第 16 章　データ中心アプローチ

データ中心アプローチへの道

1970 年代になって、いくつかの種類のデータベース管理システムを我々が容易に入手できるようになり、ピーター・チェンの論文 [CHE76] がデータベース理論の発展の起爆剤になって、データベースを作り、利用することについての考え方や理解が深まった。一方ビジネス・アプリケーションでは、処理（プロセス）はしばしば変更になるけれど、データ構造はたいへん安定しているということについての認識が広がって、さまざまな面からのデータ中心アプローチの前提が整ってきた。その上で、「安定したものをベースに構築した情報システムは、不安定なものを基にした情報システムよりも、システムとして安定している」ということ、「データを社内のいろんな部署で共用することはビジネス遂行の上でも効果が大きい」ということなども我々が理解するようになって、データ中心アプローチの花が開いた。

データ構造の安定についていえば、私自身の経験が 1 つのサンプルになるだろう。私は 1962 年（昭和 37 年）に大学を卒業して、ある証券会社でプログラマとして仕事を始めた。

証券会社のビジネスのポイントは、顧客からの株式売買の注文を証券取引所につないで売買の約束（証券界ではこれを「約定（やくじょう）」と呼ぶ）を成立させ、その 4 日目に証券と金銭を交換してその約束を履行し（「受け渡し」と呼ぶ）、その一連の作業に関して顧客から手数料を頂くことにある。受け渡しを的確に遂行するために株式売買のデータは、私が駆け出しのプログラマだった頃から既にかなり複雑だった。注文段階で項目数がすでに約 10 項目あり、約定成立後にはこれが 20 項目弱になった。それから 45 年が経過し、データ面でいえば約定成立後の項目として税金関係の項目が 2 つ増えたに過ぎない。

しかし、株式売買のプロセスはどうだろうか。以前は顧客からセールス、株式部、市場部の順にその間を電話だけで流していた注文を、私が入社した証券会社は私の入社直前に、テレタイプを導入して営業店と市場部の間を直結し、従来の電話と両方を使えるシステムを作り上げたところだった。そのテレタイプがすぐに情報システムによるオンライン伝送に変わった。しかし株式の売買は長い間「場立ち（ばだち）」と呼ばれた証券会社の若い社員が証券取引所の立会場の中を走り回り、独特の手信号で情報の伝達をしあって、成立させていた。

それが今では、証券取引所のコンピュータが静かに売買を成立させている。顧客からの注文もセールスが取り次ぐのではなく、顧客自身が自分でコンピュータに入力し、その注文がインターネット経由で証券会社のコンピュータを経由してそのまま取引所に届くシステムに変わっている。

データの変化に比べると、プロセスの変化はたいへんに著しい。もっとも、いくらデータ構造が安定しているといっても、テレタイプ時代にデータ中心アプローチで作られたシステムが仮にあったとしても、今の取引所の自動化の時代に対応できないのは明らかである。

余談だが、「データ中心アプローチ」という言葉は和製英語で、欧米では通用しない。欧米ではこの方式を「インフォメーション・エンジニアリング」と呼ぶが、本当のインフォメーション・エンジニアリングとデータ中心アプローチとは対象の広さや深さの面で、後述するように大きな違いがある。

データ中心アプローチの考え方

データ中心アプローチでビジネス・アプリケーションを構築する時でも、各プログラムの目

的が出力を作成するところにあるということは、構造化技法の場合と変わらない。データ中心アプローチではその必要な出力を作るために、「何のデータをデータベースに持てばよいのか」の議論から検討を始める。このデータベースに格納するデータを明らかにする作業を、「データ分析」という。データ分析の具体的な手順は、すぐ後で述べる。

データベースが固まると、入力データからそのデータベースを更新する手順と、更新済みのデータベースから必要な出力を作成する手順をそれぞれ付け加えることで、情報システムの全体像が完成する。

「データ分析」の方法

それではデータ中心アプローチでの中心的存在であるデータベースは、どのようにすれば設計できるのだろうか。これには、トップ・ダウンとボトム・アップの 2 つの方法がある。

トップ・ダウンでこれを行うには、充分な経験を積んだスーパーSE が必要になる。彼／彼女は仕様書に目を通しただけで実体関連図を描いて、データベース化の対象物を明確にできる。この方法だと時間は少なくてすむが、スーパーSE がいなければ実施できないという難点がある。それに対してボトム・アップの方法は、時間はかかるけれど基本的には誰でも行うことができるという利点がある。

以下で、そのボトム・アップの方法の概略を述べる。しかし以下に述べるものは、非常に一般的な方法である。今の日本には、データベースの設計法に関していくつかのすばらしい方法が紹介されている（[SAT05]、[TUB05]）。実際の情報システムの構築を行う場合には、その設計法の 1 つを導入して行うことが望ましい。

ボトム・アップの方法によるデータベース化の対象物を明確にする作業は、これから開発しようとする情報システムの出力を分析することから始まる。データベースとは「必要な出力を作るためのデータを格納したもの」であるから、その情報システムで何を出力するべきかを明確にすると、必然的にそのデータベースに格納するものが明確になる。別のいい方をすると、出力する必要のないデータはデータベースに格納する必要がない。これは、自明のことである。データベースに格納するべきデータは、この自明のことを裏返しにした作業を行うことによって、明らかにすることができる。

まだ手作業でも行われたことがない新たな業務をシステム化する場合でも、その情報システムで作成しなければならない出力を明確にすることができれば、そのシステム化で以下で述べる方法を採用することができる。

ここでいう出力には、画面と帳表の他に、他のアプリケーションとのインタフェースになるファイルや、これからの情報システムでは、他の情報システムやユーザにネットワーク経由で送るメッセージも対象にすることが必要である。

出力の種類とそれを構成しているデータ項目が確定すると、そのデータ項目を全部名寄せする。つまり全部の出力からデータ項目だけを集めて、重複しているものは 1 つだけを残して後は消去する。データの標準化（「1 つのデータ項目に 1 つの名前」を実現すること）が実施されていると、この名寄せでは表面的に名前だけを見て消去すればよい。データの標準化が実施されてなければ、名前だけを見て単純に消すことができないため、消すに当たって注意が必要になる。インフォメーション・エンジニアリングが唱える「コーポレート・データベース」を実現するのであれば、データの標準化を実施しなければならない。いずれにしろここでは、重複と漏れのないデータ項目の集合が求まればよい。

次にこのデータ項目を 1 つずつ、何に属するデータ項目であるのかを丹念に問いかけて、データ毎にグループを作成する。例えば「顧客氏名」は顧客に属するデータ項目であり、「従業員の住所」は従業員に属するデータ項目である。「商品単価」が商品毎に付けられた標準単価であれば商品に属するデータ項目であり、「取引単価」が標準単価を参考にして決められる取引毎の単価であるなら取引に属するデータ項目である。この段階では、対象業務をよく知っているユーザの参画が必要になる。

このようにして作られたデータ項目の集まりであるデータが、データベース化の対象物である。ここで、そのデータに名前を付ける。あるいは改めて名前を付けるといわなくても、それを意識してこれまでグループ分けを行ってきたのかもしれない。

ここでこの名前には、2 つの種類がある。純粋の名詞（「する」を付けても動詞にならないもの。例えば、顧客、商品など。）と、動詞の語幹としての名詞（「する」を付けると動詞になるもの。例えば、取引、購入など。）である。以降で純粋の名詞を「実体」、動詞の語幹を「関連」と呼ぶ。

正規化について

更新が入るデータベースでは、そのデータベースに格納するものには「正規化」と呼ばれる作業を行わなければならない[MAS03]。これはデータベースを更新するに当たって、更新の対象ではないデータを消去してしまったり、変更ししてしまったりする「更新異常」と呼ばれる事態を引き起こさないようにするためである。

ある概念データモデル作成の方法論提唱者は、「正規化によって“One fact, One Place”が実現される」と表現している。正規化を正しく行うと、まさに「データベースの 1 つの場所には 1 つの事実だけが格納され、逆に 1 つの事実は 1 つの場所にしか格納されていない」ということを実現することができる。ただしパフォーマンスなどを考慮して、意識してデータベースの複製（レプリカ）を持つ場合は別の話になる。

我々が普通に使うデータベースでは、正規化は「第 3 正規形」と呼ばれるところまで行えば良いとされている[MAS03]。しかし次の「実体関連図の描き方」で述べるように作業すると、第 3 正規形の次に高いレベルの「ボイス・コッドの正規形」のレベルで正規化を行うことができる。

上に述べたデータ分析の方法で、データの冗長性がない、コンパクトなデータベースを作ることができる。ただし教科書に書いてある正規化の方法で作業を行うと、ユーザにとってなじみがある「情報」がバラバラの「データ」にばらされ、分かりにくい、難しいものに変わってしまうという印象が持たれる可能性がある[MAS03]。

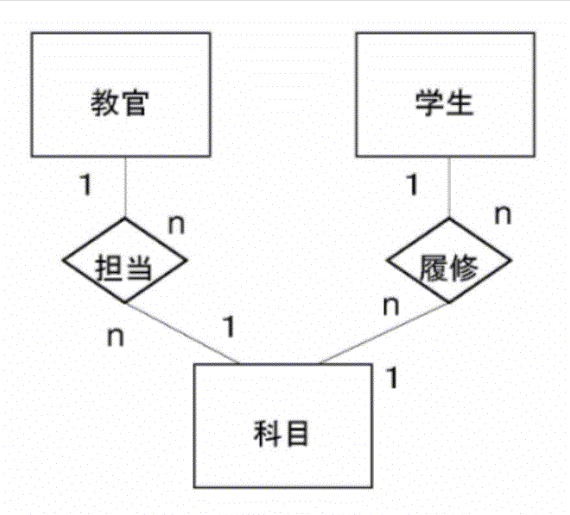
このようなこともあって、データ・ウェアハウスなど更新を伴わないデータベースの場合は、そもそも「更新異常」が起きるはずがないということから、正規化の作業は必要としない。

実体関連図の描き方

データベース化の対象物を決めたとき、その対象物を名前が純粋な名詞で表されるもの、つまり「実体」と、動詞の語幹で表されるもの、つまり「関連」に二分した。もう一度ここで、名詞で表される実体を「人など」に関連するもの（例えば、顧客、学生など）と「ものなど」に関連するそれ以外のもの（例えば、商品、科目など）に分ける。結果としてデータベース化の対象物は、次の 3 つに分けられたことになる。

- 人などに関わるもの
- 動詞の語幹
- ものなどに関わるもの

ここで、人などに関わるものを主語、動詞を述語、ものなどを目的語にして、全ての組み合わせで文章を作ってみる。その結果はその業務で、あるものは正しく、あるものは正しくない、という形になる。その正しいものだけを残しておき、実体関連図（ER 図、Entity relationship diagram：ERD）に記述する。



図表 16-1　実体関連図の例

実体関連図では、実体（名詞）は四角、関連（動詞）は菱形で表し、その間を線で結ぶ。したがって 1 つの文章は、主語と目的語の 2 つの四角（実体）と述語の 1 つの菱形（関連）が一本の線で結ばれた形になる。何度も同じ実体が図の上に表れるとき、それらをまとめて、1 つにしても良い。

その線の実体と関連に近いところに、多重度を記す。多重度は、普通は実体側が 1 、関連側が n （1 以上）になることが多いが、時には関連側も 1 になることもある。

このようにして、2 つの実体の間に必ず関連を挟む形で概念データモデルを作ると、そのモデルは第 3 正規形よりもう一つレベルの高い「ボイス・コッドの正規形」になる[MAS03]。

実体関連図のシンプルな例を、図表 16-1 に示す。

キィについて

ここで、全ての実体と関連を構成するものをそれぞれ唯一無二に識別できるデータ項目を探す。探して見つからなければ、そのようなデータ項目を作ることがある。このデータを唯一無二に識別できるようにするデータ項目を「キィ」という。

実体の場合は、キィ専用のデータ項目を持つのが普通である。例えば、顧客には顧客コードを用意してそれをキィにする、商品の場合は商品コードを持つ、という形にする。既にこのようなデータ項目が用意されているのが普通だが、仮に用意されてない場合は、SE が自分で作る必要がある。

関連は 2 つの実体に挟まれているので、その 2 つの実体のキィ項目の組み合わせをまずキィとする。例えば、「顧客が商品を購入する」というというケースでの「購入」のトランザクションのキィは、まず顧客コードと商品コードの組み合わせで考える。その組み合わせだけでそのトランザクションが唯一無二に識別できるのなら、キィはこれに決めればよい。

しかし、これでは決まらないのが普通である。これで決まるケースとは、その顧客が彼／彼女の人生の間で、その商品を購入することは常にたかだか一回しかないことを意味する。普通は、そのようなことはあり得ない。このような場合には、何かのデータをこれに付加して「唯一無二」を実現する。例えば、同じものを一日に二回以上買うケースが全くない場合には、購入年月日をキィの 3 つ目の項目にすればよいかもしれない。一日に何回も買うケースがある場合はどうするか。それを決めるのが、ある意味での SE の腕の見せ所になる。

データ中心アプローチによる情報システム構築手順

以上の議論と繰り返しになるが、データ中心アプローチによる情報システムの構築手順は、以下のようになる。

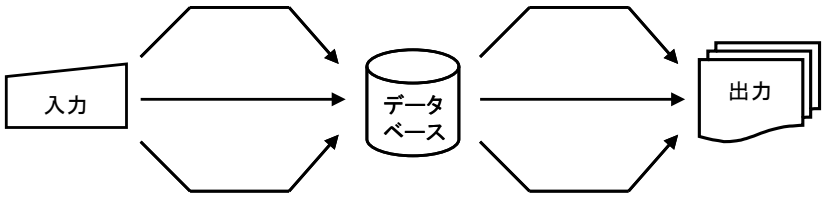
1. データ分析を行い、システムの目的を達成するために必要なデータを明らかにする。
2. データベースを設計する。
3. データ加工の処理の内容を明らかにする。
4. プログラムを設計する。
5. プログラムとデータベースを作成する。
6. テストする。

データ中心アプローチの特徴

構造化技法では、入力から直接出力を作成する方法を考えていた。それと比較するとデータ中心アプローチでは、情報システムの中心にデータベースを位置づけて、それが常に最新の正しい状態を保つことを大命題とし、そのデータベースから必要な出力を作るという形にしたところに大きな特徴がある。

またデータ中心アプローチの特徴の最大のものの 1 つは、組織全体でデータベースの共用を可能にしたことである。構造化技法では、その必要性が強く議論されていたにもかかわらずそれをなかなか実現することができなかった。

さらにコンピュータ処理の中心にデータベースを据えることで、情報システム全体の構造がたいへんシンプルになった。具体的には、データの発生時にそのデータを受け取ってデータベースに反映させ、出力が必要な時にその更新済みのデータベースから必要な出力を作成する形になった。このことから分析者の個性が情報システムに反映される部分が少なくなって、構造化技法で作ったものと比較すると情報システムを理解しやすくなったという変化が起きた。



図表 16-2　データ中心アプローチで作られる情報システムの広がり

またデータ中心アプローチによって、情報システムにおけるプログラムとデータについての考え方が変わった。つまり、データ中心アプローチでは、次のような考え方を取る。

「組織にとって大切なものはデータであって、プログラムではない。プログラムはデータを処理するために使うものである。したがって仮にプログラムを外部から購入してその機能を果たせるなら、何も自社内でプログラムを作る必要はない。」

この言葉を最初に聞いた時、私はまだ単なるプログラマだった。どうすれば良いプログラムを作ることができるのかを毎日真剣に考え、実践しようとしていた。さらに自分たちが作るプログラムは、私が所属する企業にとって必要不可欠なものと信じていた。だからこの言葉が私のその頃の生き甲斐や目標を無視していることに、たいへんに腹を立てた。しかし考えてみると、組織にとってこの言葉は正しい。この日を境に私は、データ中心アプローチの信奉者に代わった。既に ERP パッケージはたいへんに普及しているが、この ERP パッケージの利用はこの考え方に基づいている[1]。

データ中心アプローチの処理のパターンを、図表 16-2 として示す。これは構造化技法で示した図表 15-5 と同じものを表そうとしているものである。両者を見比べて、データ中心アプローチの特徴を明確に認識していただきたい。

ただデータ中心アプローチは、開発方法論としては中途半端なものかもしれない。専用のモデル記述の方法を持たず、プログラム言語も持たず、全て構造化技法のものを使う形になっている。つまりデータベース化の対象物は実体関連図（ER 図）で表し、プログラムは COBOL などで記述する。表面的には、構造化技法と差がないように見える。

しかし内容は、構造化技法と大きく異なっている。オブジェクト指向技法と構造化技法の違いは大きいが、考え方からいえばデータ中心アプローチは、構造化技法よりオブジェクト指向技法に近い。つまりデータ中心アプローチの少し先に、オブジェクト指向技法が位置づけされていると私は見ている。

データ中心アプローチの将来

データベースの共用は、ビジネス・アプリケーションでの必要な要件の 1 つである。しかしオブジェクト指向技法では、データの保存方法が情報隠蔽の対象になってその詳細が公開されない。したがってオブジェクト指向技法ではデータベースを保持ししているかどうかという情報すら公開されず、データベースの共用は図ることができない。このためこの領域では、データ中心アプローチが引き続き使われ続けるということが起きている。

具体的には、データベースに関する部分はオブジェクト指向技法がデータ中心アプローチの考え方と方法を取り込んで、ハイブリッドな開発方法論がすでに実現している。この場合、使用されるデータベースはリレーショナルデータベース、プログラム言語は COBOL や C 言語のような構造化言語ではなく、Java などのオブジェクト指向言語である。

いずれにしろ何らかの形で、データ中心アプローチはビジネス・アプリケーションと共に生き続けて行くものと思われる。

インフォメーション・エンジニアリングとエンタープライズ・アーキテクチャ

---

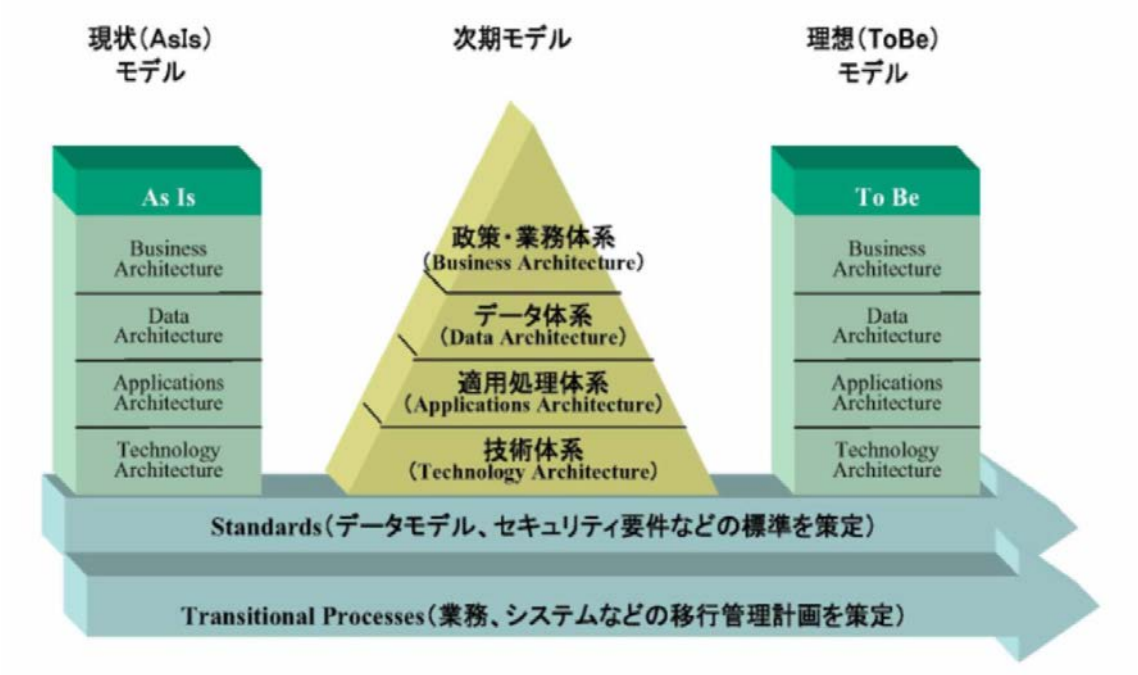
[1] 超高速開発で生成される情報システムも、データ中心アプローチの考え方をベースにするものが多い。超高速開発については、第 28 章で記す。

1980 年代の終わり頃にアメリカで、「インフォメーション・エンジニアリング」という言葉がよく使われたことがある（[MART89]、[MART90a]、[MART90b]）。これを提唱した人たちは、「情報システムを構築するに当たって、まずコーポレート・データベースを設計しなさい。そうすることで、組織にとって何が重要なデータであるかということが明確になり、それによって情報システム化の優先順位も明確になる」と主張していた。

この主張は、まっとうなものだったかもしれない。しかし、現実的なものとはいえなかった。このため日本では、コーポレート・データベース作りを始めた企業はなかったと推測する。しかしアメリカでは、何社かがこれを手がけたという話を当時聞いた。それでも結局インフォメーション・エンジニアリングは、大きな花を咲かせることなく散ってしまった。

今これに関連する考え方は、エンタープライズ・アーキテクチャ（Enterprise Architecture : EA）と呼ばれる形に変わっている[METI06]。インフォメーション・エンジニアリングでは、単にデータについて議論していただけだった。しかしエンタープライズ・アーキテクチャでは、政策・業務体系、データ体系、適用処理体系、技術体系の 4 つの面からアプローチをかける。インフォメーション・エンジニアリングのアプローチより、もっと本格的である。この考え方を、図表 16-3 に示す。

我々の組織の情報システムも、どこかのタイミングで一度はこういう本格的なアプローチをとることが必要になるだろう。



図表 16-3　EA の考え方（[MET06]より）

キィワード

データ中心アプローチ、データベース、データ分析、実体関連図、ER 図、ERD、データの標準化、実体、関連、正規化、第 3 正規形、ボイス・コッドの正規形、概念データモデル、キィ、ERP パッケージ、インフォメーション・エンジニアリング、コーポレート・データベ

ース、エンタープライズ・アーキテクチャ、EA

略語

ERD：Entity relationship diagram
EA：Enterprise Architecture

人名

ピーター・チェン（Peter Pin-Shan Chen）

参考文献とリンク先

[CHE76], “The Entity-Relationship Model – Toward a Unified View of Data, “ ACM Transactions on Database Systems, Vol.1, No.1, pp9-36, 1976.
このペーパーは、ACM のデジタル・ライブラリ（http://portal.acm.org/portal.cfm）からダウンロードすることができる。（ただし、ACM のメンバーであることが必要。）

[MART89] ジェームズ・マーチン著、竹林則彦監修、三菱 CC 研究会 IE タスクフォース訳、「インフォメーション・エンジニアリング I　統合化 CASE のための方法論」、（株）トッパン、1991 年.
この本の原書は、以下のものである。
James Martin, “Information Engineering Book 1 Introduction,” Prentice-Hall, 1989.

[MART90a] ジェームズ・マーチン著、竹林則彦監修、三菱 CC 研究会 IE タスクフォース訳、「インフォメーション・エンジニアリングⅡ 統合化 CASE による計画と分析」、（株）トッパン、1992 年.
この本の原書は、以下のものである。
James Martin, “Information Engineering Book 2 Planning & Analysis,” Prentice-Hall, 1990.

[MART90b] ジェームズ・マーチン著、竹林則彦監修、三菱 CC 研究会 IE タスクフォース訳、「インフォメーション・エンジニアリングⅢ 統合化 CASE による設計と製作」、（株）トッパン、1994 年.
この本の原書は、以下のものである。
James Martin, “Information Engineering Book 3 Design & Construction,” Prentice-Hall, 1990.

[MAS03] 増永良文著、「リレーショナルデータベース入門［新訂版］－データモデル・SQL・管理システム－」、Information & Computing – 43、サイエンス社、2003 年.

[METI06] 「EA ポータル」.この情報は以下の URL からダウンロードできたが、今は削除されている。（確認日：2017 年（平成 29 年）1 月 11 日）
http://www.meti.go.jp/policy/it_policy/ea/index.html

[SAT05] 佐藤正美著、「データベース設計論－T 字型 ER 関係モデルとオブジェクト指向の統合をめざして」、ソフト・リサーチ・センター、2005 年.

[TUB05] 椿正明著、「名人椿正明が教えるデータモデリングの技 データ中心システム開発原論」、翔泳社、2005 年.

（2007 年（平成 19 年）6 月 2 日　初稿作成）
（2014 年（平成 26 年）3 月 21 日　一部修正）
（2016 年（平成 28 年）4 月 14 日　一部追加）