データ分析に於けるデータ項目整理要領と データ中心のシステム構造について

グローバルフォーカス（株）

■ 執筆者Ｐｒｏｆｉｌｅ ■

小見山　昌　雄

1972 年　（株）オリエントファイナンス入社
　　現：（株）オリエントコーポレーション
　　システム業務担当
1999 年　グローバルフォーカス（株）分社
2002 年　現在 経営企画部所属
　　DOA 設計技法コンサルタント

■ 論文要旨 ■

データ分析・データベース設計といえば「正規化技法」だが，一般的な技法の知識だけでは，実際のデータベース設計に適用するのは難しい．

そこで，ほかの技法でデータ分析を進める中で，データ分析の普遍的実施要領として得られたノウハウで「正規化技法」の実務的実施要領を整理すると同時に，データ中心アプローチの解説では，さりげなく回避されている導出項目の取り扱いの問題について，実装を踏まえた整理の基準・要領をまとめる．

また，データベース設計とプログラムの設計は関連がないと一般に認識されているが，データベースが本来の一元的で共用可能なデータ基盤として確立された場合に，その形を最大限に生かすデータベースの実装要領について述べる．

■ 論文目次 ■

■ 図表一覧 ■

はじめに

今の時代に，企業の業務システムでデータベースを使っていないものは，まず考えられない．

しかしながら，それは，ほとんどの場合，単に Oracle などのＤＢＭＳ配下にデータを置いていることを意味しており，データベースの定義に沿ったデータ運営が行われていることはまれである．

ＢＰＲ・ＳＣＭ・ＣＲＭやＤＷと，業務系・分析系を問わず，統合化・一元化の潮流の中にありながら，データベースが整備されていない（ＤＢＭＳ配下ではあるが，一般のファイルシステムと変わらない）のでは，本来の目的を達成しないばかりか，結局，データの多重化を許し，混乱を増すことになる．

ちなみに，それらのツールとして販売されている各種ソフトは，皮肉なことにそのほとんどが整備されたデータベースの存在が前提となっている．

また，正規化のことを問うと，知らないエンジニアはほとんどいない．

言下に「第一正規化は繰り返し項目の排除云々・・・」と手順の解説をしてくれるのは良いが，これまた「では，演習問題にある画面について正規化して下さい」というと，「いや，じつは・・・」ということになる．

要するに「テストに出るので正規化の定義は覚えても，データベース設計の品質を問われることはないので，応用可能な技術として身につける必要はない」ということのようである．

データベース設計に関して，解説に数学的な表現が用いられることが多いことも，実務者の理解を阻む一因と考えられるが，実際の設計要領はまことに業務の理にかなったもので，業務に精通していれば，分析・設計の方法を覚えること自体は容易である．

ここでは，安定性の高いデータベースを設計する実務的データベース設計要領と，それを実装する場合のポイントについてまとめた．

１．データベースとデータ整備

１．１　データベースとデータベース管理システム

データベースはデータベース管理システムと混同されており，大多数の企業では，従来のファイルシステムと同じ設計内容をデータベース管理システムで実装したものを指してデータベースと称している．

もちろんデータベースはその定義（複数の処理目的に共用される，相互に関連付けられた冗長性のないデータの集まり）に定められているように，概念的な存在であり，「冗長性のないデータの集まり」を形成するようにデータ整備をおこない「複数の処理目的に共用」できるかたちで実装されていないと，データベースの効果として期待される生産性や保守性の向上を実現することはできない．

以下，一般的なデータ整備の技法である，正規化と佐藤正美氏が提唱するＴ字形ＥＲ技法を比較しながら，如何なるデータ項目の集合に整理するのが最適であるかを検討するデータベース設計の要領について検討する．

１．２　データ整備要領

正規化については多くの文献に解説されているが，いずれも正規化の手順を機械的な操作手順として解説したものであり，実際の作業要領として適用することは難しい．

ここでは，佐藤正美氏が提唱するＴ字形ＥＲ技法の分析アプローチとの比較で明らかとなった，関数従属の盲点とその対処について述べる．

(1) Ｔ字形ＥＲ技法の考え方

正規化技法で第二正規化・第三正規化属の関数従属の解析に該当する部分を，Ｔ字形ＥＲ技法ではアイデンティファイアの捕捉という．

正規化技法では，従属の相手としてキーの確認をおこないながら，一つのデータ項目集合からエンティティを切り出していくが，Ｔ字形ＥＲ技法では，番号・コードなどのキーワードを持つデータ項目を，存在・行為などの実体を象徴しているものという意味で，特別なデータ項目「アイデンティファイア」とし，それで象徴される管理対象の存在を個別に捕捉する．

「アイデンティファイーとは，エンティティにただ一つ存在し，エンティティを代表するデータ項目」と定義されている．

ここには「業務上大量に発生する顧客・売上・商品などの管理対象には，それを特定するために，人工的なコードを付与して管理を容易にする工夫がされている」と言う概念的要件と，それをコンピュータであつかうためには，「レコードを一意に識別する人工的なコードを付与せざるを得ない」という物理的要件があるが，かならずしもこれがイコールにならないところに，従来の正規化技法のみでデータ構造の整理をおこなおうとする困難さが存在する．

すなわち，名前・年月・場所などで特定できる件数の限界を超えた管理対象には，その管理を容易にするために独自のコードを付与しなければ，システム対応の有無に係わらずその扱いが煩雑になるので，コードが付与されているが，逆に，名前・年月・場所など，その管理対象自体が持つデータで何とか特定可能な範囲の量であれば，独

自の管理コードが付与されていないので，機械的な操作だけで，それらの存在を認識することが困難となる．

実際業務の分析現場では，これら独自のコードを付与されないケースや，複数の管理対象に同じコードを付与しているケース（複写伝票などの使用）がかなりの頻度で出現しており，意識的に分離を試みない限りは，もとの集団に残ることになり，データ構造の安定性を損なう一つの要素となる．

(2) 正規化技法

Ｔ字形ＥＲ技法の考え方から，正規化技法の従属概念を整理すると「顧客氏名が顧客番号に従属している」わけではないことはあきらかとなる．

「顧客氏名も顧客番号も顧客という管理対象のデータという位置づけは同じだが，顧客番号は，業務上の必要から，大量の顧客を容易に識別する目的で，個々の顧客を特定するために人工的に付与したものなので，結果的に顧客番号が顧客を象徴しており，あたかも顧客に関するほかのデータ項目は顧客番号に従属しているように見える」のである．

よって，キーに従属するデータ項目を整理することは，本来キーで象徴している実体を想定し，その実体に関するデータ項目の集合を特定する作業である．

(3) 従属概念の問題

名前・年月・場所などで特定できる件数の限界を超えた管理対象は，（システム対応の有無に係わらず）その管理を容易にするために独自のコードを付与するので，業務上の管理対象を象徴するデータ項目を管理対象自体と見なして，ほかのデータ項目は従属しているとの解釈でもデータ整理に支障はない．

問題は「名前・年月・場所などのデータ項目で何とか特定可能な量なので，独自の管理コードが付与されていないケース」や，当初そのつもりであったものが，業務の変化や管理様式の変更で想定された件数を大幅に上回る結果となったものである．

則ち，正規化技法で説明されているように，もとのデータ集合の中からキーを見つけて，そのキーへの従属性を問う分析アプローチでは，これら特定のキーをもたない管理対象に属するデータ項目は従属すべきキーが不明確なので，整理されないまま放置されるか，誤ってほかのエンティティに混入されてしまう可能性が高くなる．

(4) Ｔ字形ＥＲでの帰属概念

Ｔ字形ＥＲでは，データ項目を管理対象ごとの集合に整理する場合に「帰属」概念による検証を行う．則ち，データ項目が，管理対象を説明しているか否かの確認を行うことで帰属関係を判断する．

例えば，管理対象「顧客」に関連するデータ項目の帰属を考えると，

氏名は，顧客そのものを説明しており，顧客そのもののデータ項目である．

住所は，その顧客が存在しなくても，独自に存在するので，別の管理対象であり，顧客とは，居住するという関係にある．

電話は，その顧客が存在しなくても，独自に存在するので，別の管理対象であり，顧客とは，所持（または加入）するという関係になる．

などである．

よって，一旦，従属関係として整理されたデータ項目，特にもとのデータ項目の集合にそのまま残っているデータ項目を，その管理対象の帰属性で検証することで，その時点で，認識されている管理対象に帰属しないデータ項目を明らかとすることが，隠された管理対象の存在を検討する手がかりとなるので，結果的に業務実態を詳細に表現する業務構造の作成を可能とする．

(5) 帰属から管理対象の想定

以上より，従属関係にないデータ項目については，それが本来帰属する管理対象を象徴する人工的な付与コードとの従属関係では整理出来ないので，直接，本来帰属する管理対象を想定することでデータ項目整理の完全性を補完する．

データ中心アプローチの説明では，この部分を称して「正規化技法などのボトムアップ技法では，トップダウンアプローチによる補完をする必要がある」と表現しており，まさに正規化技法だけでは解決困難な部分の存在を示唆している．

１．３　データベースと初期値

前節で，正規化のいう従属概念だけでは，整備しきれない部分が残ることを確認したが，従来，それらのデータ項目は，本来属すべきではない何れかの管理対象に属することになり，そのままの形で実装されると，データ発生時（データ登録時）データ項目はあるが値は入れられない状態になり，いわゆる「初期値」の設定が必要となる．

逆に，データベースを設計する段階で初期値設定を前提にしていることが分かれば，本来その管理対象に属すべきデータ項目ではない可能性が高いので，新規管理対象の設定を考慮する必要がある．

以上，データベースでは，安易な初期値設定は，設計未了の指標となるが，１．１節で確認したデータベースに関する本質的な理解（データベースは，業務の実態から必然として導き出されるもので，データレイアウトのように恣意的に作られるものではない）がないと見過ごすことになる．

１．４　データベースとユーザインターフェース

データベースとユーザインターフェースの関係は，１：１である．

即ち，ユーザインターフェース相互の関連はゼロとなり，保守作業に於いて，ほかとの関連を考慮する必要はなくなり，高い保守性を継続的に維持することが可能となる．

ただし，以上は概念的なユーザインターフェースとしての関連であり，物理的には，複数のユーザインターフェースに分かれる事態が発生する．

それは「顧客に関する入力データ項目が多いので，勤務先関係のデータ入力は別画面から入れる仕様にする」と同様，単にユーザインターフェースの作りの問題であり，物理的に如何なる仕様となるかが，概念的なユーザインターフェースとデータベースの関係を変えるものではない．**図１**に，データベースとユーザインターフェースの関係を説明する．

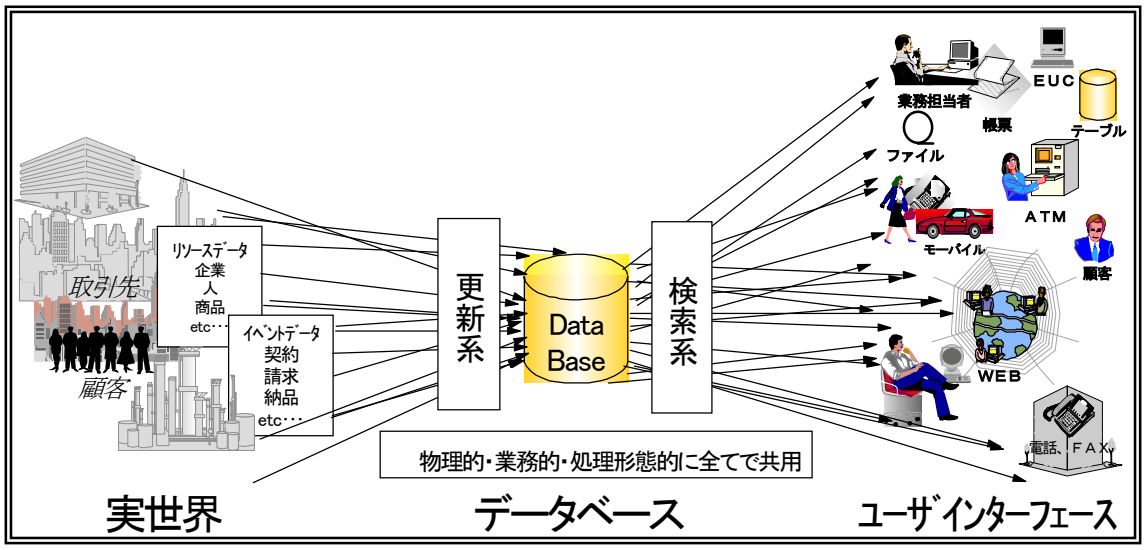

図１　データベースとユーザインターフェースの関係

また，基本的に「中間ファイル」と呼ばれるデータは，データベースからユーザインターフェースを実現する経路で，プログラムを複数に分割した場合に，そのプログラム間でデータを引き継ぐ目的に限定することで，データベースの拡張（データベース以外の部分でデータの共用が発生することで，実質的にデータベース化すること）を防止する．

１．５　データベースとプログラム仕様

前項より，全データ項目がデータベースに一元化されるので，入力されたデータ項目は，すべてデータベースに収斂し，出力は，すべての導出・編集をデータベースから取得したデータ項目で行う．

よって，当然ながらプログラム別の個別仕様でデータ項目の収容先や取得元を，一々記述する必要はなくなる．

また，データ中心アプローチでは，データに関する各種ルール（導出項目の導出ルールと入力項目のチェックルール）は，データ項目単位で定義し管理するので，これもプログラム単位の記述は必要ない．

結果的に，プログラム仕様として残るのは，ユーザインターフェース自体のデザイン（これも管理の仕方で，プログラム単位でない場合も考えられる）と，それが物理的に分割された場合の関連程度なので，文章で表現せざるを得ないあいまいな記述が残る余地は少ない．

２．導出項目の整理

導出項目の扱いに関するデータ中心アプローチの一般的対応は「入力データでの導出可能性の確認」までで，導出項目の基本は，非実装である．

しかしながら，一般的な入力項目数とデータベースの実装項目数，及び，最終的にユーザインターフェース上で使われる総データ項目数の比率を考えると，総データ件数の１～２％，データベースの実装項目数と較べても，せいぜい１割程度しかない入力項目のみを実装し，あとはすべて利用時に一々導出計算をするというのは現実的ではない．

また，ある程度の実装を容認する技法でも，その判断基準があいまいである．

データ中心アプローチで安定したデータ構造を確立するためには，この導出項目の扱いについて明確に規定する必要がある．**表１**にデータ項目の実装／非実装の判断要素となる不安定レベルについて説明する．

表１　データ項目の不安定レベル

不安定レベル	位置づけ	共用/実装	概要
Level1	Fact	共用/実装	・入力項目
Level2	準 Fact	共用/実装	・Level1/Level2ﾃﾞｰﾀから導出 ・時期･ﾀｲﾐﾝｸﾞの影響を受けない
Level3	凍結 View	共用/実装	・Level4 の導出データから，週間/月間などの期間的範囲指定で値を凍結したﾃﾞｰﾀ ・時期･ﾀｲﾐﾝｸﾞの影響を受けない
Level4	View	非共用/非実装	・導出ﾀｲﾐﾝｸﾞで値の変わるﾃﾞｰﾀ ・相対的時期(前月など)を修飾語に持つ項目

(1) ファクトデータ

Level1 のデータ項目は，データ項目としての不安定性がもっとも低い（安定している）ファクトデータである．基本的にファクトデータはすべて入力データ項目である．

(2) 準ファクトデータ

Level2 のデータ項目は，導出項目ではあるが再計算可能なデータ項目であり，扱いもファクトデータに準ずるので準ファクトデータと命名する．

例えば，消費税率５％の時期に契約された１００万円の取引に掛かる消費税は５万円である．導出データではあるが，何度再計算しても５万円であるということは，逆に５万円の値を保持して共用すれば，再計算の必要はなくなる．

ここに，プログラムロジックとデータ項目は置き換え可能となる．

則ち，導出ロジックは，導出データ項目を導出するものなので，導出データ項目自体を実装することになれば，導出ロジックは入力時に１度だけ処理することになり，当該データ項目を実装することによって，そのデータ項目を利用するユーザインターフェースの実現プログラムから，それら導出ロジックを排除することが可能になる．

プログラムは，単純な編集機能のみになるので，プログラムの生産性は向上する．

よって，準ファクトデータについては，利用の多寡に関わらず共用対象としてそのすべてを実装することで，出力・検索側でプログラムステップを抑制する．

(3) 凍結 View

Level3 のデータ項目は，本来時間経過とともに変化するデータ項目を，過去の期間で特定することで値を凍結するものである．例えば，月間販売台数なるデータ項目を想定する．

３月の「月間販売台数」は３月の間は販売契約の都度，販売台数を加算する必要があるので，値は刻々と変化するが，４月になれば（月が変われば）３月の「月間販売台数」は変化することはない．

月間合計自体は，日々の売上を販売量として総合的に見ようとする View であり，それを更にある視点で値を固定化しようとするので，将来的視点の変化には堪えられない．前の企画部長は，商品別に施策を展開していたので，商品別の集計を希望していたが，部長が異動し，今の部長は，商品別よりは，地域別に施策を展開しようとすれば，当然新しい視点での集計を求められる．

よって，ファクト，準ファクトのデータ項目とは異なり，全体での共用は考慮を要するが，導出の負荷を考えると，情報系データベース・データウェアハウス・データマートなどの構成部品としては，実装せざるを得ない．

(4) View

Level4 のデータ項目は，時間経過で値が変化する可能性を持つ不安定なデータ項目なので，使用時に導出計算を行うことで，原則的に実装は禁止である．

ただし，View 項目にも，準ファクトデータと同様，導出ロジックとデータ項目は置き換え可能の原理は働くので，量的な考慮が必要となる．

則ち，当該データ項目の更新回数と利用回数を比較し，**表２**に説明する対応に収斂する．

表２　View 項目の更新回数と利用回数の相対的比較による対応方針

対応方針	更新回数	利用回数	概要
非実装	大	少	・利用の都度導出計算を行う． ・但し，導出に使うデータ件数が多い場合は，値を実装し，それを中間値として計算する導出ルールで，導出負荷を軽減する．
実装	少	大	・値を実装することで，利用時に発生する計算処理重複を回避する．
実装	少	少	・計算負荷は軽微なので，実装する．
実装	大	大	・通常の導出ルールで導出した値を実装し，その値を中間値として使う導出ルールで，計算量を少なくし更新時の導出計算負荷を少なくする．

以上，基本的には導出対象データ入力時も，ユーザインターフェース上での利用時も，データ導出ルールは同じなので，総導出回数の極小化を図る．

３．実装基準

３．１　データ項目の不安定レベルと実装上の位置づけ

不安定レベル別の，実装上の位置づけを述べる．

表３に，不安定レベルと実装上の位置づけの関係を，**図２**に，データベースを中心とするシステム構造を説明する．

表３　データ項目不安定レベルと実装上の位置づけ

不安定レベル	位置づけ	共用/実装	実装上の位置づけ
Level1	Fact	共用/実装	・データベースとして実装
Level2	準 Fact	共用/実装	・データベースとして実装
Level3	凍結 View	共用/実装	・共用 View テーブルとして実装
Level4	View	非共用/非実装	・View データ項目（中間値）として単独項目でテーブルを実装 ・但し，同じ位置づけの View が存在する場合（当日販売個数と当日販売金額などのように導出対象範囲や導出タイミングが同じ場合）は，同じテーブルに実装可.

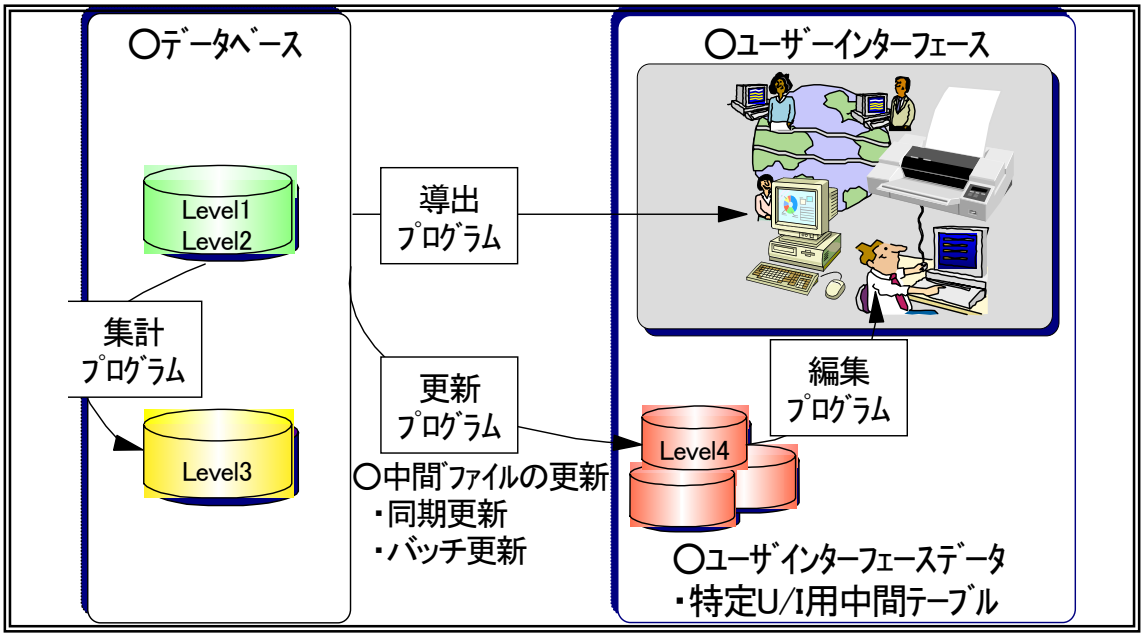

図２　データ（ベース）中心のシステム構造

３．２　データの物理構造

表３及び**図２**から，データベースは，不安定レベル１・２のデータ項目群だけを正規化することで作る．

よって，この部分では基本的に実世界の実体の変化を反映する場合以外には，データの追加はあっても更新はないため，物理的にも極めて安定した形を長期間維持できる．

レベル３のデータ項目は，View ではあるが，その視点（例えば，業績評価の指標はすべて，事業別・商品別で行うとか，個人の競争心こそが企業の活力源なので，すべては担当者別に視るなど）が継続する期間は，社内の基準を統一するためにも，一元化・共用化の対象とすべきであり，準データベースと位置づける．

この部分も，過去の期間集計値が中心になるので，一定期間の業務終了後に導出処理を行い，その都度新しいデータの追加を行うが更新は排除可能である．

但し，従来，頻繁に行われていた「前月～」「前々月～」方式の相対的時期表現を用いると，データ項目としては更新の必要がなくても，データの組み合わせとして更新が必要となるので，データの組み合わせについても，不用意に更新を発生させることのないように留意する必要がある．

レベル４のデータ項目は，基本的には実装しないが，**表２**の判断基準にしたがって，実装する場合にも共用は許さない．

ファイル・テーブルの形態でも，データベースではなく，ユーザインターフェースと位置づけることで，データベースの安定性を損なう危険性を回避する．

データベースは概念的存在だが，ＤＢＭＳ管理下のデータがデータベースであると理解していると，この「実装せざるを得ないレベル４のデータ項目」をデータベースの項目に加え，データベース全体を定期的・頻繁な更新に晒す結果となる．

レベル４のデータ項目は実装の対象とはなっても，それは，総導出計算負荷最小化が目的であり共用の対象ではない．

３．３　プログラム構造

図２で説明するように，データベースの安定性を最大とするデータ構造で実装すると，プログラムは概念的に４つの位置づけに収斂することができるので，パターン化などでステップ数が削減できる可能性がある．

第一は，基本となる「導出プログラム」で，これは，入力ユーザインターフェースを実現するものと，出力ユーザインターフェースを実現するものに分かれる．

入力側では，入力されたデータ項目をチェックし・実装する導出項目の導出計算をして，データベースに書き込む．

出力側は，必要なデータ項目をデータベースで指定・導出してユーザインターフェース別の編集仕様に沿って編集して表示する．

基本的には，この形態で，すべてのユーザインターフェースに対応可能である．

しかし，当該形態でレスポンスが悪い場合は，特定のユーザインターフェースを早く実現する中間データが必要となる．

その場合は，中間データを作る「更新プログラム」と，この中間データから要求に対応してユーザインターフェースを実現する「編集プログラム」に機能が分離する．

よって，第二の「更新プログラム」は，基本的に「導出プログラム」と部分的に同じ処理を行う．

但し，処理のタイミングが「導出プログラム」は入力・検索要求があった場合だが，「更新プログラム」は更新要件（「新しいトランザクションが発生した」など）が発生する都度行う「同期更新」か，一定の処理対象を定期的に処理する（毎日・月末など）「定期更新」となる．

また，第三の「編集プログラム」も「更新プログラム」同様「導出プログラム」の編集機能と同じ処理を行うが，データをデータベースからのみではなく，専用の中間ファイルからも取得すると同時に，中間ファイルでは，導出負荷の高い導出項目も実装しておくので，ユーザインターフェースの実現時間を短縮する．

以上，ソフトで対応するレスポンス向上は，システム構造の複雑さを増す上，直接ユーザインターフェースを実現するプログラムの比率を下げることになる．

つまり，「導出プログラム」と「更新プログラム・編集プログラム」の組み合わせはトレードオフの関係にあるが，安易に「更新プログラム・編集プログラム」の組み合わせでレスポンスを上げようとすると処理がピークを形成し，それに合わせた機械・設備が必要となるので全体の効率低下を招く．

よって，ある程度ハードウェアーを増強しても「導出プログラム」での処理平準化が運用上も簡明である．

第四の「集計プログラム」は，データベースから特定の View で集計期間単位に集計値を導出するもので，レベル１・２のデータ項目から，レベル３を導出する．

おわりに

規模の肥大化は，システムにとってもっとも困難な課題であるが，データ中心アプローチは，その課題に対する数少ない答えの一つであると考える．

しかしながら，その中核技術である「正規化技法」が，プログラム作りを中心とする思想・体制の中では理解され難く，作法・形式として形骸化しているのは実に残念だ．

また，データ中心アプローチでは，概念と実装（論理と物理・業務とシステムも同様）の使い分けが必要なので，プログラムを作ることを目的と考えると迂遠な印象を与えることも否めない．

しかし，時代が業務系・分析系を問わず統合一元化の流れを示している中で，肝心のデータが概念的な統合デザインを持たず，システムのデータベース（データ基盤）となっていなければ，ＢＰＲ・ＳＣＭ・ＥＲＰなどの新しい概念の適用も，いたずらに重複を繰り返し肥大化し，動きが鈍くなったシステムが逆に経営の枷となるのは明らかである．

必要なことは，学問的な証明や論理の整合性を主張することではなく，現場で如何に考えどう行動するかの指針である．

この論文でまとめた内容は，実務的データ中心アプローチとしてのデータベース設計現場での指導要領である．

今後もシステム開発は「データ中心アプローチが常識」となるように，作業内容・手順・技法・思想の明確化とプレゼンテーションのレベルアップを図りたい．

以上

参考文献

［１］佐藤　正美：Ｔ字形ＥＲデータベース設計技法，1998年10月25日，(株)ソフト・リサーチ・センター

［２］山谷　茂：現場主義のＲＤＢ設計，1995年11月10日，(株)リックテレコム