クラスを表す．長方形，楕円の左肩に付けられた名前は特性名である．＊印を付した特性は，対応する領域の複数のオブジェクト，すなわちオブジェクトの集合を値としてとる多価特性であることを示す．

オブジェクト図では長方形 E_i を長方形 E の内部に描くことによって，E が集約クラス，E_i がその成分クラスであることを表す．とくに has_constituent (E, E_i) の場合，E_i を太線で囲んだ長方形で表す．図-2 において，「予約管理」は「ホテル」および「旅行者」の集約クラスで，次の関連が表現されている．

has_constituent（予約管理，ホテル）
has_constituent（予約管理，旅行者）

一方「ホテル」は「部屋」および「予約」の集約クラス，「部屋」は「宿泊客」および「予約」の集約クラス，また「予約」は「予約客」および「部屋」の集約クラスで，次の関連を仮定している．ただし has_component は図には明示的に記入されていない．

has_component（ホテル，部屋）
has_component（ホテル，予約）
has_component（部屋，宿泊客）
has_component（予約，予約客）

さらに「部屋」と「予約」の間に，汎関連を表す特性「部屋予約」および「予約部屋」が定義されているものとする．この事実は「部屋」と「予約」を表す二つの長方形が互いに分離していることによって示される．「部屋予約」と「予約部屋」は互いに逆の関係にある．このことを示すために，部屋予約＊（inverse：予約部屋），予約部屋（inverse：部屋予約）のように指定する．

［3］ 汎化・特化 (generalization, specialization)

汎化はいくつかのクラスの共通的な性質に着目し，その共通性によって一般化されたクラスを定

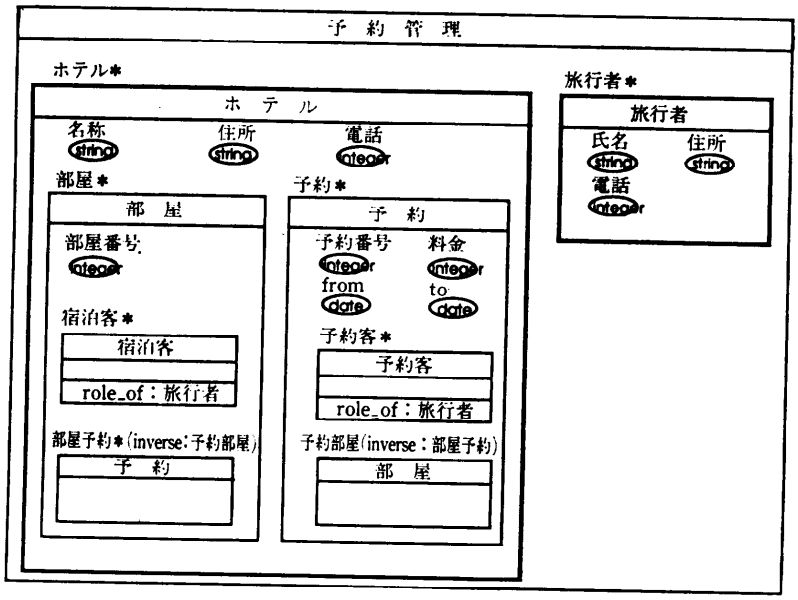

図-2 「予約管理」のオブジェクト図

義する考え方をいう．汎化の逆を特化という．すなわち特化はあるクラスに対していくつかの特性を付加することによって，特殊化されたクラスを定義する考え方である．クラスの特性の集合としての型を考えたとき，汎化と特化の関連をもつ型をスーパタイプとサブタイプという．インヘリタンス (inheritance) は，スーパタイプの性質がすべてサブタイプに継承されることを指す．

ここで特化に着目すると，これは次の二つの場合に分けられる．

(1) subtype_of：クラス E (たとえば自動車) がクラス $E_1, \cdots, E_n$ (たとえば乗用車，バス，トラック) に特化されているとき，各クラスの型に着目して，E_i と E の関連を subtype-of (E_i, E) $(i=1, \cdots, n)$ で表す．

(2) role_of：ある環境におけるクラス E のインスタンス e が，別のいくつかの異なる環境において，それぞれある役割をもつクラス E_i のインスタンス e_i としてみられることがある．この場合 e_i は e と同じオブジェクトである．このとき E_i と E の関連を role_of (E_i, E) で表す．これは(1)の「型の特化」に対して，「インスタンスの特化」ということができる．

<例2> 図-2 において，それぞれ「宿泊客」および「予約客」のオブジェクトは「旅行者」のオブジェクトの中で特別の役割をもつものを指すため，関連 role_of (宿泊客，旅行者) および role_of (予約客，旅行者) が定義される．すなわち「旅行者」の一つのインスタンスは，「部屋」のインスタンスに対する「宿泊客」のインスタンスとして，あるいは「予約」のインスタンスに対する「予約客」のインスタンスとして現れる．一般には一つのインスタンスは複数のクラスのインスタンスになり得る．同一の人が「宿泊客」あるいは「予約客」のインスタンスとして存在しなくなっても，「旅行者」のインスタンスとして存続することはあり得る．role_of は図-2 のようにクラスを表す長方形の底辺に指定する．また図-2 の例には現れていないが，「ホテル」より一般的なクラス「宿泊施設」を考えるならば，subtype_of (ホテル，宿泊施設) が定義されることになる．これも「ホテル」を表す長方形の底辺に指定する．

4. オブジェクトクラスタリング

オブジェクトの OODB への統合には，データベース設計におけるビュー統合および実体クラスタリングの技法を適用することができる[2),5),22)]．とくに後者は多様なクラスの集りを俯瞰し，クラ

ス間の関連を洗練し，OODB に統合する手段として有効である．

クラスタリングは細部から全体へ，木から森へという考えにしたがって，環境別にそれに関与するクラス群をエリアとして集積し，詳細度レベルの異なるエリアの階層を作る実務的な技法として注目される．レベル数は実世界の複雑さや多様性に依存するが，高レベルエリア，主題エリア，および情報エリアの3レベルが実務的には有用といわれる．

クラスタリングは抽象化概念の適用にほかならないが，ここではとくに集約化によるオブジェクトのクラスタリングに着目する．一つのエリアは集約クラスとその成分からなる．ここで集約オブジェクトは成分オブジェクトにメッセージを送ってその操作を起動し，成分オブジェクトのサービスを要求する役割をもち，アクタとよばれる．また成分オブジェクトは集約オブジェクトにサービスを提供する役割をもち，サーバとよばれる．中間レベルのエリアにあるクラスのオブジェクトはアクタおよびサーバのいずれの役割をももつため，エージェントとよばれる．

図-3 のオブジェクト図は企業を構成するクラスのクラスタリングを表し，高レベルエリア，主題エリア，および情報エリアの3レベルから構成されている．高レベルエリアにおける「会社」は「製品管理」と「総務」の集約クラスであり，主題エリアにおける「製品管理」は「受注」と「在庫」，また「総務」は「人事」と「マーケティング」のそれぞれ集約クラスである．情報エリアも図のように集約クラスと成分クラスからなる．たとえば「受注」は「得意先」と「製品」の集約クラスである．とくに情報エリアにおける成分クラスは，情報資源として複数の情報エリアに共有される．たとえば図-3 の情報エリアの成分クラスのうち，「スタッフ」，「得意先」，「製品」は複数の集約クラスに共有されている．

アクタおよびエージェントの役割をもつオブジェクトは，ある業務環境を提供する，あるいはある機能を果たす実体であり，これまで応用プログラムとして位置付けられていたものである．OODB は共有資源としてサーバの役割をもつオブジェクト，すなわち情報エリアにおける成分クラスのオブジェクトによって構成される．

5. オブジェクトの振舞い

5.1 基本的考察

オブジェクトの設計においては，オブジェクト固有の操作を洗い出し，オブジェクトと一体化しなければならない．オブジェクトに関する検索，更新（生成，変更，削除）のうち，とくに更新操作はオブジェクトの一貫性を維持するうえで重要である．ここではオブジェクトの更新操作の体系をオブジェクトの振舞い (behavior) とよぶことにする．OODB における振舞いの設計の目標は次のように整理される．

（1） オブジェクト固有の操作の抽出：まずオブジェクト固有の操作をすべて抽出し，操作の順序関係を整理しなければならない．操作はオブジェクトの状態変化をもたらすものであるから，オ

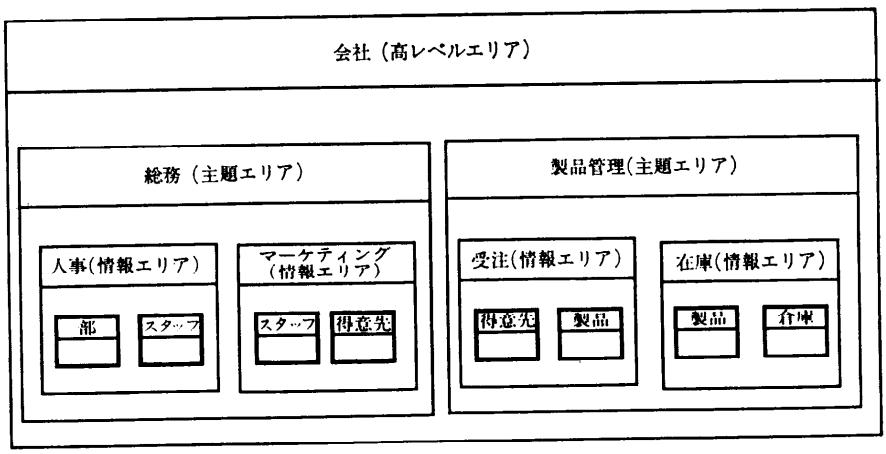

図-3 集約化によるオブジェクトのクラスタリング

ブジェクトの発生，変化，消滅の過程を調べることによって，固有の操作が明らかにされるであろう．オブジェクトがしたがうべき状態変化の順序に関する規則をライフサイクルスキーマあるいは単にライフサイクルという．同一クラスのすべてのオブジェクトは同じライフサイクルにしたがって状態変化する．ここではライフサイクルを形式化し，オブジェクトの振舞いの表現として用いる．

（2） 振舞いの共有性：オブジェクトの振舞いは特定の応用のみに依存せず，長期にわたって多様な応用に供されるよう設計されなければならない．たとえば図-3 の情報エリアにおける成分クラスの例にも見られるように，同じオブジェクトは多様な応用環境の中で共有され，それぞれの環境の中で状態変化するであろう．振舞いの共有性を実現するためには，ライフサイクルには多様な状態変化の過程が反映されなければならない．

（3） 振舞いの抽象化：同一クラスのオブジェクトは共通のライフサイクルにしたがうから，振舞いはクラスに 1 対 1 に対応付けてモデル化されるべきである．クラスは抽象化概念にしたがって関連付けられるから，これに対応して振舞いにも抽象化を導入し，振舞いの構造を洗練する手段が求められる．オブジェクト指向言語におけるメソドの継承は操作に関する特化概念の適用であるが，ライフサイクルの継承という形では十分論議されていない．一方振舞いの集約化は，集約クラスの振舞いが成分クラスの振舞いに分解されるという意味で，振舞いの顆粒化と，同期化を含む操作の制御構造の決定に関係する．

（4） 振舞いにおける一貫性制約：振舞いは一貫性制約に支配される．制約の表現手段をライフサイクルの枠組みの中で与えることによって，制約の体系化の実現が期待される．

5.2 振舞いのモデル

振舞いのモデル化および設計技法についてはいくつかの報告がある[4),9),12),16),17),19)~21)]．ただしモデルの形式化は十分整理されてはいないので，ここでは直観的なグラフ表現を用いて概要を述べる．

オブジェクトの振舞いはライフサイクルとしてモデル化される．ライフサイクルは，オブジェクトの状態 (state) とその状態変化をもたらす事象 (event) の集合によって定義され，ペトリネット[15)]を利用したライフサイクル図として図式的に表現される．

ライフサイクル図はクラスごとに一つ存在する．状態はペトリネットにおけるプレイス（円）に，事象は遷移（直線）に，またオブジェクトはトークンに対応する．任意の時点でクラスの一つのオブジェクトを表すトークンは，ライフサイクル中の一つまたはいくつかの状態の中に存在する．

事象は対応するクラスのオブジェクトに対する操作を表す．事象の入力側，出力側にある状態を，それぞれ事前状態，事後状態という．オブジェクトがある事象の事前状態にあるときその事象は起動可能となり，事象が起動されるとオブジェクトは事前状態から事後状態に移る．

図-4 は「予約管理」に現れるいくつかのクラスのライフサイクル図である．「予約」と「部屋」のライフサイクルは互いに独立に設計されている．「部屋」の状態 'exist' はオブジェクトの存在を示すマクロな状態である．状態 'exist' は図のようにさらにいくつかの状態と事象に洗練される．＊印を付した事象は 0 回以上繰り返し起動され得ることを表す．

振舞いにも集約化概念が適用される．集約クラスの振舞いは成分クラスの振舞いに分解されると考えるのは自然であろう．図-4 の例では，集約クラス「ホテル」の状態 'exist' は，いくつかの業務処理を表す事象と業務処理サービス待ちの状態 'waiting' からなる．「ホテル」の個々の事象は，成分クラス「部屋」および「予約」のいくつかの事象に分解される．たとえば事象「予約処理」は成分クラス「部屋」の状態「更新予約状況」を参照し，「部屋」の「予約割当」，「予約」の「新規予約」，「予約変更」などを適切な順序で起動するであろう．

5.3 振舞いにおける一貫性制約

オブジェクトの一貫性制約はライフサイクルの要素に対応付けることによって整理され，またライフサイクルとともにクラスへの対応付けが可能になる．振舞いにおける制約として，たとえば次の種類が考えられる．

（1） 状態制約：ライフサイクルにおける個々の状態に対応して制約を設けることができる．この制約は，たとえば「部屋」の状態「更新予約状

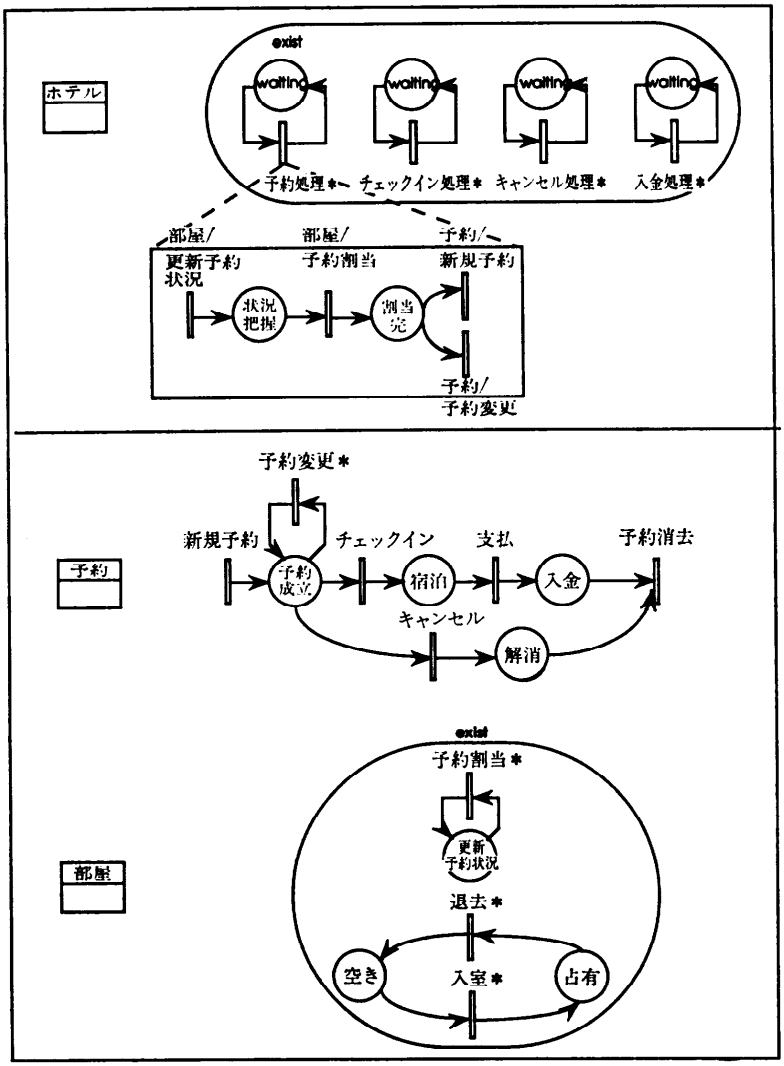

図-4 「予約管理」におけるクラスのライフサイクル図

況」において予約の重複がないこと，あるいは状態「空き」において宿泊客がないことなどを表現する形で記述される．

（2） 領域状態制約：E' を領域とする E の特性 p が汎関連であるとする．領域状態制約は，「クラス E のオブジェクト e がある状態にあるためには，e が特性 p によってとる E' のオブジェクト e' が特定の状態になければならない」という制約である．たとえば「予約」のオブジェクトが状態「宿泊」になれば，特性「予約部屋」によってとられる値，すなわち領域「部屋」のオブジェクトは状態「占有」にならなければならない．

（3） 集約状態制約：クラス E' がクラス E の成分クラスで，has_component (E, E') の関連があるとする．集約状態制約は，「E のオブジェクト e が特定の状態にあるときのみ，e の成分オブジェクト e' はある状態になり得る」という制約である．たとえば「ホテル」のオブジェクトが「割当完」の状態においてのみ，その成分である「予約」のオブジェクトは「予約成立」の状態になり得る．

（4） 成分状態制約：クラス E' がクラス E の成分クラスで，has_constituent (E, E') の関連があるとする．成分状態制約は，「E' のオブジェクト e' が特定の状態にあるときのみ，e' を成分オ

ブジェクトにもつ E の集約オブジェクト e はある状態になり得る」という制約である.

(5) 同期化制約:「クラス E のオブジェクトの事象は,クラス E' のオブジェクトがある状態になることがトリガとなって起動される」という制約である. E, E' は同一クラスであってもよい. たとえば「ホテル」の「予約処理」の要素の中で事象「予約割当」にキャンセル待ちを考慮するならば,「キャンセル処理」によって「予約」が「解消」状態になるのを待って,「予約割当」が起動される.

これらの制約はいずれもなんらかの形式論理を用いて表現可能であり,いくつかの試みがみられる[16),21)].

6. おわりに

現在多数の OODB システムの試作,商品化が進められているが,重要なのは OODB の設計方法論であろう. 本稿ではその前提となる基本的な考え方を述べるにとどまったが,これは次のように要約される.

(1) オブジェクトを,その構造的性質に着目して秩序体系化することが基本である. そのためには表現力の高い抽象化機能をもつ標準的なオブジェクトモデルと言語の開発が望まれる.

(2) OODB は,多くの応用から共有資源として利用されるオブジェクトによって構成される. OODB の共有性と拡張性を高めるためには, OODB を構成するオブジェクトとこれを利用するオブジェクトを統合化する技法の確立と経験の蓄積が必要である.

(3) オブジェクトの振舞いと一貫性制約は,研究,開発の余地が残された分野である. OODB の質を決定する振舞いと制約のモデルは,オブジェクトモデルの枠組みの中で早期に確立されるべきである.

最後に,本稿の内容について多くの詳細かつ興味深いご指摘をいただいた大堀淳氏ならびに査読者の方に深謝いたします.

参考文献

1) Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D. and Zdonik, S.: The Object-Oriented Database System Manifesto, Proc. 1st Int. Conf. on Deductive and Object-Oriented Databases (1989).
2) Batini, C., Lenzerini, M. and Navathe, S. B.: A Comparative Analysis of Methodologies for Database Schema Integration, ACM Computing Surveys, Vol. 18, No. 4, pp. 223-364 (1986).
3) Bloom, T. and Zdonik, B.: Issues in the Design of Object-Oriented Database Programming Languages, Proc. Object-Oriented Programming Systems, Languages and Applications 87 (1987).
4) Brodie, M. L. and Ridjanovic, D.: On the Design and Specification of Database Transactions, in: On Conceptual Modeling, Springer-Verlag (1986).
5) Feldman, P. and Miller, D.: Entity Model Clustering: Structuring a Data Model by Abstraction, The Computer Journal, Vol. 29, No. 4 (1986).
6) Greenspan, S., Borgida, A. and Mylopoulos, J.: A Requirements Modeling Language and Its Logic, in "On Knowledge Base Management Systems" (eds. Brodie, M. and Mylopoulos, J.) (1985).
7) Hull, R. and King, R.: Semantic Database Modeling: Surveys, Applications, and Research Issues, ACM Computing Surveys, Vol. 19, No. 3 (1987).
8) Jarke, M., Jeusfeld, M., Mertikas, M., Schmidt, J. W. and Wetzel, I.: Database Application Development as an Object Modeling Activity, Proc. 16th Int. Conf. on Very Large Database Systems (1990).
9) Kappel, G. and Schrefl, M.: Using Object-Oriented Diagram Technique for the Design of Information Systems, Proc. Int. Working Conf. on Dynamic Modeling of Information Systems (1990).
10) Kappel, G. and Schrefl, M.: Object/Behavior Diagrams, Research Report Technical University Wien (1990).
11) Kim, W.: Object Oriented Databases: Definition and Research Directions, IEEE Trans. on Knowledge and Data Engineering, Vol. 2, No. 3 (1990).
12) Mylopoulos, J., Bernstein, P. A. and Wong, H. K. T.: A Language Faculity for Designing Database-Intensive Applications, ACM Ttrans. on Database Systems, Vol. 5, No. 2 (1980).
13) Neuhold, E. J. and Schrefl, M.: Dynamic Derivation of Personalized Views, Proc. 14th Int. Conf. on Very Large Data Bases (1988).
14) Peckman, J. and Maryanski, F.: Semantic Data Models, ACM Computing Surveys, Vol. 20, No.

3 (1988).
15) Peterson, J. L.: Petri Net Theory and the Modeling of Systems, North-Holland (1981).
16) Put,F.: The ER Approach Extended with the Action Concept as a Conceptual Modeling Tool, Proc. 7th Int. Conf. on Entity-Relationship Approach (1988).
17) Sakai, H.: A Method for Entity-Relationship Behavior Modeling, Proc. 3rd Int. Conf. on Entity-Relationship Approach (1983).
18) 酒井博敬, 堀内 一: オブジェクト指向入門, オーム社 (1989).
19) Sakai, H. and Horiuchi, H.: A Method for Behavior Modeling in Data Oriented Approach to Systems Design, Proc. 1st Int. Conf. on Data Engineering (1984).
20) Sakai, H.: An Object Behavior Modeling Method, Proc. 1st Int. Conf. on Database and Expert Systems Applications (1990).
21) Schrefl, M.: Behavior Modeling by Stepwise Refining Behavior Diagrams, Proc. 9th Int. Conf. on Entity-Relationship Approach (1990).
22) Teorey, T. J., Bolton, D. L. and Koenig, J. A.: ER Model Clustering as an Aid for User Communication and Documentation in Database Design, CACM, Vol. 32, No. 8 (1989).
23) 堀内 一: データ中心システム設計, オーム社 (1988).

(平成2年12月4日受付)

酒井 博敬 (正会員)

昭和 36 年京都大学理学部理学研究科修士課程 (数学専攻) 修了. 同年日立製作所, 昭和 58 年日立ソフトウェアエンジニアリング, 昭和 59 年京都産業大学教授, 現在中央大学理工学部管理工学科教授, 工学博士. 研究テーマ: オブジェクト設計方法論. 著書「情報資源管理の技法」, 「オブジェクト指向入門」. 電子情報通信学会, ACM, IEEE 各会員.