7 章 利用者定義のデータ型機能

COBOL では、レベル番号、項目の名前、PICTURE や USAGE 句の様なデータ属性を指定して、データ項目の構造や形式を表現します。従来の COBOL では、同じ構造のデータ項目を複数定義する場合、レベル番号、項目の名前、各種属性という一連の記述を繰り返し指定する必要がありました。COBOL2002 では、利用者が自由にデータ型(データ構造の雛型)を定義して、そのデータ型を参照することで、同じ構造のデータ項目を簡単に記述することができます。

7.1 データ型と型名

データ型とは、あるデータ項目とその従属項目の形式を全て含む、データ構造の雛型です。
型名とは、データ型を識別するために利用者がその雛型に付与する名前です。

利用者定義のデータ型や型名は、データ記述項にTYPEDEF句で宣言します。データ型の記述は、TYPEDEF句を指定すること以外は、データ項目の宣言と同様に、利用者が自由に記述します(データ型の宣言だけでは、記憶域は確保されません)。

「利用者定義のデータ型」をデータ項目の記述に書く場合は、データ記述項にTYPE句(型名を伴う)を指定します。

データ型を定義し、利用する例を次に示します。

```
01  FEATURE  TYPEDEF.      *> この記述で型名  FEATURE を定義する
  02  FEATURE-NAME  PIC  X(15)  OCCURS  10.
01  EQUIPMENT.             *>
  02  EQUIPMENT-ID  OCCURS  100.
    03     EQUIPMENT-LIST  TYPE  FEATURE.
                           *> 型名  FEATURE を利用する
```

前記の例は、次のレコード記述と同じです。

```
01  EQUIPMENT.             *>
  02  EQUIPMENT-ID  OCCURS  100.
    03  QUIPMENT-LIST  PIC  X(15)  OCCURS  10.
```

データ型の形式は、その型を構成する基本データ項目の相対位置と長さ、およびこれらの基本項目のそれぞれに指定された PICUTURE, USAGE, SIGN, SYNCHRONIZED, JUSTIFIED及びBLANK句によって決まります。

型名を使って宣言したデータ項目は、型名宣言のTYPEDEF句にSTRONG指定があるか否かによって、次の二つに分類されます。弱く型付けされた(基本・集団)項目

- 強く型付けされた(集団)項目

強く型付けされた項目とは、集団項目のデータ内容の妥当性を確保するため、COBOL2002で新たに導入された仕組みです。COBOLの集団項目は英数字項目として扱われるため、送り出し項目と受け取り項目のデータ構造が一致しなくても転記できました。これは、柔軟な転記の記述が許される反面、集団項目中の各基本項目に「不正なデータ」を格納する原因にもなっていました。この様な誤りを防ぐため、強く型付けされた項目という仕組みが導入されました。ただし、基本項目は強く型付けすることができません。

7.1.1 弱く型付けされた項目

型名を使って宣言したデータ項目のうちで強く型付けされていない項目のことを、弱く型付けされた項目と呼びます。弱く型付けされた項目とは、STRONG 指定のない型宣言を参照するものか、STRONG 指定が有効でない型宣言に従属するものです。弱く型付けされた項目は、基本項目であることも、集団項目のこともあります。

弱く型付けされた項目は、指定された型名からそのデータ構造が決まることを除き、型付けされていない項目と同様に使用できます。ですから、TYPEDEF 句で指定する型宣言は、以降で、一連のデータ記述の組を型名という一単語で記述するための手段といえます。

7.1.2 強く型付けされた項目

強く型付けされた項目とは、STRONG 指定付きの型宣言を参照するものか、STRONG 指定付きの型宣言で記述された集団項目だけです。

集団項目中の基本データ項目に格納する内容の整合性を保つため、強く型付けされた(集団項目)は通常の集団項目と比べて許される操作が非常に制限されます。強く型付けされた集団項目中の基本データ項目に格納する内容の整合性を損なう可能性のある操作は全て禁止されます。

＜強く型付けされた項目に関する主な制約＞

- 型の異なる集団項目からの転記は許されない。
- 強く型付けされた項目のデータ記述項に VALUE 句は書けない。
- 強く型付けされた項目やその従属項目を、再定義や再命名してはならない。
- 一部の例外を除いて、部分参照してはならない。
- 強く型付けされた項目を受け取り作用対象として参照できるのは、次の場合のみである。
  - （a） 仮引数や返却項目
  - （b） INITIALIZE文
  - （c） MOVE文
  - （d） READ文のINTO指定
  - （e） RELEASE文のFROM指定
  - （f） RETURN文のINTO指定
  - （g） REWRITE文のFROM指定
  - （h） WRITE文のFROM指定

7.2 TYPE 句と TYPEDEF 句

利用者定義のデータ型機能のため、データ部の句として TYPE 句と TYPEDEF 句の二つが導入されました。ここではそれぞれの概要を説明します。

7.1.1 TYPE 句

データ記述項中の TYPE 句は、型名で示すデータ型の宣言をこの記述項のデータ記述に使うことを表します。

7.1.1.1 書き方

TYPE TO 型名１

7.1.1.2 主な規則

- TYPE 句は、この記述項の左辺のデータ記述が、型名１によって指定されることを表す。TYPE 句の効果は、型名１で識別されるデータ記述をこの TYPE 句の代わりに書いたのと同じである。

7.1.1 TYPEDEF 句

TYPEDEF 句は、このデータ記述がデータ型の宣言であることを表す。

7.1.1.1 書き方

IS TYPEDEF [STRONG]

7.1.1.2 主な規則

- TYPEDEF 句を指定した場合、このデータ記述項はデータ型の宣言となる。この記述項に指定されたデータ名は型名となる。従属するデータ記述項や、条件名記述項、及び RENAME 句は、このデータ型の型宣言の一部となる。これらの従属項で記述された項目のデータ名は、この型名を使用して定義される集団項目の従属項目としてだけ、参照してもよい。その様な集団項目が複数存在する場合、集団項目名での修飾が必要となる。
- 型宣言には、それに関連する記憶域というものがない。
- 記述項の左辺が基本項目であるとき、STRONG 指定を書いてはならない。

7.3 例題

例題として、次の集団項目「人事レコード１」と同じ構造の集団項目「人事レコード２」を記述する場合を考えます。

```
01  人事レコード１.
    05  従業員コード      PIC X(8).
    05  氏名              PIC N(10).
    05  生年月日.
        10  年                  PIC X(4).
        10  月                  PIC X（２）.
        10  日                  PIC X（２）.
```

従来のCOBOLで同じ構造の集団項目「人事レコード２」を記述するには、「人事レコード１」と同様に、レベル番号、項目の名前、各種属性を従属する項目のものを含めて繰り返し記述する必要がありました。

```
01  人事レコード２.
    05  従業員コード      PIC X(8).
    05  氏名              PIC N(10).
    05  生年月日.
        10  年                  PIC X(4).
        10  月                  PIC X（２）.
        10  日                  PIC X（２）.
```

上記の様に何度も同様の記述を書く必要があるため、生産性が悪く、プログラミング時点で誤りを作り込み易いという欠点がありました。又、この様に繰り返し記述したデータ項目の構造を変更する際には、ﾌﾟﾛｸﾞﾗﾑ中で全く別々に宣言された同じ構造のデータ項目を洗い出し、それぞれの宣言に対する修正の要否を検討する必要があり、保守性も悪くなっていました。

COBOL2002では、利用者定義のデータ型をあらかじめ宣言しておいて、同じデータ構造のデータ記述に繰り返し利用することで、上記に示した生産性や保守性の悪さを改善できます。先に述べた「人事レコード１」と「人事レコード２」を、利用者定義のデータ型(人事情報)を使って宣言すると次の様になります。

```
01  人事情報 IS TYPEDEF.
    05  従業員コード        PIC X(8).
    05  氏名                PIC N(10).
    05  生年月日.
        10  年                  PIC X(4).
        10  月                  PIC X（２）.
        10  日                  PIC X（２）.
*> 上記では、「人事情報」という利用者定義のデータ型を宣言しています。
*> 「人事情報」というデータ型の宣言だけで、記憶域は取られていません。
    01  人事レコード１ TYPE  TO 人事情報.
*> 型名「人事情報」を使い、「人事レコード1」を宣言(記憶域の確保)しています。
    01  人事レコード２ TYPE  TO 人事情報.
*> 「人事レコード２」も「人事レコード１」と同様に宣言しています。
```

利用者定義のデータ型を利用する利点を整理すると、次の通りです。

- 同じ構造のデータ項目を簡単に宣言できます。これにより、複雑なデータ構造を何度も記述する必要がないため生産性が向上し、誤りを作りこみにくくなります。
- 同じ構造のデータ項目を同じ型名に統一しておくことで、次の様な副次効果を得ることもできます。
    (ア) 型名を目印に同じ構造のデータ項目を検索できるため、データ型の構造変更による影響範囲の調査が容易になります。
    (イ) データ型の宣言を修正することで、そのデータ型の記述を参照する全てのデータ項目の構造を容易に変更できます。
- 強く型付けされた項目を利用して、集団項目に従属するデータ項目のデータの整合性を確保し易くできます。