関係データベースに基づく パーソナルコンピュータ用 アプリケーション開発システムの動向

三　井　大三郎
佐　藤　清　美
石　丸　敬　二

目次

まえがき
I　パーソナルコンピュータ用アプリケーション開発システムの発展経緯
II　パーソナルコンピュータ用アプリケーション開発システムの構造
III　ビジュアル化，オブジェクト化
IV　イベント駆動型プログラミング
V　テーブル単位の処理
VI　アプリケーション開発システムとしての支援範囲
まとめ
参考文献
SUMMARY

まえがき

パーソナルコンピュータやワークステーションの高性能化，低価格化により，従来の汎用機やオフコンを，パーソナルコンピュータ，ワークステーションで置き換えるダウンサイジングの傾向が定着してきている。特にパーソナルコン

ピュータの高性能化が著しく，パーソナルコンピュータを企業の業務に本格的に利用しようとする企業が増加している。そのため，応用システムを開発するためのプログラミング言語や開発支援システムに対する要求が強まり，各社から応用システム開発支援のためのアプリケーション開発システムが提供されるようになった。

これらのアプリケーション開発システムはいずれも関係データベースを中心とし，画面上の図形によるグラフィカル・ユーザー・インターフェースを採用し，言語もＣＯＢＯＬ等の第３世代言語ではなく，オブジェクト指向の独自の言語を装備している。

これらのアプリケーション開発システムによる開発は第３世代言語の場合とは様相を異にする。これらのアプリケーション開発システムが応用システムの開発の効率化にどのように影響するか。また，その要因は何かという視点から最近のパーソナルコンピュータ用アプリケーション開発システムの動向を分析した。

Ⅰ　パーソナルコンピュータ用アプリケーション開発システムの発展経緯

1　企業における情報システムの処理形態の変遷

1950年頃に始まった企業におけるコンピュータの利用は汎用機を中心に発展を続け，その後データ通信と結びつき，データを中央に集めて処理する集中型の大規模なオンライン企業情報システムに発展した。

75年頃からミニコンピュータやワークステーションが開発され，中央コンピュータの負荷を軽減し，また，処理をなるべく現場に近い所で行うという思想に基づきミニコンピュータやワークステーションを用いた分散処理による部門コンピューティングが行われるようになった。

さらに80年頃からは，パーソナルコンピュータが普及しはじめ，企業でも各職場の個人の仕事であるデスクワークに使われるようになった。そこで主に使われたのは，ワードプロセッサーや表計算ソフトウエアであった。

職場内や事務所，工場内など一定の地域内のコンピュータを結合するＬＡＮ（Local Area Network）が普及すると，プリンターやファイルを複数のパソコンから利用できるようにするためこれらをＬＡＮで結んだパソコンネットワークが構成されるようになった。

マイクロプロセッサーやメモリー，ディスクの高性能化と低価格化が急速に進んだ結果，これまで高価な汎用機で行われてきた企業の基幹業務を，ワークステーション，パソコンあるいは一部汎用機も組み合わせた，より低価格のシステムで実施できないかという考え方が生じてきた。

データベースは汎用機，ワークステーション，高性能パソコンで集中管理しこれに各職場に置かれた多数のパソコンをＬＡＮで結合したもので，データを提供する側のコンピュータをサーバー，受け取ったデータでそれぞれの業務を処理する側をクライアントと呼び，こうした構成のシステムをクライアント／サーバー・システム（Ｃ／Ｓシステム）と呼んでいる。

Ｃ／Ｓシステムは単にコスト削減を実現するだけのものではなく，これまで情報システムを専門に担当する部門（情報システム部等）に集中しがちであった企業情報を全社的に共有することが可能となり，場合によっては企業の組織や戦略にも影響を与えるものである。Ｃ／Ｓシステムの１つの特長は業務担当者（エンドユーザー）の使用するコンピュータが操作性のよいパソコンという点である。パソコンは個人使用から始まったという事情から操作性を重視して開発が進められてきた。

2　データベースソフトウエアの発展

企業のコンピュータ利用は給与計算や売上管理など個別業務をコンピュータ化する事から始まったため，処理プログラムとデータは一体として扱われ，デー

タは処理に最も適した形のファイルとして構成された。コンピュータ化が全社的な情報システムに発展するにつれて各ファイルのデータ間に重複や矛盾が生ずるようになりデータの集中一括管理が要求されるようになり，これに応えるものとしてデータベースシステムが開発され，情報システムの中で重要な位置を占めるようになった。当初のデータベースシステムは，階層型やネットワーク型で構造が複雑であり，その構築や運営には高度の専門知識が必要であった。

こうした中で，ＩＢＭのE.F.Coddにより提唱されたリレーショナル（関係）データベースは，２次元のテーブル形式を基礎とし，データ操作は関係代数に基づくもので，“一般ユーザー向けの簡易言語によるユーザーインターフェースを作りやすいこと”，“理論的な明快さ”を特長とするものであった。当初は実用化が難しく普及には時間がかかったが，コンピュータの性能，ソフトウエア技術の向上に伴い主力データベースシステムに発展していった。

ワークステーション，パソコンがビジネス処理に利用されるようになり，汎用コンピュータ向けに開発されたデータベースシステムがワークステーションやパソコンに移植され，また新たに開発された。関係データベースの上記の特長から，汎用機から移植されたデータベースはその殆どが関係データベースであった。パーソナルコンピュータが使われる用途は２つあり，第１は名刺管理等に代表される個人利用であり，他は企業の業務処理である。第１の場合にはカード型データベースが，後者には関係データベースが多く使用される。本論文では関係データベースを対象とする。

企業における業務の形態には定型業務と非定型業務とがある。

＊　非定型業務：売り上げデータから売れ筋商品を分析するなどの試行錯誤を伴う分析，計画立案業務

＊　定型業務：販売管理，生産管理などで，企業によりほぼ定まった処理手順を持っており，大量のデータを効率的に，また高信頼度で処理することが要求される。

非定型業務に利用する場合は，業務の担当者がデータベース管理システムの提供する会話形式の簡易言語を使って順次操作を行い，必要な情報を取り出すことができる。定型業務の場合は担当者が必要な操作を会話処理により実行することは不可であり，何らかのプログラムによりコンピュータによる自動処理が必要となる。汎用機の場合はデータベースが出現するまでにビジネス処理用のプログラミング言語ＣＯＢＯＬが確立されており，ＣＯＢＯＬを親言語としてデータベース言語を呼び出す形がとられた。パーソナルコンピュータの場合は，ビジネス用プログラミング言語が確立されていなかったためデータベース管理システム自身が単にデータベース専用言語のみでなく一般のビジネス用プログラミング言語の役割も果たさなければならなかった。このようにパーソナルコンピュータの場合はデータベース管理システムが１つの完結した開発環境を提供するアプリケーション開発支援システムとしての性格を持つこととなった。

ビジネス用に利用されるパーソナルコンピュータはＣ／Ｓの構成をとることが多いため，データベース管理システムもサーバー用とクライアント用がある。クライアント用は小規模システムの場合は単独で即ちスタンドアロンとしても使用される。

以下で取り上げる特徴はほぼこの両者に適用されるが，主としてスタンドアロン用のものについて述べ，サーバー用については後日に譲りたい。

3　パーソナルコンピュータ用アプリケーション開発システムの種類とその適用範囲

パーソナルコンピュータ用アプリケーション開発システム（以後ＰＣ用ＡＰＰ開発システムという）には３種類ある。言語形式のもの，データベース管理システムに基づくもの，表計算ソフトに基づくものである。言語形式のものは適用範囲も広く融通性も高いが開発工数は大きくなる。データベース管理システムはサーバー用とクライアント用があり，これらを組み合わせてＣ／Ｓシス

テムを構築する場合が多い。小規模システムの場合はクライアント用がスタンドアロンで使用されることもある。表計算ソフトは単純なシステムの場合はスタンドアロンで使用されることもあるが，通常はサーバー用データベース管理システムと組み合わせC／Sシステムとして使用される場合が多い。

4　PC用APP開発システムの特徴

PC用APP開発システムの特徴は，次のようにまとめることが出来る。

(1) グラフィカルユーザーインターフェース（GUI）

ユーザーはキーボードからコマンドを入力するという形ではなく，ディスプレイ画面に配置されたフォームやテキストボックスをマウスでクリックするなど図形要素を通じてシステムと会話する。

(2) オブジェクト化

これらの図形要素にはデータ入力，演算，処理機能が割り当てられている。即ちデータと機能を一体化した目に見えるオブジェクトとして構成されている。このオブジェクトは手続型言語のプログラムに当たり，手続型言語によるプログラムより記述がはるかに容易になっている。

(3) 標準機能の提供

下記①②のように応用ソフトウエアで共通に使われる機能は会話形式のコマンドとして，あるいは標準のオブジェクトとして，またはシステム内蔵の機能としてPC用APP開発システムにより提供され，ユーザーのプログラミングの負荷を軽減している。

① データ入力やテーブルの作成などが会話形式コマンドで提供され，プログラミングが不要である。

② イベント検知と対応プログラムへの分岐機能がシステムに内蔵されており，ユーザーがプログラムを作る必要がない。

(4) 非手続言語の使用

データの抽出，集計，更新等のデータ操作の殆どが，非手続言語である関係

データベースのデータ操作言語で記述される。

(5) 応用ソフトウエアのユーザーインターフェース

ＰＣ用ＡＰＰ開発システムによって作り出された応用ソフトウエアが，元の開発システムと同じ形のユーザーインターフェースを持つことが出来る。

(6) 総合開発環境

上記の機能がそれぞれ単独で提供されるのではなく，それらが有機的に統合された１つの総合開発環境として提供される。

このようにＰＣ用ＡＰＰ開発システムは，オブジェクト化，非手続言語の採用，標準機能の提供によってユーザーが作成しなければならないプログラムの量を大幅に減少し，またＧＵＩによりユーザーの操作性を向上している。

II　パーソナルコンピュータ用アプリケーション開発システムの構造

1　ＡＰＰ開発システムによって作り出された情報システムの構造

一般に情報システムは次の構造を持っている。

(1) 処理の対象

① 対象業務の処理に使われるデータ

テーブルに記憶されているもの，または，それから派生するもの

② インターフェース要素

システムとユーザー間の意志疎通のための情報

＊入力画面のフォーム，プロンプト，ヘルプ，メニュー，プルダウンメニュー，およびリストボックスの表示等

＊出力画面のフォーム

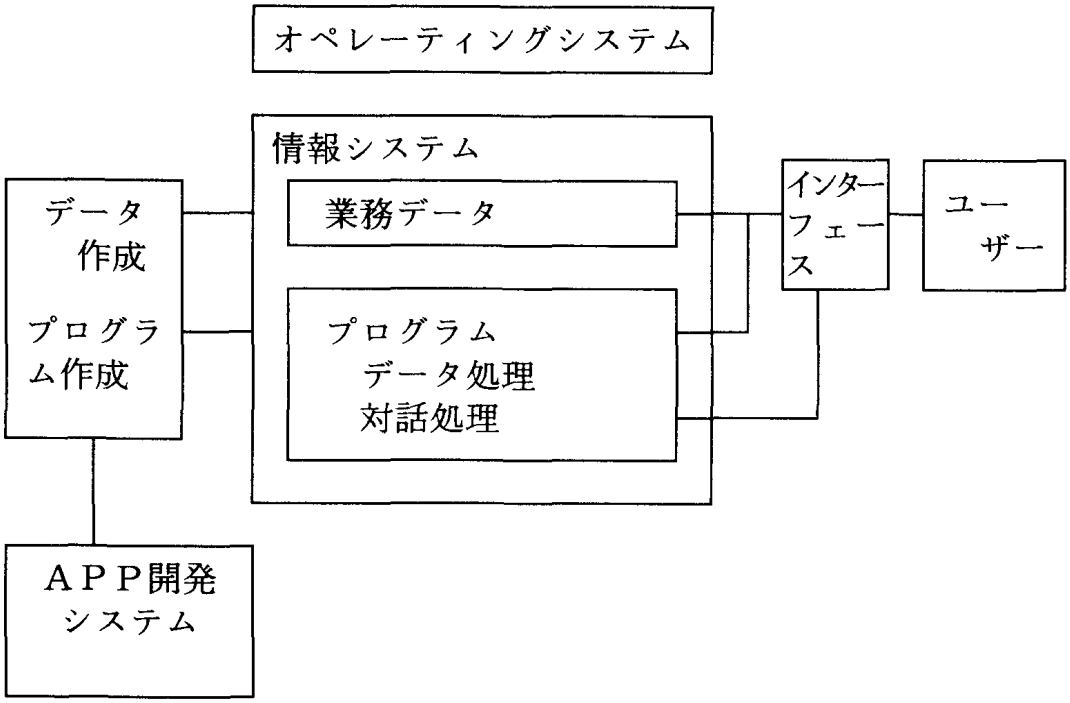
＊報告書（印刷されたもの）のフォーム

インターフェース要素はプログラム内やシステムファイル内に保存される。

(2) 処理

対象に対し何らかの変化をもたらす行為である。

情報システムの構造と，情報システム，ユーザー，ユーザーインターフェース，さらに情報システムを開発するためのアプリケーション開発システムの関連を図II－1に示す。



図II－1　情報システムの構造

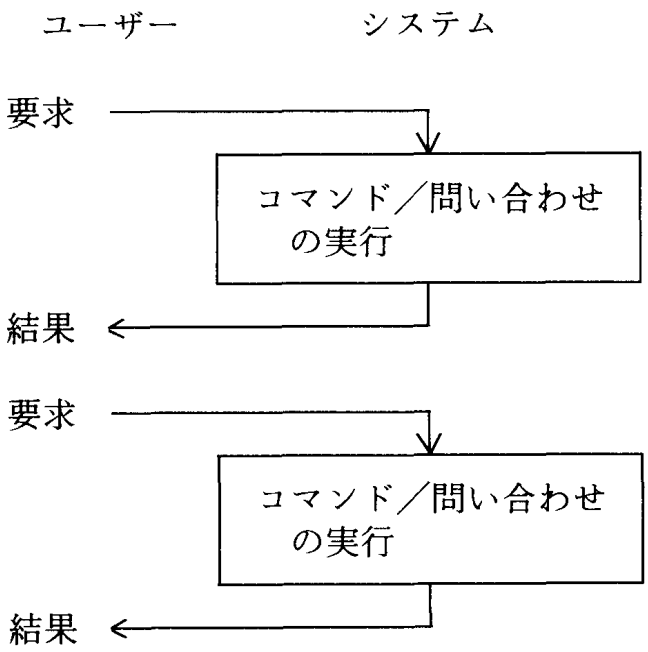
2　APP開発システムの構造

最近の殆どのAPP開発システムは2つの処理形式を持っている。1つは，対話形式であり，他はプログラム形式である。

(1) 対話形式

対話形式により，テーブル，問い合わせ，フォーム，およびレポート等のオブジェクトの作成と，実行が可能で，簡単な業務処理システムの作成は一切プログラムを作らなくても可能である。即ち，処理対象であるデータとフォーム等のインターフェース要素を対話処理で作成すれば，処理機能はコマンドとし

て提供されているのでこれにより業務処理が可能となる。“問い合わせ”はテーブル操作を行うもので処理としての性格とその結果であるテーブルとしての性格を併せ持つものと考えることが出来る。このように考えれば“問い合わせ”を作るのはプログラミングを行うことになるが，これも対話形式により作成可能である。対話形式ではユーザーの要求により１コマンドずつシステムの制御下で実行が行われる。

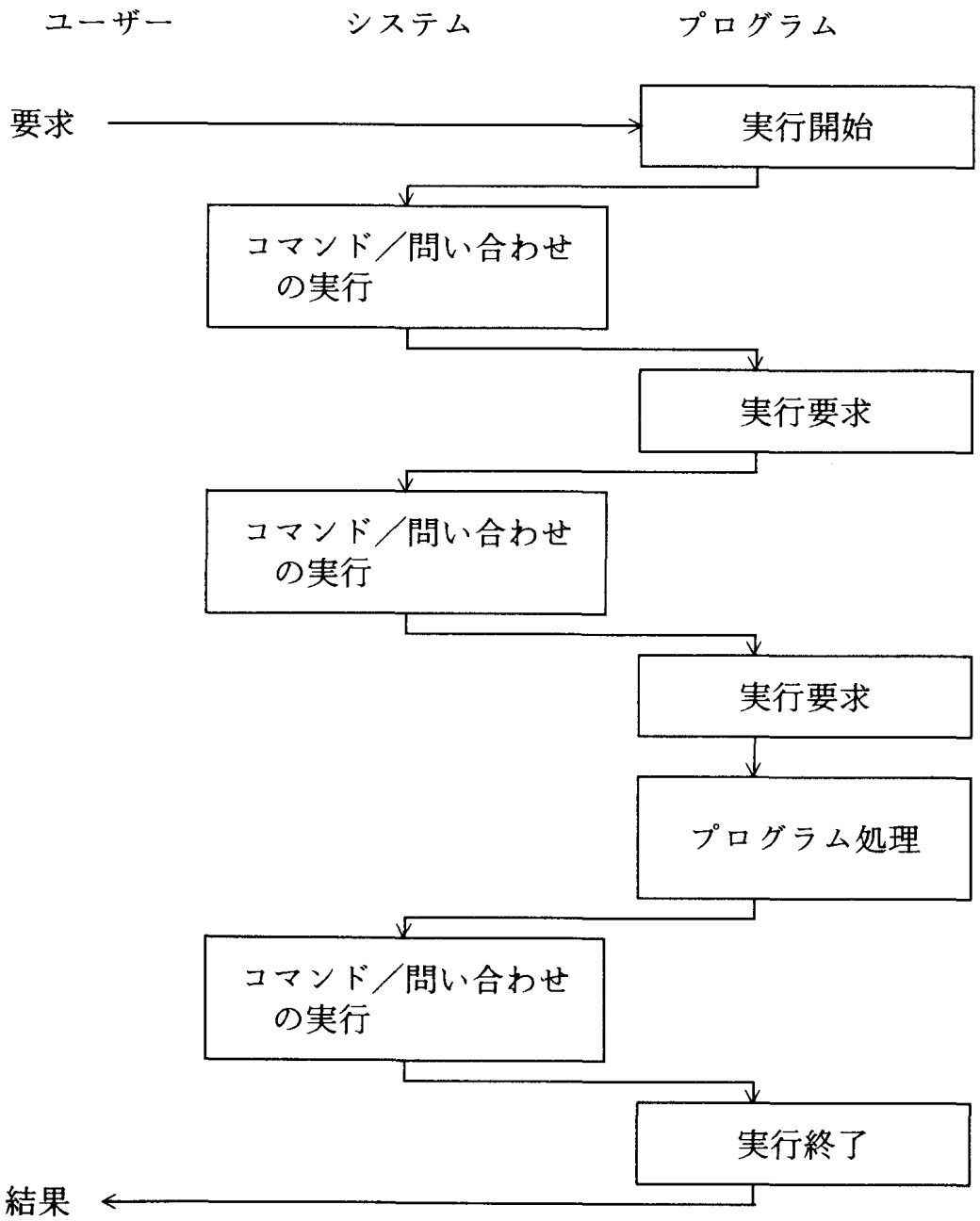


図II－２　対話形式の実行制御

(2) プログラム形式

適用業務の１つの論理処理単位が１つのコマンドや問い合わせで処理できることは少なく，通常は，これらを複数個続けて一定の順序で実行しなければならない。これを行うのがプログラム形式である。ＡＰＰ開発システムはコマンドとは独立したプログラミング言語を持つものが多い。この言語は一般の言語と同様に，データ定義やデータ処理，実行制御等の機能の他に対話形式で定義されたオブジェクトを取り扱うためのデータ型と，そのデータ型に属するメソッ

ドを持っている。メソッドは言語の中でメニューコマンドと同様の機能を持つ。このデータ型とメソッドを使って予めプログラムされた順序でコマンドや問い合わせを実行することが出来る。また，言語は対話形式では実行が困難なレコード単位処理の機能を持ち，これらの処理はプログラム形式で実施される場合が多い。



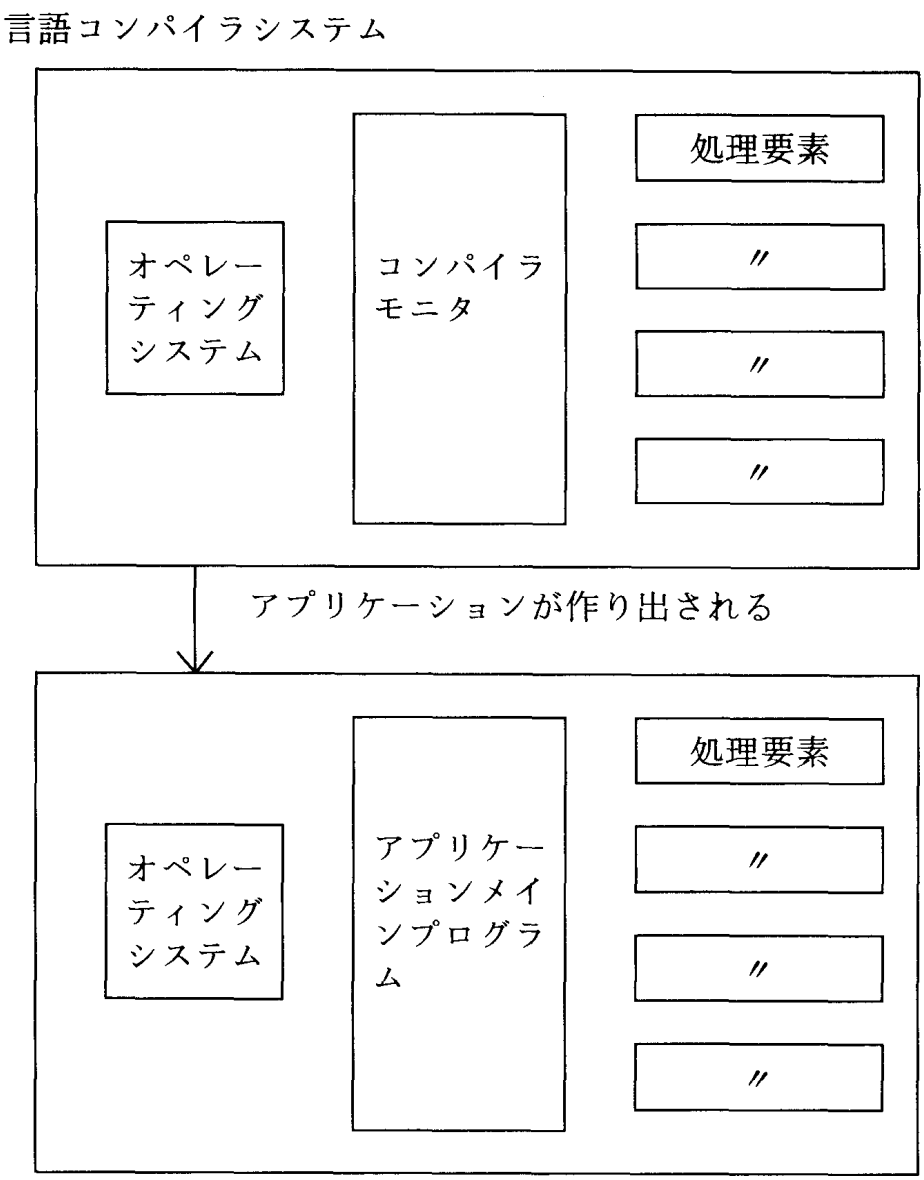
図II－3　プログラム形式の実行制御

(3) アプリケーション

業務を処理する一連の応用ソフトウエアのシステムをアプリケーションと呼ぶ。

① 手続き言語型アプリケーション

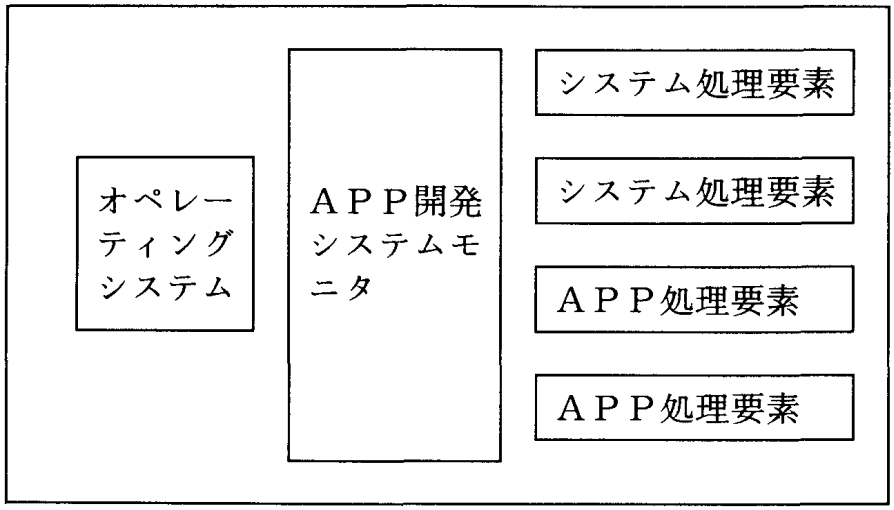
この場合は，アプリケーションは言語システムによって作り出され，言語システムとは独立のものとして稼働する。



図II－4　言語コンパイラシステム

② ＰＣ用ＡＰＰ開発システム型のアプリケーション

この場合はＰＣ用ＡＰＰ開発システム自身の中にアプリケーション用のオブジェクトや，コマンド処理要素，プロシージャ等が組み込まれ，これらの要素が開発システム自身の一部の要素と連携して稼働し，手続型言語の場合とは，かなり様相が異なる。このような構造をとることにより，開発システム自身のユーザーインターフェースと同様の構造を持つ，アプリケーションを作り出すことが可能となる。



図Ⅱ－５　ＰＣ用ＡＰＰ開発システム

③ ＰＣ用ＡＰＰ開発システム型の一つの解釈

ＰＣ用ＡＰＰ開発システム型の場合は次のような解釈も可能である。メニューコマンドやオブジェクトの機能を命令として持つ仮想計算機を考えれば，メニューコマンドによる対話処理は，仮想計算機の操作卓からの計算機の運転に当たり，プログラム形式は仮想計算機のストアードプログラムに当たる。このようにＰＣ用ＡＰＰ開発システム型の命令体系は，マクロ命令とマクロのストアードプログラムの２階層の体系と考えられる。

III　ビジュアル化，オブジェクト化

次にＰＣ用ＡＰＰ開発システムにおいてビジュアル化，オブジェクト化がどのように行われ，それが，アプリケーション開発効率化にどのように貢献しているかを見ていく。

1　ファイルへのデータ入力の検討

(1) データ入力の論理構造

ビジネスデータ処理の基本機能であるファイルへのデータ入力において，ビジュアル化がどのように行われているかを見る。ファイルへのデータ入力の基本的な論理構造は，以下のようにまとめることが出来る。

① 入力装置（D），入力プログラム（P）を通して，ファイル（F）へデータを入力する。

② Fは項目F-1，・・，F-i，・・，F-nよりなる。

③ Pに対して，Dの種類，装置番号等と，Fの名前，ディスク等物理的なアドレス等を指定する。

④ Dを通して，項目F-iの項目名FN-iと項目の値X-iの組

{　FN-i，X-i　}

をPに与えることにより，Pは項目F-iに値X-iを記憶する。

⑤ 入力順序を予め決めておくことにより，F-iの代わりに入力順序iを用いることもできる。

⑥ また，ディスプレイ画面から入力する場合は，画面上の位置を識別子（項目名）の代わりに使うこともできる。

(2) 1個のデータを入力する場合

手続き型言語による場合と，ＡＰＰ開発システムの場合を比較する。手続き型の場合のプログラムは，たとえば図III－1のようになる。このプログラムを

見ると

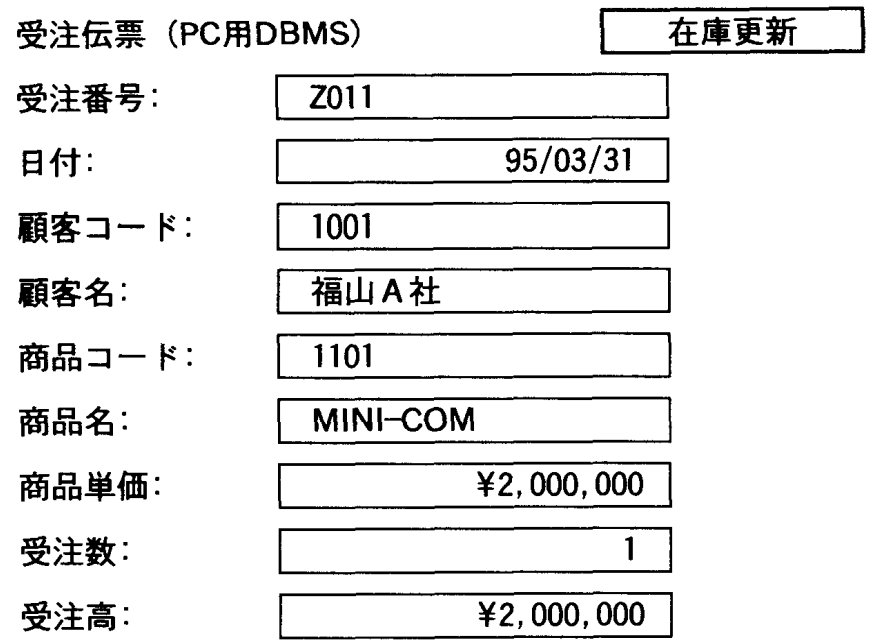
① ファイル名，項目名，項目名の画面上の表示位置，およびデータの入力位置等に関する情報がプログラム上に散在している。

② これら散在している情報間の相互関係を正しく維持しなければならない。たとえば，入力される値と項目名を対応づけるため項目名の表示座標と入力座標を正しく設定するなど詳細な注意が必要である。

```
 IDENTIFICATION  DIVISION.
*  受注ファイル入力プログラム（PC用DBMS）
 PROGRAM-ID.        DENT-02.
 ENVIRONMENT        DIVISION.
 CONFIGURATION      SECTION.
 SPECIAL-NAMES.     CONSOLE IS CRT.
 INPUT-OUTPUT       SECTION.
 FILE-CONTROL.
     SELECT  JYUT-F  ASSIGN TO  "V:JYUT-F.DAT".
 DATA               DIVISION.
 FILE               SECTION.
 FD  JYUT-F.
 01  JYUT-F-R.
     02  UD-HIZUKE          PIC 9(6).
     02  UD-DENPYO-NO       PIC X(4).
     02  UD-KYAKU-KODO      PIC X(4).
     02  UD-SYOHIN-KODO     PIC X(4).
     02  UD-URIAGE-SU       PIC 9(4).
 WORKING-STORAGE SECTION.
 01  IH-GAMEN.
     02  FILLER  PIC X(80).
     02  HYOJI1  PIC X(80)  VALUE
   "     受注ファイル入力プログラム".
     02  FILLER  PIC X(80).
     02  HYOJI2  PIC X(80)  VALUE
   "     日付            [      ]".
     02  HYOJI3  PIC X(80)  VALUE
   "     伝票番号        [    ]".
     02  HYOJI4  PIC X(80)  VALUE
   "     顧客コード      [    ]".
     02  HYOJI6  PIC X(80)  VALUE
   "     商品コード      [    ]".
     02  HYOJI8  PIC X(80)  VALUE
   "     数量            [    ]".
```

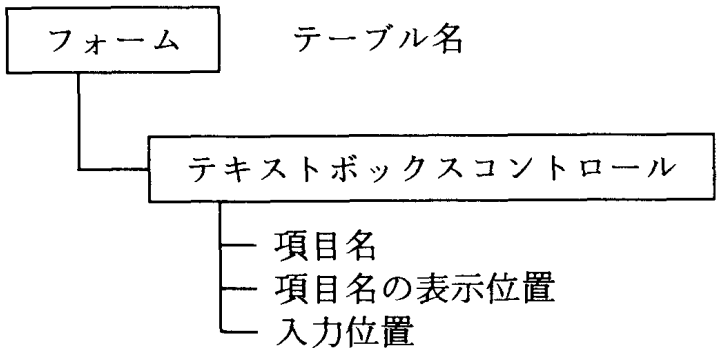
```
    02  FILLER   PIC X(1040).
    02  HYOJI15 PIC X(80)   VALUE
  "    伝票終了        [ ]".
01  IN-RYOIKI.
    02  IN-HIZUKE          PIC 9(6).
    02  IN-DENPYO-NO       PIC X(4).
    02  IN-KYAKU-KODO      PIC X(4).
    02  IN-SYOHIN-KODO     PIC X(4).
    02  IN-URIAGE-SU       PIC 9(4).
01  DENPYO-SYURYO     PIC X.
01  ST             PIC X.
PROCEDURE      DIVISION.
MEINSYORI         SECTION.
    OPEN  EXTEND  JYUT-F.
    MOVE SPACE TO  DENPYO-SYURYO.
    PERFORM  UD-SYORI THRU UD-SYORI-EXIT
                UNTIL  DENPYO-SYURYO = "Y".
LA-1.
    CLOSE  JYUT-F.
    DISPLAY  SPACE.
    STOP  RUN.
UD-SYORI.
    DISPLAY  SPACE.
    MOVE  SPACE TO IN-RYOIKI.
    MOVE  SPACE TO JYUT-F-R.
    DISPLAY  IH-GAMEN.
    ACCEPT   IN-HIZUKE          AT 0421.
    ACCEPT   IN-DENPYO-NO       AT 0521.
    ACCEPT   IN-KYAKU-KODO      AT 0621.
    ACCEPT   IN-SYOHIN-KODO     AT 0721.
    ACCEPT   IN-URIAGE-SU       AT 0821.
*
    MOVE  IN-HIZUKE   TO  UD-HIZUKE.
    MOVE  IN-DENPYO-NO TO  UD-DENPYO-NO.
    MOVE  IN-KYAKU-KODO  TO  UD-KYAKU-KODO.
    MOVE  IN-SYOHIN-KODO  TO  UD-SYOHIN-KODO.
    MOVE  IN-URIAGE-SU    TO  UD-URIAGE-SU.
    WRITE  JYUT-F-R.
    ACCEPT   DENPYO-SYURYO    AT 2221.
UD-SYORI-EXIT.
    EXIT.
```

図Ⅲ－１　手続型言語によるプログラム



図Ⅲ－２　ＰＣ用ＡＰＰ開発システム　入力フォーム

これに対し，ＰＣ用ＡＰＰ開発システムでは，入力フォーム上にテーブルのその項目に連結したテキストボックスと呼ばれる入力枠を画面上に配置するという図形的な操作で行われる。



図Ⅲ－３　ＰＣ用ＡＰＰ開発システムにおける情報の集中化

フォーム上にテキストボックス等のコントロールを配置することは，手続型言語（例えばＣＯＢＯＬ）で入力画面を表示し（DISPLAY），データを入力

し（ACCEPT），それを出力領域に転記し（MOVE），さらにファイルに出力（WRITE）するプログラムを作ることに相当する。この両者を比較すると，手続型ではデータ入力の画面から主記憶内の領域へ，さらにディスク上のファイルへと物理的な移動の手順を詳細に指示している。ＰＣ用ＡＰＰ開発システムでは，フォームとそのレコードソースとなるテーブルを結合し，テーブルの項目に対応するテキストボックスをフォーム上に配置するなど論理的関連の操作（即ち非手続的操作）になっている。

2　ビジュアルプログラミングにおけるオブジェクトの取り扱い

テーブルの取り扱いを例に考える。

テーブル定義は設計画面において，決められた書式の中で項目名とデータの形式等をキーボードから入力していくことによりテーブルが定義され，次に入力画面に切り替えて，データを入力すればテーブルが完成する。

完成したテーブルはコンピュータ内に保存され対話画面上にいつでも表形式で表示出来る。物理的にどのような形で記憶されているかはユーザーは一切関わらない。ユーザーの理解としては，紙に書かれたテーブルがコンピュータ内にあり，それをいつでも取り出し，ペンで書き込むのと同じ感覚でキーボードから記入，消去が出来る。即ち，現実の世界で紙に書かれたテーブルが持つ物理的な属性がコンピュータの中で模擬実現されている。人が紙に書かれたテーブルを扱うときにこれらの属性を無意識のうちに前提としているが，ビジュアルプログラミングにおけるテーブルもこの前提が成り立つように設計されているため，ＣＯＢＯＬ等の手続型に比べ違和感の少ないものになっている。

このようにＰＣ用ＡＰＰ開発システムではソフトウエアの対象を画面上で視覚的（ビジュアル）な図形（表等も含む広い意味の図形）として表現し，これに必要な属性を結びつけ，その全体をオブジェクトとして捉えている。

Ⅳ　イベント駆動型プログラミング

1　イベント駆動型プログラミング言語と手続型プログラミング言語

イベント駆動型プログラミング言語とは，次の３つの機能を持つものである。

① イベントとして認識しうる事象の範囲の定義

通常，キー入力やマウスのボタン操作などが含まれる。

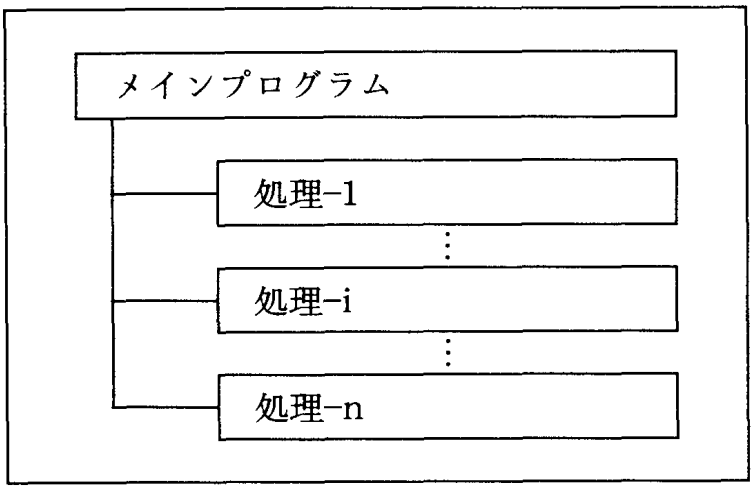
② イベントの発生を検知する機能

③ イベントに対応したプログラムに制御を渡す機構

これに対し，ＣＯＢＯＬのようにマウスのボタン操作等を検知する機能を持たない言語（手続型第３世代高位言語）を単に手続型言語と呼ぶことにする。

2　応用ソフトウエアの制御方式

一般に応用ソフトウエアは，図Ⅳ－１のように，幾つかの処理プログラムと全体を統括するメインプログラムから成る。



図Ⅳ－１

メインプログラムが処理-iに制御を渡す方式によって応用ソフトウエアは次の２つの型に分かれる。

① 順次制御型

処理-1，・・・，処理-i，・・・，処理-nの実行順序が予め決まっており，メインプログラムにより順序通り実行される場合

② イベント起動型

予め順序は決まっておらず，次に実行する処理はユーザーの選択によって決まる場合

応用ソフトウエアがイベント起動型であるか否かということと，応用ソフトウエアを記述する言語がイベント駆動型であるかということは別問題であり，イベント起動型の応用ソフトウエアを手続型言語で記述することも，ユーザーからの要求を受け付けるタイミングがある程度決まっていれば可能である。

トランザクションや情報要求など外部事象により起動される応用システムはイベント起動型である。ユーザーからの要求の受け入れは，入力画面を媒介とする会話により行われる。この会話の形式として，メニュー方式，フォーム記入方式，プロンプト方式などがあるが，いずれにせよ，文字のみによるユーザーインターフェース（ＣＵＩ）を採用するなら，従来の手続型プログラミング言語によってもイベント起動型応用ソフトウエアを構築することは可能である。しかし，ＧＵＩによるシステムを構築しようとすれば，イベント駆動型が不可欠である。

以後，イベント駆動型言語によるイベント起動型応用ソフトウエアの構築について検討するが，応用ソフトウエアにおいてユーザーとの会話が行われる代表的な場面は，起動する処理をメニュー画面から選択する場合と入力画面からデータを入力する場合である。以下ではこの２つの場合について検討する。

3　メニュー画面による処理の選択

ＰＣ用ＡＰＰ開発システムでは，いずれもイベント駆動型プログラミングを採用しているが，それがどのタイプのオペレーティングシステム（ＯＳ）の上に構築されているかによってイベント駆動の実現の方法が異なってくる。ＣＵ

Ｉ型のＯＳ上に構築されている場合をイベント駆動１型，ＧＵＩ型のＯＳ上に構築されている場合をイベント駆動２型と呼ぶことにする。

(1) イベント駆動１型

２型ではメインフォームが応用ソフトウエアのメインプログラムとメインメニューの役割を果たすが，１型では２型ほど高度のグラフィックス機能を持たないので，メニューにメインプログラムの機能を持たせることが出来ず，メインプログラムをユーザーが作成しなければならない。例えばBorland社のバージョン4.0Jまでの Paradox はこの型に属する。Paradox の場合メニューによる処理の選択は次のようになる。

ユーザーが作成するメインプログラムの中で，SHOWPULLDOWN 命令を実行してメニューを表示しGETMENUSELECTION 命令で選択された処理を検知し，次に SWITCH-CASE 命令によりそれぞれの処理を実行するプログラムに分岐する。

このように，ユーザーが１項の①，②，③の機能を使ってメインプログラムを作成する事によりイベント駆動を実現している。

(2) イベント駆動２型

Microsoft 社の Access は同社のＧＵＩ型ＯＳである Windows 上に構築されたもので，２型に属する。Access の場合は先ずメインフォームを作り，その上に各処理に対応するコントロールボタンを設定する。このボタンをマウスでクリックすればボタンに対応した処理が起動される。この場合メインフォームがメインプログラムの役割を果たし，イベントを検知しその種類に応じて分岐する機能は Access のシステム自身が持っている。

4　画面からのデータ入力

データ入力はユーザーが入力プログラムを作成しなくともシステムが持つ会話形式の編集，または入力コマンドを使って行われる。この点は１型２型とも同じである。

(1) イベント駆動１型

１型では，原則として応用ソフトウエアが制御を持っている。Paradox の場合以下の構文で応用ソフトウエアと Paradox の間で制御のやり取りが行われる。

```
WAIT
  PROC WaitProc
  EventList
ENDWAIT
```

WAIT コマンドで制御が応用ソフトウエアから Paradox に移り，Paradox の入力コマンドを使った入力が可能となる。EventList に指定したイベントが発生すると，ユーザー作成の応用ソフトウエアの一つであるプロシージャ：Wait Proc が呼び出され，その中でイベントの種類を判定しそれぞれに対応したプログラムを実行する。

(2) イベント駆動２型

２型の場合は，制御は原則としてシステムが持っており，ユーザーが要求したときにのみ，そのイベントに対応するユーザー作成の応用ソフトウエアに制御が渡される。その機構はメニュー画面処理の場合と同じものが使われる。

5　ＰＣ用ＡＰＰ開発システムに於けるイベント駆動型の位置づけ

イベント駆動型の本来の意味は，任意のタイミングでユーザーがプログラムに介入しうることである。従来のオンラインプログラムはこの意味で総てイベント駆動型であり，イベント駆動の概念は新しいものではない。しかしこの種のプログラムは，多くアセンブラ等低位言語で組まれ，かなり高度の技術が必要で，一般のユーザーが作成するのは困難であった。ＰＣ用開発システムでイベント駆動型を導入した目的は，この開発システムで構築された応用システムが柔軟なＧＵＩを持ちうることである。そのためにはディスプレイ画面上の図形要素に基づくイベントを検知する機能が必要で，しかもその取り扱いが一般

ユーザーでも可能でなければならない。ＰＣ用ＡＰＰ開発システムでは，フォーム上にコントロールを配置し，イベントプロパティにプロシージャを割り当てることでイベント駆動プログラミングが可能になっており，いわば，非手続型言語の中にイベント駆動の機能を持ち込むことにより上記の目的を達成している。

イベント駆動の機能を一般ユーザーに開放することにより応用ソフトウエアが柔軟なＧＵＩを持ち得ることになったが，これはＰＣ用ＡＰＰ開発システム自身が持つ豊富なＧＵＩの機能を応用ソフトウエアに移植できることを意味しており，例えば，応用ソフトウエアの中で入力時リストボックスを表示するなどのヘルプ機能も容易に実現出来るようになっている。

Ⅴ　テーブル単位の処理

ＲＤＢの操作言語に於けるテーブル単位処理と手続き型言語のレコード単位処理を比較し，それがソフトウエア開発の効率にどのように影響しているかを検討する。比較の焦点を絞るため，検討対象として，受注処理時のテーブル参照，在庫テーブルの更新と出荷処理時の入出庫テーブルへのレコード追加を取り上げる。これらの処理に於ける処理アルゴリズムは，テーブルの性格に強く依存するので先ずテーブルの種類を検討する。次に受注処理，出荷処理の概要を記述し，その後テーブル参照，在庫更新，レコード追加について，テーブル単位処理とレコード単位処理の比較を行う。

１　テーブルの種類

マスター・テーブル：システムの中核となるデータを集めたテーブルで，顧客マスターテーブル，商品マスターテーブル等がこれに当たる。

トランザクション・テーブル：システムの諸活動により発生するデータを

集めたテーブルで，例えば受注テーブル等がこれに当たる。

ヒストリー・テーブル：実績として保存すべきデータを集めたテーブル。このデータは通常，トランザクション・データを蓄積することにより作られる。入出庫テーブルの場合は，入出庫が行われる度にデータが蓄積されていき在庫もその時点の在庫が記録されていく。

ステータス・テーブル：そのときのシステムの状態を表すテーブルで，在庫テーブル等，受注や出荷が行われ状態が変化する度にそれを反映するように更新される。

2　受注処理，出荷処理の概要

処理内容は極力簡単化し，以下のように想定する。

＜受注処理＞

(1) 使用テーブルとデータ項目

① 受注テーブル

{受注番号，日付，顧客コード，商品コード，受注数}

② 顧客マスターテーブル

{顧客コード，顧客名}

③ 商品マスターテーブル

{商品コード，商品名，商品単価}

④ 在庫テーブル

{商品コード，日付，在庫数，引当数，受注可能数}

(2) 処理概要

① 受注データの入力

＊ データ入力

受注番号，日付，顧客コード，商品コード，受注数を入力する。

＊ テーブル参照

顧客コードより顧客マスターテーブルの顧客名を参照，表示する。

商品コードより商品マスターテーブルの商品名と商品単価を参照，表示する。

＊　受注高の計算表示

受注高 ＝ 商品単価 × 受注数

② 在庫テーブルの更新

当日受注分につき，在庫テーブルの下記更新を行う。

引当数 ＝ 引当数 ＋［商品毎の受注数の合計］

受注可能数 ＝ 受注可能数 －［商品毎の受注数の合計］

（ここの等号“＝”は両辺が等しいという意味ではなく，右辺を左辺に代入するという意味である。）

＜出荷処理＞

その週に受注したものを週末にまとめて出荷することとする。即ち，週末に，その週の受注に対し，出荷処理を行うことになる。

(1) 使用テーブルとデータ項目

① 受注テーブル

受注処理と同じ

② 入出庫テーブル

{受注番号，商品コード，日付，顧客コード，入庫数，在庫数}

③ 在庫テーブル

受注処理と同じ

(2) 処理概要

① 入出庫テーブルへの出庫レコードの追加

出荷日付を入力し，これに等しい日付を持つ受注テーブルのレコード（即ち今週の受注分）を抽出する。これらをもとに入出庫テーブルに出荷情報を記録する。

② 在庫テーブルの更新は受注の場合とほぼ同様である。

(3) 入出庫テーブルの在庫数計算アルゴリズム

① “在庫数”は，各商品毎のその時点での在庫数を示す履歴データである。商品の入出庫は，時間の経過に従って時系列的に行われる。

＊ 在庫数は入出庫が行われた直後に於けるその商品の在庫数を表す。

＊ 在庫数は新たな入出庫が行われて，現在の在庫数が更新されてもその時点のデータとして保存されなければならない。

② “入出庫テーブル”の定義及び表現

入出庫テーブル ＝ {ROW(i) | i=1,2,…,N}

ROW(i) は行であり，また i は行番号であるとともに，入出庫が行われた時間的順序を表している，N 行の個数である。

ROW(i) ＝ {受注番号(i)，・・・，在庫数(i)} である。

新たに入出庫が行われると ROW(N+1) が追加される。

J ＝ Max(i | 商品コード(i) ＝ 商品コード(N+1),i<(N+1))

とし

在庫数(N+1) ＝ 在庫数(J) ＋ 入庫数(N+1)

を計算する。

3 受注処理時のテーブル参照

受注処理時入力データを極力少なくするため，ある項目を入力すればその項目により一意に定まる他の項目はコンピュータが参照するようにする。例えば商品コードを入力すれば，商品名や商品単価は商品マスターテーブルを参照して取り出すようにする。

(1) 手続き型言語の場合

入力した商品コードをキーとして商品マスターテーブルを読み，読みとったレコードから目的の商品名や単価を得る。手続き型では目的を達成するための手続きを詳細に記述しなければならない。

(2) ＰＣ用ＡＰＰ開発システムの場合

これに反して，ＰＣ用ＡＰＰ開発システムの場合は次のようにフォームを作成すれば上記の参照は自動的に行われる。

① テーブル間のリレーションシップの設定

② レコードソースとなる“問い合わせ”の作成

顧客コードで受注テーブルと顧客マスターテーブルを，

商品コードで受注テーブルと商品マスターテーブルを結合する。

③ この問い合わせをレコードソースとしてフォームを作成

④ この問い合わせの項目，受注番号，日付，顧客コード，顧客名，商品コード，商品名，商品単価，および受注数をフォームに配置

ここでは“プログラム”は“フォームを作成する事”に当たる。フォームの作成はＧＵＩを使って“テーブルを結合する”,“項目をフォーム上に配置する”等実現すべき目的に沿った記述，即ち非手続き的記述となる。

4　受注処理時の在庫更新

１回のデータ入力操作で複数の受注件数を処理するものとする。例えば当日の受注をまとめて入力する場合等がこれに当たる。

(1) 手続型言語の場合

手続型言語では通常レコード単位処理が行われる。即ち１件の受注データを入力する毎に下記の処理を行う。

① 商品コードをキーとして在庫テーブルを読む

引当数 ＝ 引当数 ＋ 受注数

受注可能数 ＝ 受注可能数 － 受注数

② 引当数，受注可能数を出力領域に転記

③ 在庫テーブルに再書き込み

(2) ＰＣ用ＡＰＰ開発システムの場合

① 在庫更新用のコマンドボタンをフォーム上に配置する

ＰＣ用ＡＰＰ開発システムの場合はイベント駆動の機能を持っており，

これを使って上記ボタンをクリックしたとき在庫更新用のマクロが起動されるよう設定する。

② 在庫更新用マクロは次の“問い合わせ”から成る。

＜ 商品毎の受注数合計を作る“問い合わせ”＞

この“問い合わせ”は以下の機能を持つ。

＊ 受注テーブルから指定日付のレコードを抽出し，部分テーブルを作る

＊ この部分テーブルを商品コードでグループ化する

＊ 各グループ毎に（即ち商品毎に）合計を計算する

＜ 在庫テーブル引当数更新を行う“問い合わせ”（更新問い合わせ）＞

＊ 商品コードで在庫テーブルと上記の“問い合わせ”結果のテーブルを結合する

＊ 引当数 ＝ 引当数 ＋ 受注数の合計

受注可能数 ＝ 受注可能数 ＋ 受注数の合計

在庫更新用マクロが起動されると上記２つの“問い合わせ”が続けて実行され結果として在庫テーブルの引当数と受注可能数が更新される。

(3) 比較

① アルゴリズム

今１日分の受注を１回にまとめて入力する場合を考える。即ち入力操作を開始し，複数件の受注データを入力し，入力操作を終了する。この入力操作を１操作単位と考えれば，１操作単位の中に複数件の受注データを含む場合となる。手続き型の場合は１件の受注毎に在庫テーブルを更新しており，１日分の受注の合計数は個々の受注数を在庫テーブルに足し込むことにより求められている。これに対しＰＣ用ＡＰＰ開発システムでは，操作単位中の総てのデータを商品毎にグループ化し，グループ毎の合計を計算し，その合計を使い１回の処理で在庫テーブルを更新している。グループ化，合計計算はデータ操作言語である“問い合わせ”で行っている。

② 在庫テーブル更新のタイミング

在庫の更新は，各データにつき必ず１回行われなければならない。更新が行われないデータや２回以上更新されるデータがあってはならない。手続き型の場合はプログラムで手順が総て決められるので問題はない。ＰＣ用ＡＰＰ開発システムで，会話型で操作が進められるので，更新の起動をどのタイミングで行うかは注意を要する。

5　出荷処理時の入出庫テーブルへのデータ追加

(1) 手続き型言語の場合

① 入力日付と同じ日付を持つ受注ファイルのレコードをサーチする。

② このレコードにより出荷情報を商品入出庫ファイルに追加する。

③ 在庫数の計算

在庫数(N+1) ＝ 在庫数(J) ＋ 入庫数(N+1)

④ 在庫数(N+1)の取り出し方

この方法はテーブルの編成方法に大きく依存する。

(2) ＰＣ用ＡＰＰ開発システムの場合

① 入力日付と同じ日付を持つ受注テーブルのレコードを抽出し，出来た部分テーブルを受注サブテーブルとする。

② 受注サブテーブルと受注明細を受注番号で結合する。

③ 結合されたテーブルのフィールド

受注番号，商品コード，日付，顧客コード，受注数（入庫数）

を入出庫テーブルに追加する。

④ 上記①，②，③はクエリー“入出庫追加Q”により行われる。

⑤ 在庫数の計算

在庫数(N+1) ＝ 在庫数(J) ＋ 入庫数(N+1)

⑥ 在庫数(N+1)の取り出し方

この処理は，テーブル内の行の順序に依存するためレコード単位の処理

が必要となる。多くのＰＣ用システム開発ではレコード単位処理は会話処理形式ではなく，プログラム形式で行うようにしている。ここでは，Microsoft社の Access Basic のプログラムを例にして説明する。

① Recordset 型のオブジェクト MyTable, FTable, FTable1 を定義する。

② MyTable, FTable1 に在庫テーブル，入出庫テーブルを代入する。

③ 在庫テーブルの最初のレコードにカーソルを合わせ，最初のレコード内の商品コードを取り出す（MoveFirst)。

④ 入出庫テーブルより上記商品コードと等しい商品コードを持つレコードを抽出し，その部分テーブルを FTable とする(Filter)。

⑤ FTable から入力日付と等しい日付を持つレコードの内最初のものを取り出す(FindFirst Criterion)。

⑥ FTableから（即ち同じ商品グループの中から）在庫数（Ｎ－１）を取り出すためカーソルを１レコード分もどして(FTable MovePrevious)，在庫数（Ｎ－１）を取り出し，変数Ｘに代入する。

⑦ カーソルを１つ進め

在庫数 ＝ 入庫数 ＋ Ｘ

Ｘ ＝ 在庫数

を計算する。

⑧ ⑦ を FTable の最後まで繰り返す。

⑨ ③ に於いて在庫テーブルの最後のレコードまで④～⑧を繰り返す。

(3) 比較

ここではテーブル単位ではなくレコード単位の処理を行っているが，Access Basic ではテーブル操作を意識したレコード単位処理機能

MoveFirst

MoveNext

MovePrevious

等があり，ＣＯＢＯＬよりもプログラミングが容易である。

Ⅵ　アプリケーション開発システムとしての支援範囲

1　一般的な情報システムの開発過程[注1]

［Ｉ］計画フェーズ

（1）システム構想立案

新システムの必要機能とあるべき姿をまとめる。

（2）現状の調査分析

新システムと現状とのギャップを検討し解決策を提示する。

（3）基本計画作成

新システムの実現化計画を作成し関連部門の承認を得る。

［II］設計フェーズ

（1）概要設計（論理設計）

入出力情報の確定，システム機能の細分化，論理データ設計を行う。

（2）詳細設計（物理設計）

入出力仕様の設定，物理データ設計，プログラムの機能，および構造の決定を行い，システム仕様を確定する。

［III］構築フェーズ

（1）プログラミング

プログラムの処理論理を明確にし，業務プログラムのコーディングを行う。

（2）テスト

テスト仕様に従い，プログラムの単体テスト，サブシステムの結合テ

スト，負荷テスト，および総合テストを行う。

［Ⅳ］運用フェーズ

（1）新システムへの移行

本番用のデータを登録し，新システムへ移行する。

（2）評価

目標を達成したか否かの評価。

（3）保守

不具合点の修正，機能の追加，および変更を行う。

2　各課程に於けるＰＣ用開発システムが支援する主な項目

このうちＰＣ用開発システムが支援するのは［II］設計フェーズ　（2）詳細設計（物理設計）以降の過程である。

［II］設計フェーズ

（2）詳細設計（物理設計）

①　ＡＰＰメニューの設計

②　入力フォームの設計

③　テーブル構造の設計

④　レポート書式の設計

⑤　ＡＰＰ処理論理の設計，制御の流れの設計

［Ⅲ］構築フェーズ

（1）プログラミング

①　ＡＰＰメニューの作成

②　入力フォームの作成

③　テーブル構造の作成

④　レポート書式の作成

⑤　ＡＰＰ処理論理の作成，制御の流れの作成

（2）テスト

① デバッグ

② テストデータの作成

［Ⅳ］運用フェーズ

（1）新システムへの移行

① 本番データの作成

② データインポート

3 保守

① テーブル，クエリー等の変更機能

以上のようにＰＣ用開発システムが支援するのはシステム開発の主として下流部分であるが，詳細設計，プログラム構築フェーズに於いてＧＵＩの構成要素であるメニューや入力フォームの設計・作成を支援範囲に含める等，従来のシステムに比べ支援の幅を拡げている。

注1 参考文献（株）東和コンピュータマネジメント著「情報システムの開発と設計」p14の図を参考に一部修正した。

まとめ

以上見てきたように，最近のパーソナルコンピュータ用アプリケーション開発システムはいずれもグラフィカル・ユーザー・インターフェースの採用により操作性を向上し，プログラミングのオブジェクト化と，メインプログラムの機能をシステムが提供することにより，ユーザーのシステム開発負荷を大幅に減じている。

現在，オブジェクト化を更に進めてソフトウエア部品化し，アプリケーショ

ン開発をこれら部品の組み合わせで実現する試みが行われている。今後とも，こうしたアプリケーション開発方法の変化の動向に注目する必要がある。

参考文献

1 KEFFREY D.ULLMAN, Principles of DATABASE SYSTEMS, Computer Science Press, inc. 1982.（国井利安泰・大保信夫訳，「データベース・システムの原理」，日本コンピュータ協会，1985.）

2 平尾隆行，「関係データベースシステム」，近代科学社，1992.

3 C.J.Date, A Guide to THE SQL STANDARD, Addison-Wesley Publishing Company, Inc. 1989.（芝野耕司監訳，「標準ＳＱＬ」,（株）トッパン，1994.）

4 Per O.Flaatten, Donald J.McCubbrey, P.Declan O'Riordan, Keith Burgess, Foundations of Business Systems, The Dryden Press 1989.（アンダーセンコンサルティング東京事務所訳，「情報システム構築ハンドブック」，ＨＢＪ出版局，1990.）

5 (株)東和コンピュータマネジメント，「情報システムの開発と設計」，啓学出版，1987.

6 原田実監修，「ＣＡＳＥのすべて」，オーム社 1991.

7 Brad J.Cox, Andrew J.Novobilski, Object-Oriented Programming, Addison-Wesley Publishing Company,Inc. 1991.（松本正雄訳，「オブジェクト指向のプログラミング」，(株)トッパン，1992.）

8 Borland International,「Paradox 4.0J ユーザーズガイド」,「Paradox 4.0J ＰＡＬプログラマーズガイド」，ボーランド株式会社，1993.

9 Microsoft Corporation,「Microsoft Access バージョン 2.0 ユーザーズガイド」，「アプリケーション開発ガイド」，マイクロソフト株式会社,1995.

10 日経コンピュータ，別冊，「Windows用開発ツール徹底ガイド」，日経ＢＰ社，1994.7.29.

SUMMARY

According as high-performance workstations and personal computers appear, trends of "Down Sizing" which replace host/office computers by workstations/personal computers become widespread.

Because of personal computer's remarkable progress, many corporations intend to apply personal computer to their information processing systems.

Many software vendors have provided application development systems in order to fill those corporation's needs.

Distinctive features of these application development systems are having graphical user interface and original object-oriented languages and also being based on relational databases.

In this paper, we discussed the influences of these application development systems on the development process of application systems.