第１１章　応用事例

指導目標

本章では、ＣＯＢＯＬ入出力の応用としてデータベース・アクセスと、データ処理形態の一つであるオンライン処理についてプログラミング技法の面から解説し、概念を理解させる。

データベース操作については、ＪＩＳで「データベース言語ＳＱＬ」の規格があり、その付属書として「ＳＱＬ埋込みホストプログラム」の規格もある。
この付属書は、ＣＯＢＯＬ、ＦＯＲＴＲＡＮ、Ｐａｓｃａｌ、ＰＬ／Ｉの各言語でＳＱＬと一緒に記述する場合の規格について書かれている。

データ処理形態は、バッチ処理からネットワークとデータベースを効果的に使うオンライン処理が一般的になっている。データベースやオンライン・トランザクション処理は、汎用コンピュータからワークステーション、パソコンまで幅広く普及している。また、オンライン・トランザクション処理については、ＵＮＩＸ上で構築できるようになってきた。
これらの技術について、プログラミングからみた基礎的な考え方を学習する。

内容のあらまし

内　　容	説　明	議　論	机上実習	計算機実習
ﾃﾞｰﾀﾍﾞｰｽ・ｱｸｾｽ	ﾃﾞｰﾀﾍﾞｰｽ操作とCOBOL入出力の考え方を説明する。	COBOL入出力の問題点を議論する。		
全体概要 (ﾃﾞｰﾀﾍﾞｰｽ)	COBOLとﾃﾞｰﾀﾍﾞｰｽ言語の関係を説明する。		COBOL入出力の概要を整理させる。	
埋込み構文の概要	ﾃﾞｰﾀﾍﾞｰｽ言語がCOBOLに翻訳されていく課程を説明する。	COBOL一般形式とSQLの構文表記法の違いを議論する。		COBOLｿｰｽに埋込みｺﾝﾊﾟｲﾙして理解させる
ホスト変数	ﾃﾞｰﾀﾍﾞｰｽ言語とﾃﾞｰﾀの受け渡しを行うﾃﾞｰﾀ名の使い方を説明する。			
データの型	ﾃﾞｰﾀﾍﾞｰｽ言語で扱うﾃﾞｰﾀ型との関係を説明する		COBOLﾃﾞｰﾀ型をSQLのﾃﾞｰﾀ型で記述する	
例外宣言	COBOL入出力のEND/INVALIDに対応するﾃﾞｰﾀﾍﾞｰｽの例外宣言を説明する。	例外宣言が必要な理由と使用法を議論する。		例外を発生させ分岐することを実習する
カーソル操作	ﾃﾞｰﾀﾍﾞｰｽの行をｱｸｾｽするｶｰｿﾙ操作について説明する。	COBOL入出力とｶｰｿﾙ操作の共通点を議論する。		ｶｰｿﾙ操作は重要なので使用法を充分に理解させる。
オンライン処理	ﾃﾞｰﾀ処理形態の一つとしてのｵﾝﾗｲﾝﾄﾗﾝｻﾞｸｼｮﾝ処理の概要を説明する。	ﾊﾞｯﾁ処理との違いを議論する。		
全体概要 (ｵﾝﾗｲﾝ・ｼｽﾃﾑ)	ｵﾝﾗｲﾝに必要なｿﾌﾄｳｪｱについて説明する。	身の周りで稼動しているｵﾝﾗｲﾝｼｽﾃﾑをあげて使用形態を議論する。	ｵﾝﾗｲﾝに必要なOSの機能を整理させる。	

内　　容	説　明	議　論	机上実習	計算機実習
ファイル入出力	ﾌｧｲﾙ入出力の問題点を説明し、現状の実現方式の例を説明する	ﾊﾞｯﾁ処理との違いを議論する。		
ﾌｧｲﾙの回復処理	回復処理の必要性と実現方式を説明する。			ｴﾗｰを発生させて回復処理を実習する。
端末処理	ｵﾝﾗｲﾝ端末画面の特性を説明し標準ﾌｧｲﾙとの違いを説明する。	ﾊﾟｿｺﾝ画面での入出力処理と比較を行う。	端末の機能を整理させる。	端末の色、罫線等を実際に変えてみる。
対話処理ﾌﾟﾛｸﾞﾗﾑの構造	会話型、擬似会話型の特徴を説明する。	TSS処理とｵﾝﾗｲﾝﾄﾗﾝｻﾞｸｼｮﾝ処理の対話の違いについて議論する		
ｺﾝﾊﾟｲﾙから実行まで	事前ｺﾝﾊﾟｲﾙの機能とﾛｰﾄﾞﾓｼﾞｭｰﾙの構造について説明する。			事前ｺﾝﾊﾟｲﾙを行いｿｰｽﾌﾟﾛｸﾞﾗﾑの変換課程を実習する。

第１１章　応用事例

１１．１　データベース・アクセス

ＣＯＢＯＬでは、ファイルを「記憶媒体上に格納されたり、そこから取り出されたりするレコードの集まりである」と概念づけている。
そのファイルをアクセスするために、ファイル編成、ファイルの処理、手順を規定して利用者が自由に選択できる。

・ファイル編成　－　ファイルの論理構造を記述する属性である。
順編成、相対編成、索引編成

・ファイル処理　－　ファイルのレコード操作やファイル全体の操作。
順呼出し法、乱呼出し法、動的呼出し法
入力、出力、拡張、入出力両用

これらは、ファイルの構造面からみて記憶媒体の物理構造を意識して構成されている。
そこで視点をデータ処理に重点をおいた「データベース」という概念が考案された。
データベースではデータの重複をさけ共用することで大量のデータを効率よく処理し、また、データをプログラムから独立させてデータベース管理システムが一元的に管理する方式がとられている。

データベースの構築と操作のために構文と意味を規定した「データベース言語」も登場した。
データベース言語の規格には、データベース操作文をＣＯＢＯＬ言語などに埋め込むための機能も規定されている。これらを使用すればＣＯＢＯＬのファイル操作と同じように機種に依存することなくデータベース操作が行える。

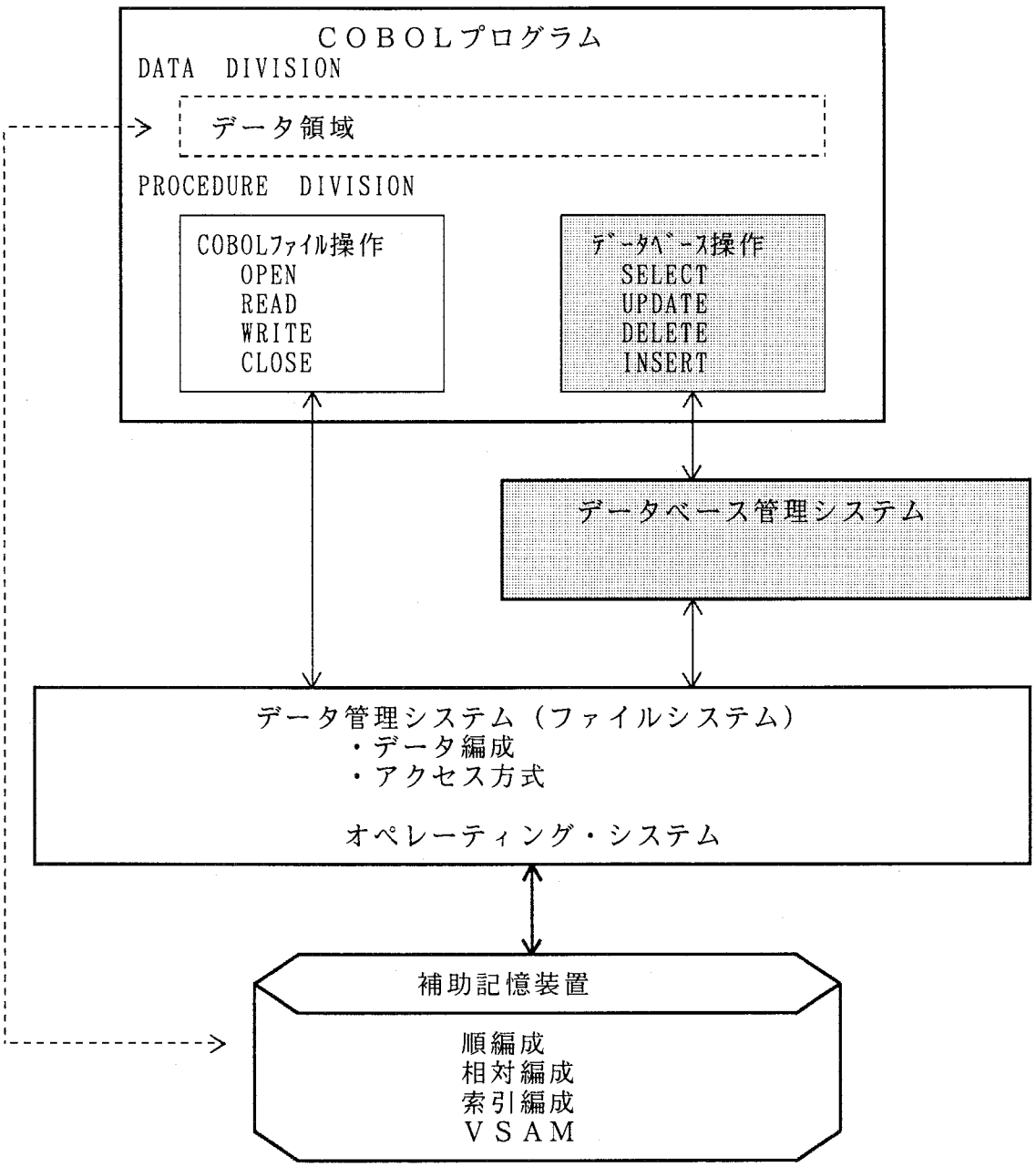
ここでは、データベース言語としてＳＱＬ（Structured Query Language）をとりあげることにする。
そして、ＣＯＢＯＬ言語からデータベース言語ＳＱＬの使い方を学習する。

ＳＱＬのデータベース操作文には、次のものがある。

・ＣＬＯＳＥ文
・ＣＯＭＭＩＴ文
・ＤＥＬＥＴＥ文
・ＦＥＴＣＨ文
・ＩＮＳＥＲＴ文
・ＯＰＥＮ文
・ＲＯＬＬＢＡＣＫ文
・ＳＥＬＥＣＴ文
・ＵＰＤＡＴＥ文

１１．１．１　全体概要

ＣＯＢＯＬ言語とデータベース言語の関係は、次のとおり。



１１．１．２　埋込み構文の概要

ＣＯＢＯＬ原始プログラムに書かれたデータベース言語の翻訳から実行までの仕組みは、次のとおり。

ＣＯＢＯＬ原始プログラムにはＳＱＬ文を、次のように記述する。

```
EXEC  SQL
                SQL文
END-EXEC
```

ＪＩＳでは「ＳＱＬ埋込みＣＯＢＯＬプログラム」として次のように規定されている。

```
<COBOL変数定義>   : : =
                  {０１ | ７７}   <COBOLﾎｽﾄ識別子>   <COBOL型指定>

<COBOLﾎｽﾄ識別子>   : : =
                  ﾎｽﾄ変数を定義するCOBOLﾃﾞｰﾀ記述項の制限された形式とする。

<COBOL型指定>   : : =
                  <COBOL文字型>
                  | <COBOL数値型>
                  | <COBOL整数型>

<COBOL文字型>   : : =
                  PIC [TURE]    [IS]   X(<長さ>)

<COBOL数値型>   : : =
                  PIC [TURE]    [IS]
                  S {<9> [V<9>]
                   | <9>V
                   | V<9>}
                   [USAGE   [IS] ]   DISPLAY  SIGN  LEADING  SEPARATE

<COBOL整数型>   : : =
                  PIC [TURE]    [IS]
                  S<9>
                   [USAGE   [IS] ]   COMP [UTATIONAL]

<９>   : : =   {9 [(<符号なし整数>)] } ･･･
```

注：ＪＩＳの構文表記法は、ＢＮＦ（ﾊﾞｯｶｽｰﾅｳｱ形）に［］（省略可能）、･･･（繰り返し）、｛｝（要素の並び）の拡張を行った表記法である。

ＣＯＢＯＬ原始プログラム中のＳＱＬ文は、ＳＱＬ埋込み文の構文を解析して規格に合致したＣＯＢＯＬ原始プログラムを生成する「事前コンパイラ」で処理されて、その後でＣＯＢＯＬコンパイラ、リンカと処理されロードモジュールが作成される。

１１．１．３　ホスト変数

ＳＱＬ文内でデータ部のデータ名を参照する場合、命名規則がありその変数をホスト変数という。
データ名の前に「：」をつけることでＳＱＬ文のキーワードと区別する。

```
：データ名
```

規則は、次のとおり。

```
＜埋込み変数名＞　：：＝
　　：＜ホスト識別子＞

＜ホスト識別子＞　：：＝
　　＜ＣＯＢＯＬホスト識別子＞

＜ＣＯＢＯＬホスト識別子＞　：：＝
　　有効なＣＯＢＯＬ変数名
```

ホスト識別子はＳＱＬのキーワードと一致してはならない。
ＳＱＬのキーワードには、下記のものがありＣＯＢＯＬの予約語と同じようにデータ名に使用しないようにする。

```
ALL       AND         ANY    AS    ASC       AUTHORIZATION    AVG
BEGIN     BETWEEN     BY
CHAR      CHARACTER   CHECK        CLOSE     COBOL
COMMIT    CONTINUE    COUNT
CREATE    CURRENT     CURSOR
DEC       DECIMAL     DECLARE      DELETE    DESC    DISTINCT    DOUBLE
END       ESCAPE      EXEC         EXISTS
FETCH     FLOAT       FOR          FORTRAN   FOUND   FROM
GO        GOTO        GRANT        GROUP
HAVING
IN        INDICATE    INSERT       INT       INTEGER   INTO    IS
LANGUAGE  LIKE
MAX       MIN         MODULE
NOT       NULL        NUMERIC
OF        ON          OPEN         OPTION        OR    ORDER
PASCAL    PLI         PRECISION    PRIVILEGES    PROCEDURE    PUBLIC
REAL      ROLLBACK
SCHEMA    SECTION     SELECT       SET       SMALLINT    SUM
TABLE     TO
UNION     UNIQUE      UPDATE       USER
VALUES    VIEW        WHENEVER     WHERE      WITH    WORK
```

１１．１．４　データ型

ＳＱＬにはいくつかのデータ型があるがＣＯＢＯＬとの関係は、次のように定義されている。

- ＜COBOL文字型＞は、文字列変数を記述する。等価なＳＱＬデータ型は同じ長さをもつ「ＣＨＡＲＡＣＴＥＲ」とする。
- ＜COBOL数値型＞は、真数変数を記述する。等価なＳＱＬデータ型は同じ精度とと位取りをもつ「ＮＵＭＥＲＩＣ」とする。
- ＜COBOL整数型＞は、真数変数を記述する。等価なＳＱＬデータ型は「ＩＮＴＥＧＥＲ」とする。

ＳＱＬのデータ型には、次のものがある。

```
＜データ型＞　：：＝
　　＜文字列型＞
　　｜＜真数型＞
　　｜＜概数型＞

＜文字列型＞　：：＝
　　ＣＨＡＲＡＣＴＥＲ［（＜長さ＞）］
　　｜ＣＨＡＲ［（＜長さ＞）］

＜真数型＞　：：＝
　　ＮＵＭＥＲＩＣ［（＜精度＞［，＜位取り＞］）］
　　｜ＤＥＣＩＭＡＬ［（＜精度＞［，＜位取り＞］）］
　　｜ＤＥＣ［（＜精度＞［，＜位取り＞］）］
　　｜ＩＮＴＥＧＥＲ
　　｜ＩＮＴ
　　｜ＳＭＡＬＬＩＮＴ

＜概数型＞　：：＝
　　ＦＬＯＡＴ［（＜精度＞）］
　　｜ＲＥＡＬ
　　｜ＤＯＵＢＬＥ　ＰＲＥＣＩＳＩＯＮ

＜長さ＞　：：＝
　　＜符号なし整数＞

＜精度＞　：：＝
　　＜符号なし整数＞
＜位取り＞　：：＝
　　＜符号なし整数＞
```

１１．１．５　例外宣言

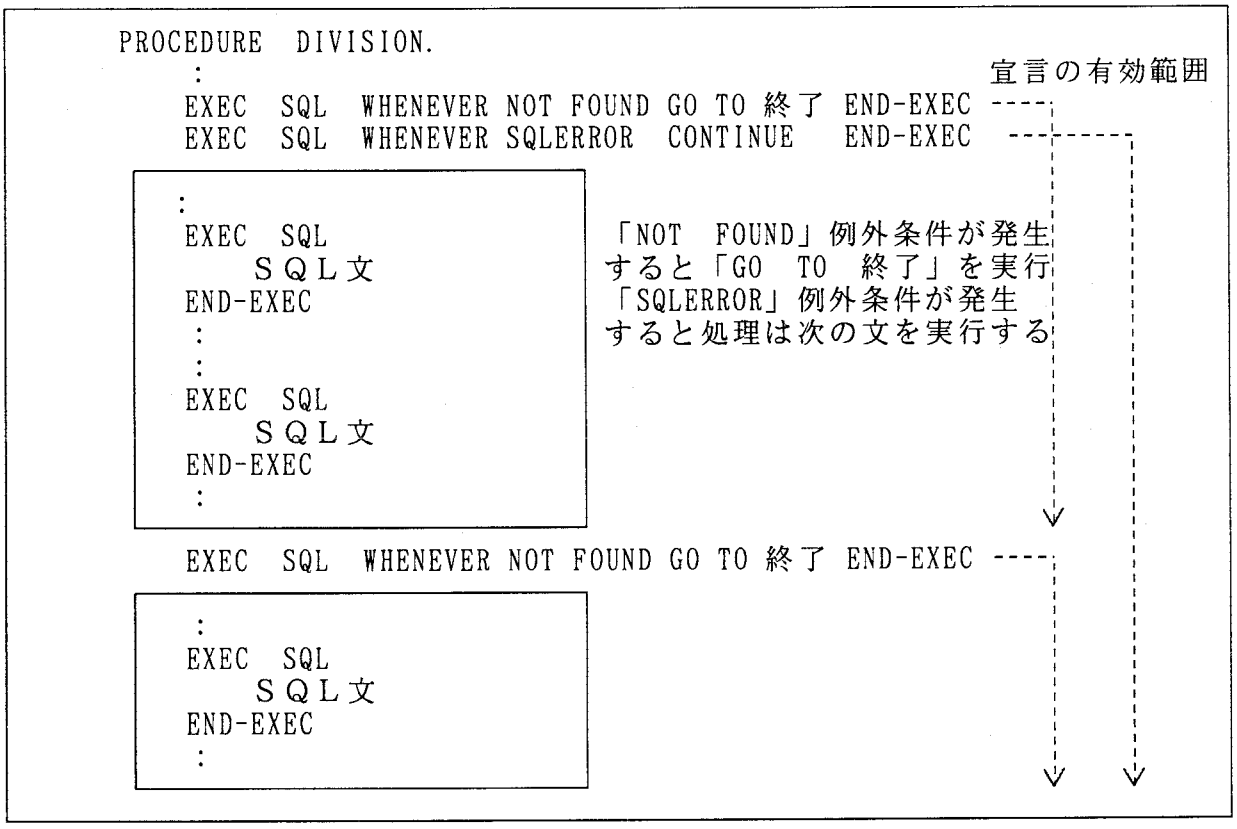
ＳＱＬ文の実行で例外条件が発生したときにとられる動作を「埋込み例外宣言」で指定する。

```
EXEC　SQL
　　WHENEVER　＜条件＞　＜例外動作＞
END－EXEC
```

ＷＨＥＮＥＶＥＲ文は、実行文ではなく宣言文である。

```
＜埋込み例外宣言＞　：：＝
　　WHENEVER　＜条件＞　＜例外動作＞

＜条件＞　：：＝
　　ＳＱＬＥＲＲＯＲ｜ＮＯＴ　ＦＯＵＮＤ

＜例外動作＞　：：＝
　　ＣＯＮＴＩＮＵＥ｜＜ＧＯ　ＴＯ＞

＜ＧＯ　ＴＯ＞　：：＝
　　｛ＧＯＴＯ｜ＧＯ　ＴＯ｝　＜相手＞

＜相手＞　：：＝
　　：＜ホスト識別子＞｜＜符号なし整数＞
```

ＷＨＥＮＥＶＥＲ文の機能は、原始プログラムで次に宣言されるまでの全てのＳＱＬ文に適用される。



１１．１．６　カーソル操作

プログラミング言語からは、複数行が選択されるＳＥＬＥＣＴ文は実行できない。そこで特殊なＳＥＬＥＣＴ文を使用すると、行を１行ずつアクセスすることができる。この機能をカーソル操作という。

カーソル操作の基本的な考え方は、COBOLのOPEN、READ、REWRITE、DELETE、CLOSE処理と同じである。

```
DATA  DIVISION.
    :①取り出す表の選択条件と列名の「カーソル」を定義する。
 EXEC  SQL  DECLARE  ｶｰｿﾙ名  CURSOR  FOR
            SELECT  列名1  列名2  列名3
            FROM    表名
            WHERE   選択条件
 END-EXEC
    :
PROCEDURE  DIVISION.
    :②OPEN文でｶｰｿﾙをｵｰﾌﾟﾝすると①のSELECT文が実行される。
 EXEC  SQL
     OPEN  ｶｰｿﾙ名
 END-EXEC
    :③ｶｰｿﾙの行が終了した場合の動作を定義する。
 EXEC  SQL
     WHENEVER  NOT  FOUND  GOTO  EOF-CURSOR
 END-EXEC
    :
LOOP.
    :④FETCH文で表の１行を取り出しﾌﾟﾛｸﾞﾗﾑのﾃﾞｰﾀ名にﾃﾞｰﾀを格納する。
 EXEC  SQL
     FETCH  ｶｰｿﾙ名  INTO  :ﾃﾞｰﾀ名1  :ﾃﾞｰﾀ名2  :ﾃﾞｰﾀ名3
 END-EXEC
    :
    :⑤必要ならばDELETE文で１行の削除ができる。
 EXEC  SQL
     DELETE  FROM  表名
         WHERE  CURRENT  OF  ｶｰｿﾙ名
 END-EXEC
    :⑥必要ならばUPDATE文で１行追加できる。
 EXEC  SQL
     UPDATE  表名  SET  列名  =  :ﾃﾞｰﾀ名
         WHERE  CURRENT  OF  ｶｰｿﾙ名
 END-EXEC
    :⑦最後の行を処理するまでFETCHを繰り返す。
   GO  TO  LOOP
EOF-CURSOR.
    :⑧ｶｰｿﾙをｸﾛｰｽﾞする。
 EXEC  SQL
     CLOSE  ｶｰｿﾙ名
 END-EXEC
    :
```

１１．２　オンライン処理

事務処理のデータ処理形態には、大量データを一括処理するバッチ処理と、データを即時に処理するオンライン・トランザクション処理（ＯＬＴＰ）などがある。
プログラミングの観点からみても処理方式が異なる部分が多い。

オンライン・トランザクション処理は、ネットワークを介して多数の端末から処理要求がオンライン・システムに送られてくるので、プログラムは処理要求を効率的に処理するように作成する必要がある。

データの流れからみるとプログラムは次のような処理方式になっている。

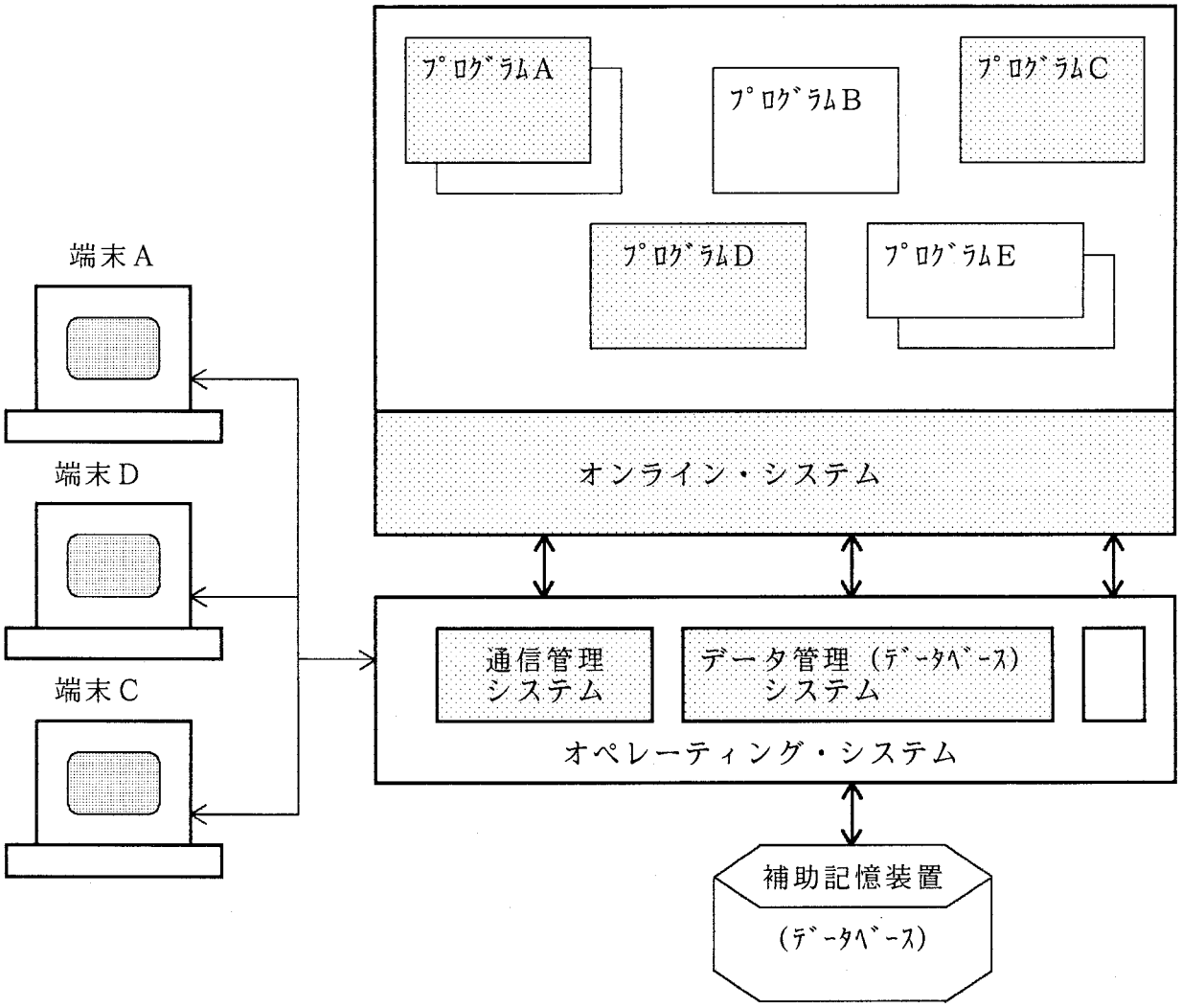
- 端末画面から処理要求であるトランザクション・コードとパラメータをデータとしてを受け取る。
- そのデータでデータベースをアクセスして要求を処理する。
- データ処理結果を端末画面へ表示する。

プログラムの大きさからみると、複数の処理要求を短時間に処理する必要があり、しかも主記憶装置の容量にも制限があるため、オンライン・システムを効率よく稼動させるには、プログラムを機能単位に分割して小さくすることが望まれる。
トランザクションとは、業務を構成する処理単位のことである。

１１．２．１　全体概要

オンライン・システムの身近な例としては、列車の座席予約や銀行オンラインがある。
これらのシステムは、データを統合的に管理するためにデータベース管理機能（ＤＢ）や遠隔地からのアクセスを可能にするデータ通信機能（ＤＣ）を統合して構築されている。

オンライン・システムの概要は、次のとおり。



オンライン処理には、処理形態としてシステム（プログラム）との対話を中心とした照会処理、照会更新処理、データ・エントリ処理、メッセージ交換処理などがある。
これらの処理は、コンピュータに接続された端末を使用して複数の利用者が離れた場所から同時に処理要求をだし、そして、その処理結果の応答についても即時性が要求される。
オンライン処理では、システムのトラブルが発生すると会社の業務が停止する事態になるため障害対策がハードウェアとソフトウェアに求められている。

１１．２．２　ファイル入出力

バッチ処理では、ファイルの入出力をＣＯＢＯＬのOPEN、READ、START、REWRITE、WRITE、DELETE、CLOSE命令文を使用するが、オンライン・システムでは各種の制御が必要になり、これらの命令文を直接使用することが禁止されている場合が多い。
これらの制限は、オンライン・システムの障害時におけるファイルやデータベースの障害からの回復機能が必要であることによる。

また、オペレーティング・システムの機能を直接使用するACCEPT、DISPLAY、SORTなどの命令文も使用禁止されている場合もある。
これらの制限は、オンライン・システムでは即時処理を要求されているため可能な限りオペレーティング・システムのオーバーヘッドを少なくしてオンライン・システムで一括処理するためである。

例としてオンライン・システムの開発をＣＯＢＯＬ言語で行う場合のファイル入出力は次のとおり。

```
DATA  DIVISION.
WORKING-STORAGE  SECTION.

     データ項目の定義

PROCEDURE  DIVISION.

    COBOL  命令文

    EXEC  CICS                                    ｵﾝﾗｲﾝ･ｼｽﾃﾑが提供する
          機能  ﾊﾟﾗﾒｰﾀ  ･･･                        ﾌｧｲﾙ入出力を使用する。
    END-EXEC

    COBOL  命令文
           :
```

具体的なファイル入出力の例は、次のとおり。
考え方はＣＯＢＯＬ言語と大差はない。

```
     EXEC  CICS
          READ      DATASET(ﾌｧｲﾙ名)  INTO(ﾚｺｰﾄﾞ領域)  ･･･
           機能                          ﾊﾟﾗﾒｰﾀ
           ・WRITE
           ・REWRITE
           ・DELETEなど

     END-EXEC
```

注：　ＣＯＢＯＬの入出力命令文を記述してオンライン・システムを開発できるシステムもあるが、これらはＣＯＢＯＬの入出力ライブラリがバッチ処理の場合と異なるものをリンクして実現している。

また、通常ファイルではなくデータベースにアクセスするときは、ＳＱＬ埋込み機能を使用する場合が多い。

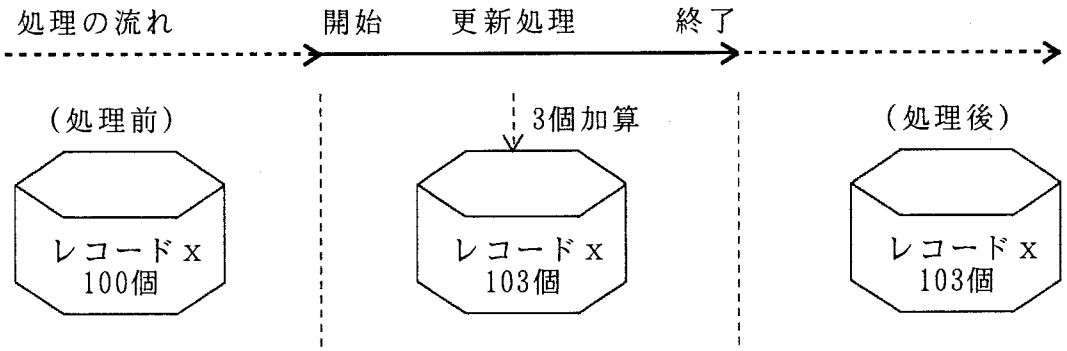
```
     EXEC  SQL
           ＳＱＬ文
     END-EXEC
```

ＳＱＬ文のデータベース入出力には、SELECT、INSERT、DELETE、UPDATE、FETCHなどがある。

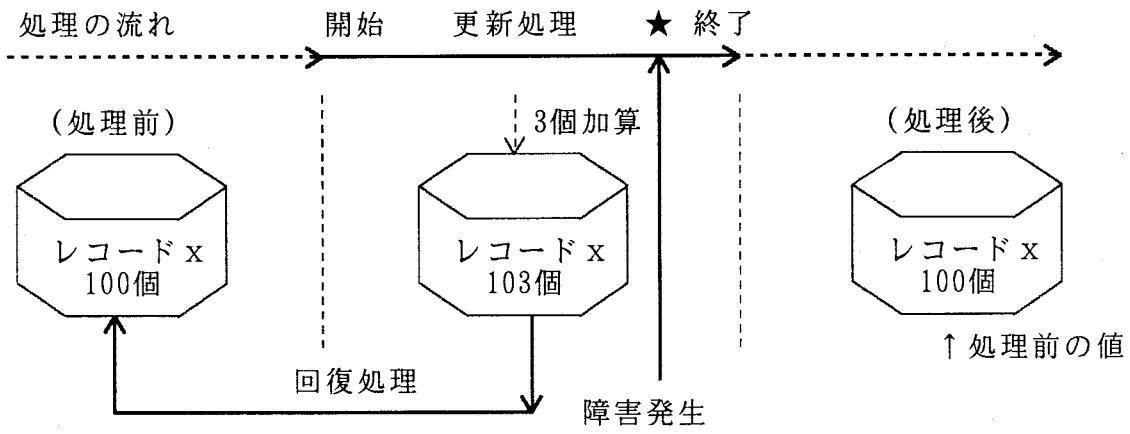
１１．２．３　ファイルの回復処理。

オンライン・システムでは、処理の途中で障害が発生しプログラムが中断した場合に回復処理が行われる。
回復処理の基本的考え方は、「その処理がなかった」かのようにすることである。
回復処理をファイルの入出力からみると、レコードの更新処理を行った後に障害が発生してプログラムが中断するとレコードはオンライン・システムによって更新前の状態に復元される。

・正常の場合



・障害発生の場合



通常、オンライン・システムではプログラム（トランザクション）が正常終了したときまたは、特別な命令（COMMIT文等）を実行したときに、実際のファイルに対してレコードの追加、削除、更新が行われる。
このように処理単位での整合性をとることを同期点（synchronous point）処理という。
回復処理をプログラミングからみると、ファイルの同期点処理を理解することである。

データベースの同期点をＳＱＬ文でとる場合は、次のとおり。

```
EXEC  SQL
      COMMIT  WORK
END-EXEC
```

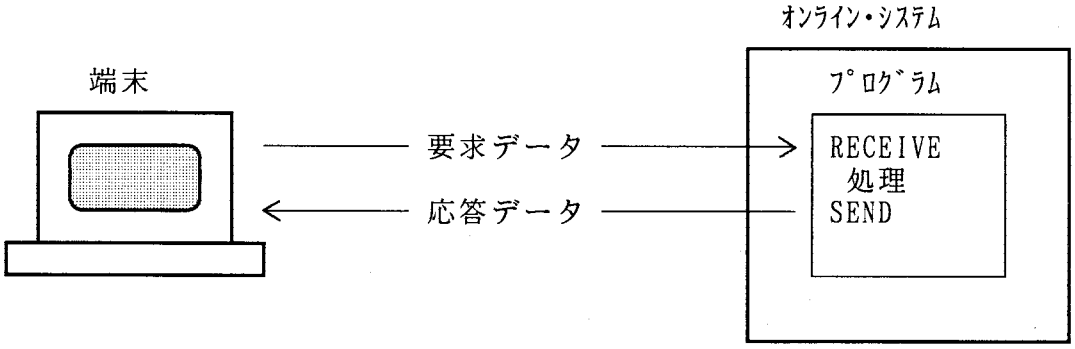
ファイルの同期点をとる場合の例は、次のとおり。

```
EXEC  CICS
      SYNCPOINT
END-EXEC
```

１１．２．４　端末処理

オンライン・システムでは、端末を使用して処理要求を出したり、その処理結果を表示したりする対話処理を行う。

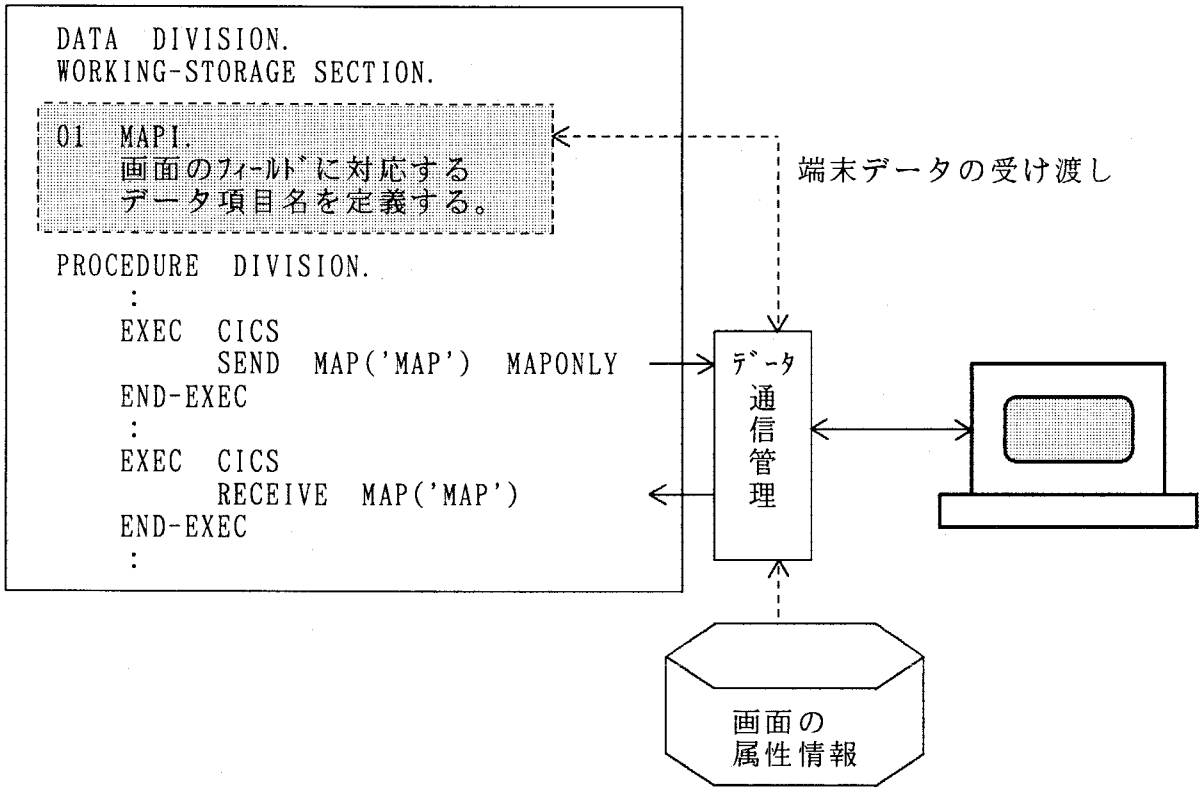
端末での対話処理は、次のとおり。



端末画面にデータを表示する場合には、表示するデータの位置（行、列）とサイズの他に、色、輝度、罫線の有無、カーソルの位置、入力・出力フィールドなどの情報も端末側に送る必要がある。

オンライン・システムでの端末の入出力は、ＣＯＢＯＬのＲＥＡＤ、ＷＲＩＴＥを使わない場合が多い。

例として、オンライン・システムでの端末入出力は、次のとおり。



具体的な端末の入出力の例は、次のとおり。考え方はＣＯＢＯＬ言語と大差はない。

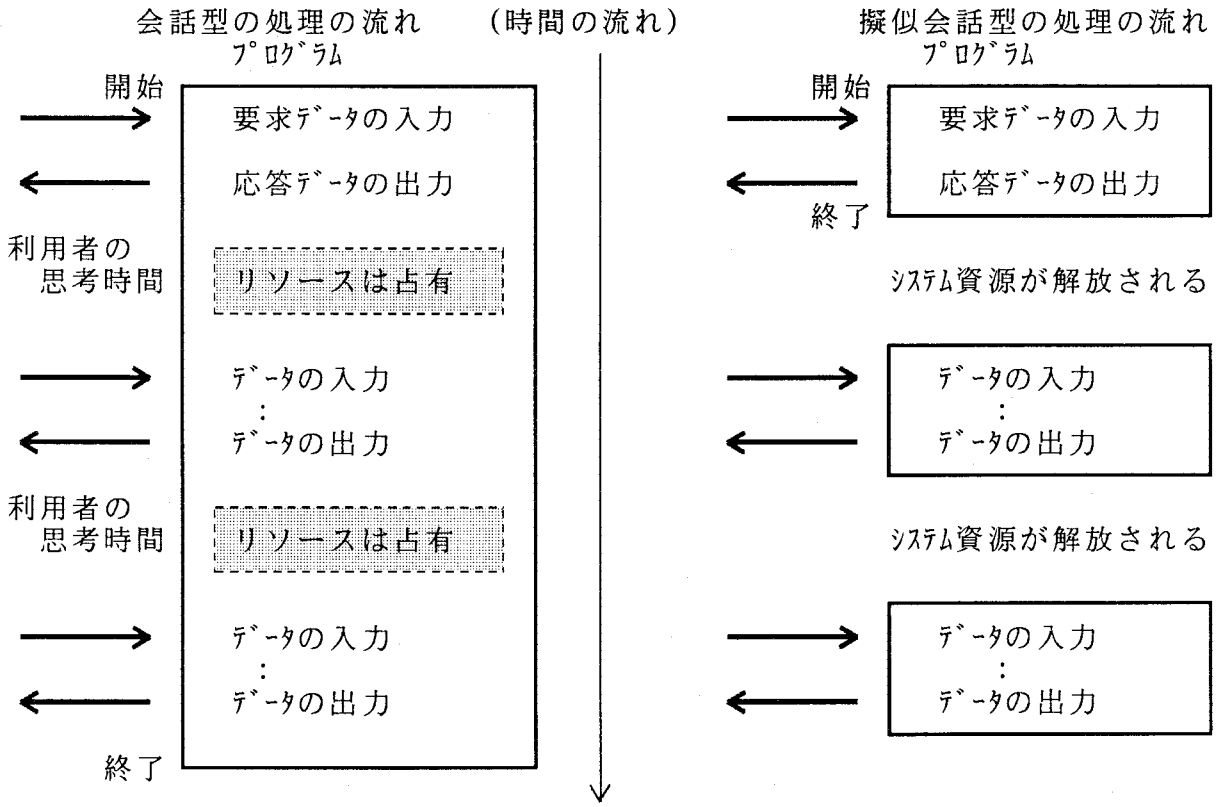
```
EXEC CICS
     SEND      MAP(ﾏｯﾌﾟ名)  FROM(ﾃﾞｰﾀ領域)  ･･･
     機能                    ﾊﾟﾗﾒｰﾀ
     ・RECEIVE
END-EXEC
```

１１．２．５　対話処理プログラムの構造

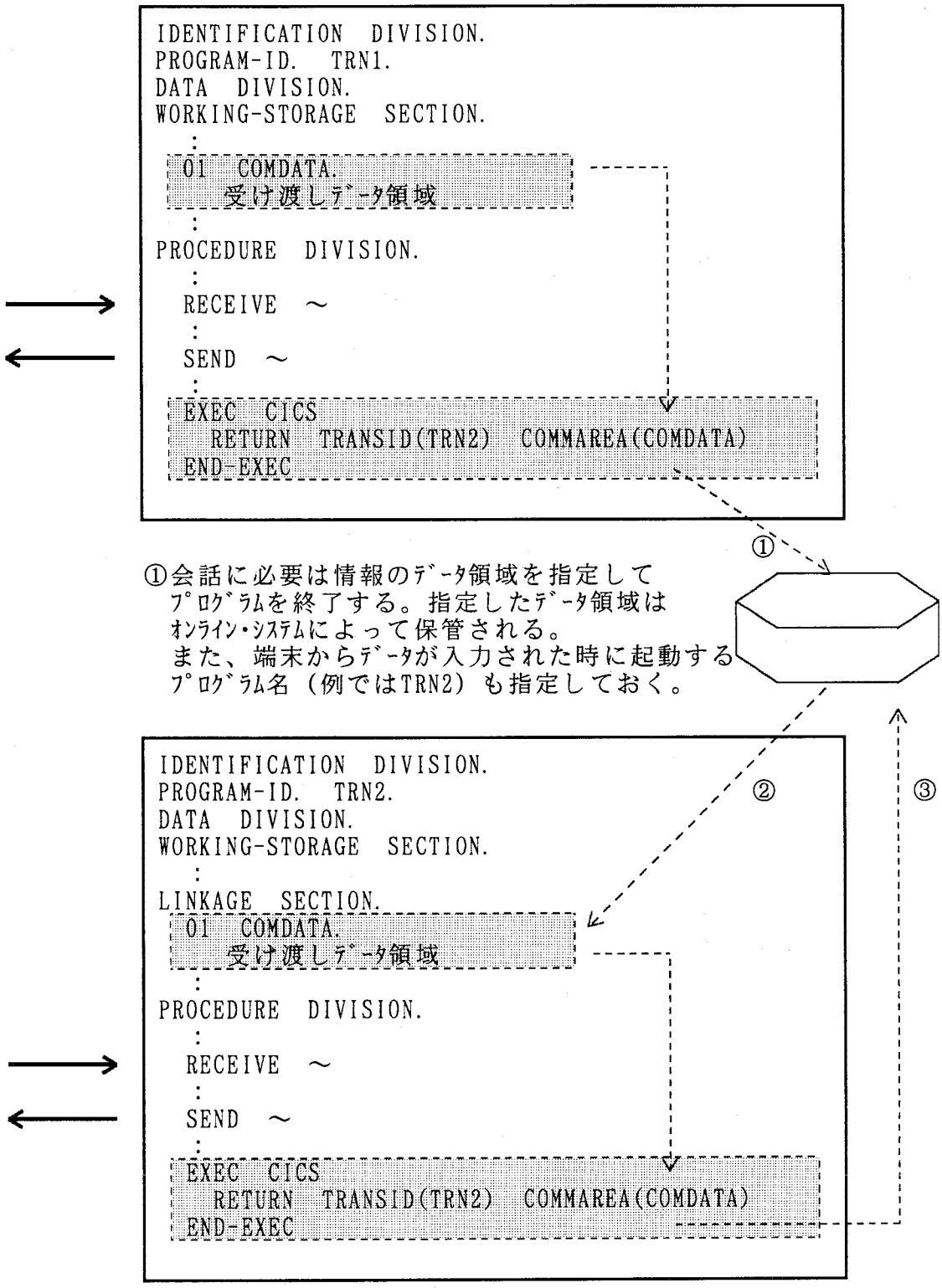
オンライン・システムの対話処理のプログラム構造として、二つの作り方がある。
基本的な考え方は、処理時間に重点を置くか、システムの資源に重点を置くかによる。

(1)会話型（繰り返しトランザクション型）
プログラムが起動されると、端末からデータを受信し、要求を処理して結果を端末へ送信する。これらの操作を終了指示がでるまで繰り返し処理する方式である。
コンピュータに接続されている端末数が少なく、データの入力操作が早い場合の業務に有効である。

(2)擬似会話型（多段トランザクション型）
端末からデータを受信する度にプログラムを起動して、その処理を行い、結果を送信してプログラムは終了する。次に起動されるまでの間のシステム資源は他のプログラムが使用できる。
コンピュータに接続されている端末数が多い場合に有効である。



擬似会話型（多段ﾄﾗﾝｻﾞｸｼｮﾝ）のプログラム構造は、次のとおり。

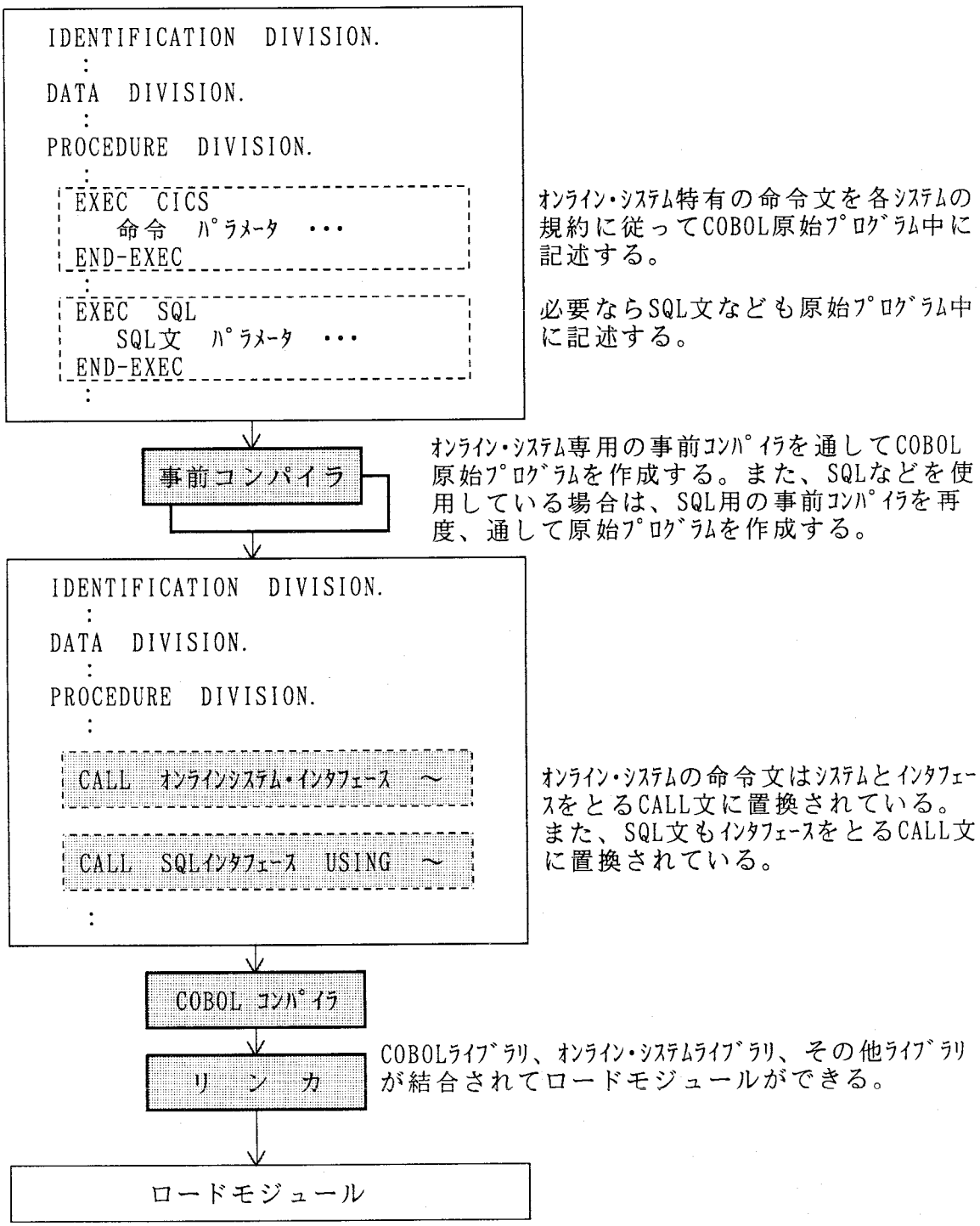


②端末からデータが入力されるとプログラムが起動されるが、その時、前回保管した会話に必要なデータ領域が、データ部のLINKAGE-SECTIONに設定されてからプログラムが実行されることになる。これによって、処理の継続性が保たれる。

③再度、会話に必要なデータ領域を指定してプログラムを終了する。この時、再起動された場合のプログラム名も一緒に指定して終了するが、業務を終了する場合には、プログラム名を指定しないで終了すればよい。

１１．２．６　コンパイルから実行まで

ＣＯＢＯＬ原始プログラム中に記述されたオンライン・システム特有の命令文は、「事前コンパイラ」によって規格に合致したＣＯＢＯＬプログラムに変換される。



オンライン・システムで、ロードモジュールを実行するにはシステム定義ファイルに、プログラム毎の属性（タイプ、優先度、実行時間など）と動作環境を定義して初めて実行可能になる。

指導上の留意点

◎データベース
ＣＯＢＯＬのファイル編成は、多量のデータを効率的に処理するように設計されているが、情報が多様化し、情報量もより大規模になった現在では、ＣＯＢＯＬのファイル編成では効率的にカバーしきれない状況が発生している。
データベースの考え方として、重複の回避、共用化、プログラムの独立などの概念があり、ＣＯＢＯＬプログラムからデータベースをアクセスすることが世界的に標準化されている。

ＣＯＢＯＬファイル編成とデータベースを比較して、各々の長所、短所を議論することによってＣＯＢＯＬのファイル処理の理解を深める。

関係型データベースと階層型データベースについてもその特徴について解説する。

◎ＳＱＬ文
ＣＯＢＯＬのＲＥＡＤ、ＷＲＩＴＥ、ＤＥＬＥＴＥ、ＲＥＷＲＩＴＥなどの入出力文と同じように、データベースをアクセスするためのＳＱＬにもそれらに対応するＳＱＬ文がある。
これらのＳＱＬ文の考え方がＣＯＢＯＬと大差ないことを理解させる。
ＳＱＬ文の概要（データ操作関連）は、次のとおり。

```
OPEN  <ｶｰｿﾙ名>

FETCH  <ｶｰｿﾙ名>  INTO  <取り出し相手ﾘｽﾄ>

SELECT  [ALL | DISTINCT]  <選択ﾘｽﾄ>  INTO  <選択相手ﾘｽﾄ>

UPDATE  <表名>  SET  <列名>=<値式 | NULL>  ･･･  [WHERE  <探索条件>]

INSERT  INTO  <表名>  [(<挿入列ﾘｽﾄ>)]  {VALUES(<挿入値ﾘｽﾄ>)}

DELETE  FROM  <表名>  WHERE  CURRENT  OF  <ｶｰｿﾙ名>

CLOSE  <ｶｰｿﾙ名>

COMMIT  WORK

ROLLBACK  WORK

DECLARE  <ｶｰｿﾙ名>  CURSOR  FOR  <ｶｰｿﾙ指定>
```

◎システム構造
データベース管理システム（ＤＢＭＳ）の役割について解説を行う。
主な項目は、次のとおり。
・排他制御
・回復機能
・機密保護

◎ＳＱＬ埋め込み構文
ＪＩＳでは、ＳＱＬ埋込みホストプログラムとして、次の言語が規定されている。

・ＣＯＢＯＬプログラム
・ＦＯＲＴＲＡＮプログラム
・Ｐａｓｃａｌプログラム
・ＰＬ／Ｉプログラム

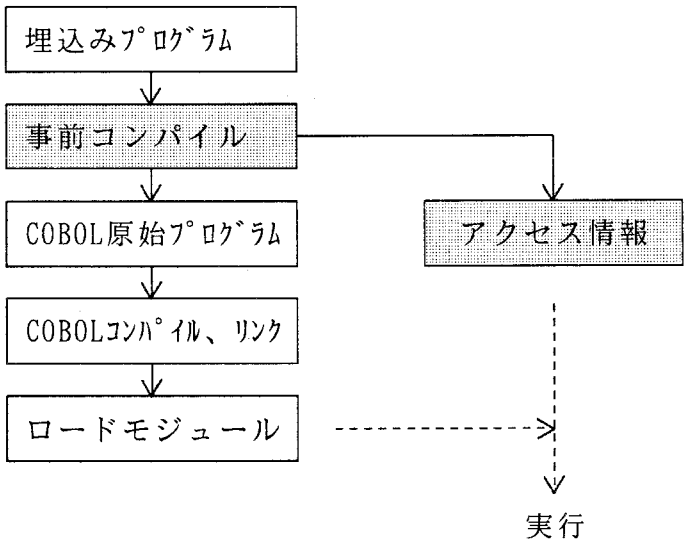
◎ＳＱＬ先頭子と終了子
ＳＱＬではＳＱＬ先頭子は「EXEC　SQL」、終了子は「END-EXEC」または「；」とある。
これら言語構文により次の制限がある。

ＣＯＢＯＬ	EXEC	SQL	ＳＱＬ文	END-EXEC
ＦＯＲＴＲＡＮ	EXEC	SQL	ＳＱＬ文	FORTRAN、Pascal、PL/I にEND-EXECは書けない
Ｐａｓｃａｌ	EXEC	SQL	ＳＱＬ文 ；	
ＰＬ／Ｉ	EXEC	SQL	ＳＱＬ文 ；	

◎ＳＱＬのデータ形式
ＳＱＬ埋込みホストプログラムで取り扱えるデータ形式は、次のとおり。

ＣＯＢＯＬ	USAGE　DISPLAY USAGE　COMPUTATIONAL
ＦＯＲＴＲＡＮ	CHARACTER　［*長さ］ INTEGER REAL DOUBLE　PRECISION
Ｐａｓｃａｌ	PACKED　ARRAY INTEGER REAL
ＰＬ／Ｉ	CHARACTER(長さ) FIXED　DECIMAL(精度［,位取り］) FIXED　BINARY［(精度)］ FLOAT　BINARY(精度)

◎事前コンパイル
ＣＯＢＯＬプログラムに埋め込まれたＳＱＬ文は、事前コンパイラ（precompiler）によってＳＱＬ構文が検査され、正常ならばＣＯＢＯＬ言語に変換される。
プログラムは事前コンパイラの変換過程でデータベースをアクセスするための各種情報が抽出されて特定のファイルに保管される。
プログラム毎に保管されたデータベースのアクセス情報と、データベースに定義されたアクセス権（機密保護、排他制御など）の情報が比較され、処理権限が許されている場合にプログラムは実行できることになる。



◎データ構造

関係型データベースとファイルの論理データ構想について解説する。
基本的には、ＣＯＢＯＬのレコードの概念と同じである。
関係型データベースの基本構造は表形式であり、行と列で構成されている。
行はＣＯＢＯＬのレコードに相当し、列はレコードを構成している基本項目と考えればよい。

	列	列	列	列	列
	学生番号	氏　名	年齢	～	～
行 →	００１	森泉　昭弘	１７		
行 →	００２	村上　　淳	１８		
行 →	００３	竹山　敏雄	１６		
行 →	００４	小原　　均	１７		

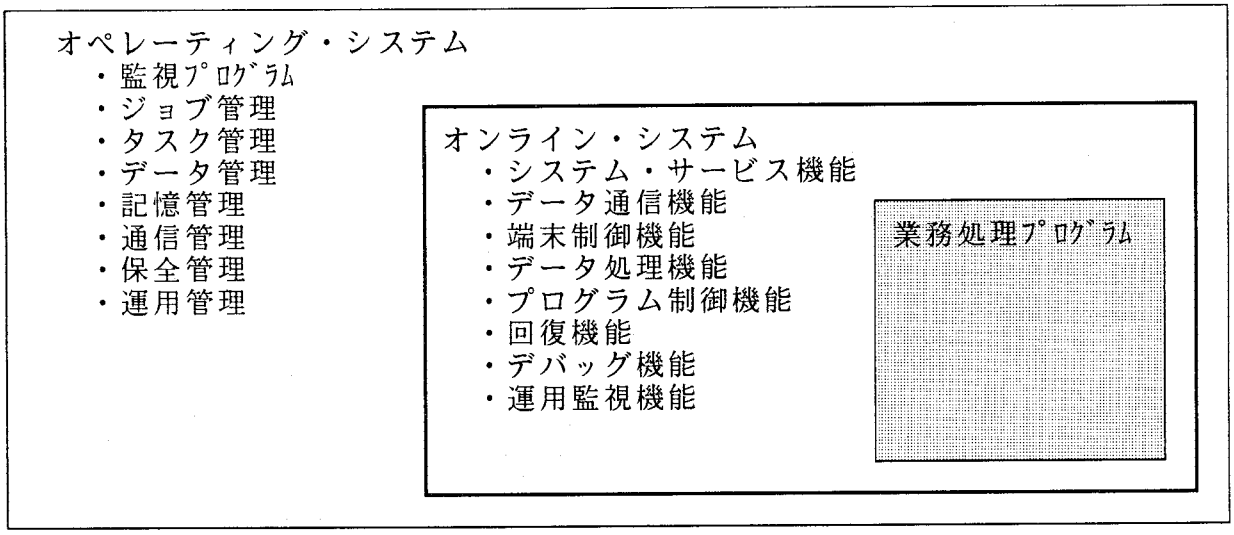
◎データ処理

関係型データベースの表操作については、ＣＯＢＯＬの索引ファイルの考え方と似ている。
行のアクセスを高速に行うため、列の値を索引（index）とすることができる。索引の有無についてはプログラムで特に意識する必要はない。
ＣＯＢＯＬのALTERNATE　RECORD　KEYと同様に、行には複数の索引を付けることもできる。

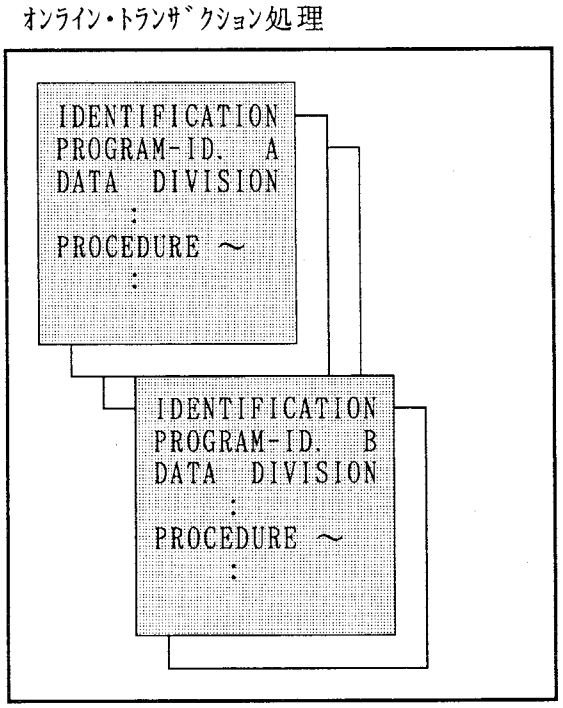
◎オンライン・システム
オンラインとは、通信回線を使用して端末とコンピュータ、コンピュータとコンピュータと接続することをいう。
処理形態からみても、オンライン・リアルタイム処理やタイムシェアリング処理がある。
本章では、オンライン・リアルタイム処理のなかのトランザクション処理を中心に解説する。

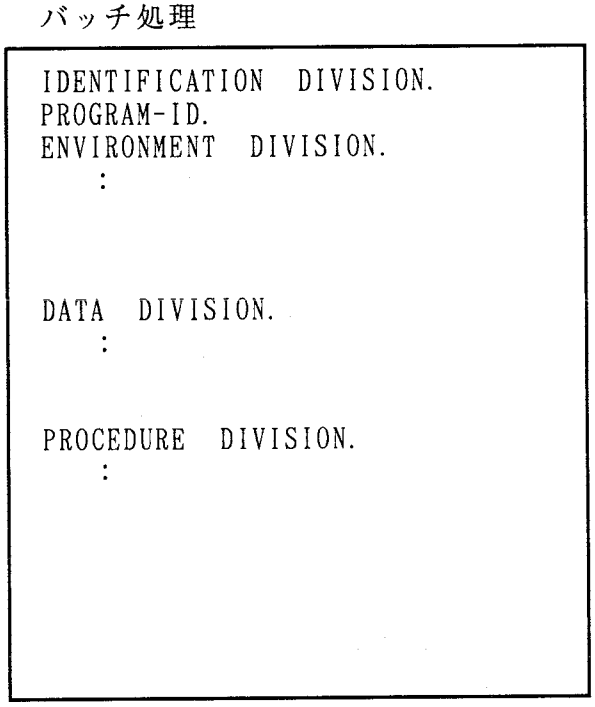
従来、汎用コンピュータを中心にオンライン・トランザクション処理は発展してきたが、今後、ＵＮＩＸワークステーションでも実現されつつある。

◎システム概要
オンライン・トランザクション処理を行うオンライン・システム自体が一種のオペレーティング・システムのような機能を持っている。
そのため、プログラミングもバッチ処理の感覚では理解しにくい状況もある。



◎バッチ処理との比較
オンライン・トランザクション処理とバッチ処理を記憶装置上での構造（ﾒﾓﾘ空間）からみると、次のとおり。（汎用コンピュータの例）





◎オンライン・データ管理
オンライン・トランザクション処理でのデータ管理機能は、ＶＳＡＭファイルとデータベースを中心にアクセスできるようになっているシステムが多い。
従来型の順ファイルもアクセスできるが、回復処理機能の対象外になっている。

オンライン・トランザクション処理では、ＶＳＡＭファイルの他に「一時記憶」、「一時データ（待ち行列）」などと呼ばれているデータ受渡機能がある。これらの機能はﾌﾟﾛｸﾞﾗﾑ開発において頻繁に使われている。
命令も「ＷＲＩＴＥＱ」、「ＲＥＡＤＱ」などと記述するがＣＯＢＯＬのWRITE、READと大差ない。

◎データベースのアクセス
オンライン・トランザクション処理では、標準ファイル（ＶＳＡＭファイル）のアクセスを提供しているが、今後はデータベースを使用してアクセス方式はＳＱＬ文を使用する方向へ向かうと思われるのでＳＱＬ文の「SELECT」、「INSERT」、「DELETE」、「UPDATE」の構文と機能の理解が必要である。

また、プログラム言語からは複数行が処理されるSELECT文は実行できないようになっているので、プログラミングからみると「カーソル操作」が重要である。

カーソル操作の概念は、次のとおり。

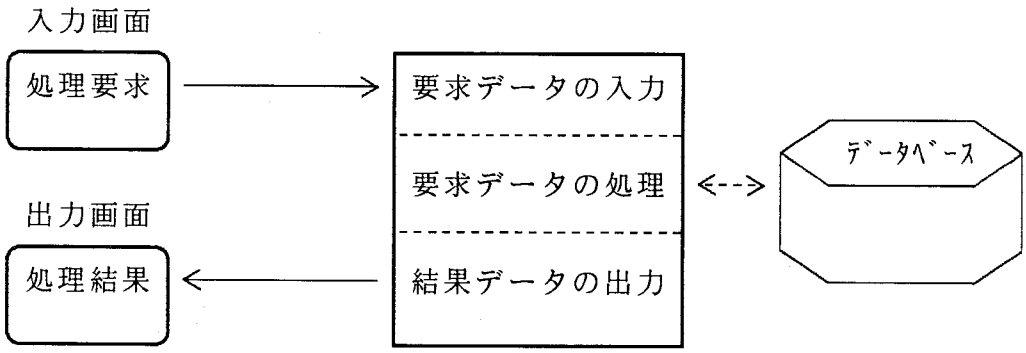
```
①  EXEC  SQL  DECLARE  ｶｰｿﾙ名  CURSOR  FOR
              SELECT      列名1  列名2
              FROM  表名
              WHERE  選択条件
    END-EXEC

②  EXEC  SQL
        OPEN  ｶｰｿﾙ名
    END-EXEC

③  EXEC  SQL
        FETCH  ｶｰｿﾙ名  INTO  ﾃﾞｰﾀ名1  ﾃﾞｰﾀ名2
    END-EXEC

④  EXEC  SQL
        CLOSE  ｶｰｿﾙ名
    END-EXEC
```

ﾃﾞｰﾀﾍﾞｰｽ中の表から選択条件に一致した行の列が取り出され対応するﾃﾞｰﾀ名に入れられる。

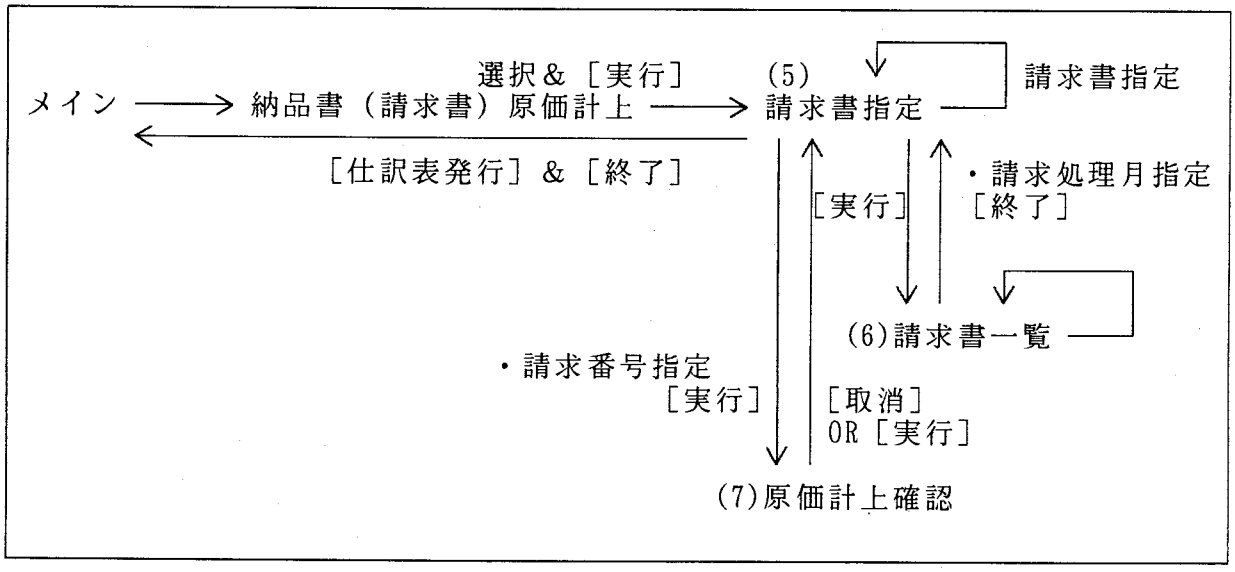
◎回復処理
通常のバッチ処理のプログラムでは、ファイルの入出力を行うとそれがファイルに反映されるがオンライン・トランザクション処理のプログラムでは異なる。また、データベースでも通常ファイルとは異なる概念を持っている。これら、回復処理と同期点処理の必要性と利用法を解説し理解させる。

◎端末処理
ＣＯＢＯＬの標準規格には、端末画面への入出力機能が規定されていない。オンライン・トランザクション処理では、ほとんどの業務処理で対話が必要であり、対話は端末画面を通して行われる。
端末画面は、オンライン・システムのユーティリティ・プログラムを使用して画面レイアウトを定義するとレイアウトに対応したデータ構造の原始プログラムが作成される。
端末画面の入出力フィールドは、利用者が指定したデータ名としてプログラムで参照できる。
端末画面への出力は、「SEND」などと呼ばれる機能で行い、入力は「RECEIVE」などと呼ばれる機能で行うシステムが多い。

◎対話処理プログラムの構造
オンライン・トランザクション処理の端末画面を通じての対話処理は、システムの効率を考えてプログラム（トランザクション）を作成する必要がある。
単純はオンライン・トランザクション処理は、次のような構成であるが実際の業務はこれを複雑に組み合わせた構成となる。



実際の業務における画面遷移の例。



◎コンパイルから実行
ＣＯＢＯＬプログラム中に記述されたオンライン・システム特有の命令文は、事前コンパイラで構文が検査され、正常ならばＣＯＢＯＬ言語に変換される。

例として、ロードモジュールの構造は次のようになっている。