第一讲 NetCobol 语法简介和编码规范

01. 变量名：支持全角汉字和半角英数字变量名，可以是全角汉字和全角英数字混排，但是不支持半角和全角混排的变量名，不能有下划线（半角和全角都不行）。

02. COPY 句：支持文件和 DB 的 COPY 句。

① DB 的 COPY 句的定义从 01 层开始，因此不需要在程序中再定义顶层的集团变量，直接引用 <COPY 句名>即可，例如：

定义方法：

```
000010*************************************************************************
000020*    保険料率マスタ(D10HKR_TBL)
000030*************************************************************************
000040 01  保険料率マスタ.
000050     03  Ｄ１０－保険会社区分  PIC  X(004).
000060     03  Ｄ１０－適用開始日    PIC  X(008).
```

引用方法：

```
*--<保険料率マスタ >
     EXEC  SQL  INCLUDE  D10HKR.CBL            END-EXEC.
```

② 文件的 COPY 句中的变量名前头是‘()’，并且是从 03 层开始，因此在使用中需要在程序中定义顶层的集团变量，再把‘()’换成具体的变量名，如

定义方法：

```
000010************************************************
000020* 契約内容(中間ワーク)(CISUF340)
000030************************************************
000040     03    ()契約番号              PIC  X(10).
000050     03    ()再リース回数          PIC S9(2)  SIGN LEADING SEPARATE.
```

引用方法：

```
 01  入力レコード.
     COPY   CISUF340  REPLACING       ==()==  BY  ==入力－==.
```

03. COBOL 中的变量名不能有下划线（包括全角和半角）。

04. 变量定义的级别（层号）：变量定义的级别号是从 01 开始，每层间隔为 2（便于以后的变更维护），例如，01、03、05、07…。

05. 变量的定义：

① 首先有一个大的集团项目，便于对项目进行初期化，如下所示。

```
*-------------------------------------------------------------*
*    作業領域定義                                             *
*-------------------------------------------------------------*
 01  ＷＯＲＫ－エリア.
```

② 错误返回码变量的定义（用于判断显示错误时用）。

```
*--< エラーコード >
     03  Ｗ－エラーコード              PIC S9(04).
```

③ 件数变量的定义。（用于显示件数信息，注意变量定义的长度与程序式样书的要求要一致）

```
*--< 件数エリア >
     03  件数エリア.
         05  Ｗ－入力－件数            PIC  9(09).
         05  Ｗ－出力－件数            PIC  9(09).
```

④ フラグアリア的定义（用于判断的标识）。

```
*--< フラグアリア >
     03  フラグ－エリア.
         05  Ｗ－終了－フラグ          PIC  X(01).
         05  異常終了－フラグ          PIC  X(01).
```

⑤ ＫＥＹ变量的定义（注意进行比较的ＫＥＹ的各个变量的定义类型要一样）。ＫＥＹ的各个变量一般在文件读取时候进行付值。

```
*--< ＫＥＹ－エリア >
     03  Ｗ－ＫＥＹエリア.
         05  Ｗ－ＮＥＷＫＥＹ.
             07  Ｗ－契約番号１        PIC  X(07).
             07  Ｗ－再リース回数１    PIC  9(02).
             07  Ｗ－契約種類１        PIC  X(03).
         05  Ｗ－ＯＬＤＫＥＹ.
             07  Ｗ－契約番号２        PIC  X(07).
             07  Ｗ－再リース回数２    PIC  9(02).
             07  Ｗ－契約種類２        PIC  X(03).
```

06. 有固定值变量的定义：因为上面定义的集团项目在程序开始阶段被初期化，因此对于一些有预先定一值的变量要从新开始一个 01 层来定义（这个变量和下面的常量定义有区别，因为变量中有些值还需要在程序中改变），例如：

```
*--< 共通情報 >
 01  Ｗ－共通情報.
     03  Ｗ－システム日付.
         05  Ｗ－世紀                  PIC  X(02) VALUE "20".
         05  Ｗ－年月日                PIC  X(06).
     03  Ｗ－システム時刻              PIC  X(08).
     03  Ｗ－担当者                    PIC  X(08) VALUE "IKOPG   ".
```

07. 常量的定义：程序中保持值不变的变量可以在常量节（CONSTANT　SECTION）中定义，变量名义以“定数－”开头。

```
CONSTANT                               SECTION.
 01  定数領域.
     03  定数－プログラムＩＤ          PIC  X(08)  VALUE "COBIS47A".
```

08. 注释：详细的注释能使程序容易理解，便于评审和维护。

①DIVISION 部和 SECTION 部的定义，如

```
*****************************************************************
*          ＩＤＥＮＴＩＦＩＣＡＴＩＯＮ　ＤＩＶＩＳＩＯＮ      *
*****************************************************************
```

②内部的某一功能块的定义，如

```
*---------------------------------------------------------------*
*    開始メッセージ出力                                         *
*---------------------------------------------------------------*
     INITIALIZE                        IF-CHOCO001.
     MOVE  "3"                         TO  共１－イベント種別.
```

③单独某一行的定义，用*--< >定义，<>内的内容和下一条语句对齐，如

```
*--< ＣＰＵ時刻を取得 >
     ACCEPT  Ｗ－システム時刻          FROM  TIME.
*
*--< ＯＲＡＣＬＥ接続 >
     PERFORM  ＯＲＡＣＬＥ接続.
```

09. 共通函数的使用（サブルーチン）

① 在数据部定义共通函数名，例如：

```
*---------------------------------------------------------------*
*    サブルーチン名                                             *
*---------------------------------------------------------------*
 01  CALL-AREA.
*--< 共通ログサブルーチン >
     03  CLOCO001                     PIC  X(08) VALUE "CLOCO001".
```

② 在程序部中调用，例如：

```
     CALL  CLOCO001                USING  IF-CHOCO001.
```

③ 程序执行的时候需要指定共通函数动态库（DLL 文件）所在的路径，即共通函数的 DLL 文件所在的文件夹，或者把 DLL 文件拷贝到当前程序项目所在的文件夹。（在 COBOL85.CBR 中讲解）

10. 程序的头部的描述：包括客户名称、程序名、程序 ID、处理概要、作者、作成日。

```
*****************************************************************
*          ＜ＮＮＮＮＮ＞                                       *
*       1. プログラム名      ：納付先コード変換マスタ移行        *
*       2. プログラム ID     ：COBIS180                          *
*       3. 処理概要          ：提携先指定のコードからＢＯＴＬ納付先 *
*                              のコードの変換テーブル             *
*                                                               *
*                                                               *
*       4. 作成者            ： 任大利                          *
*       5. 作成日            ： 2004.05.26                      *
*****************************************************************
```

11. 程序的主要模块：主要分为三大部分，即“初期处理”、“主处理”、“終了处理”、“エラー判定処理”。

① “初期处理”一般作以下操作：

```
*************************************************************************
*     初期処理                                                          *
*************************************************************************
  初期処理                                SECTION.
  初期処理－ＳＴＡＲＴ.
    1.开始信息出力
    2.作业领域的初期值设定
    3.ORACLE 连接
    4.文件的打开及游标的定义、打开
    5.读取入力文件的第一件目
    …その他処理内容…
  初期処理－ＥＸＩＴ.
      EXIT.
```

② “主处理” 一般作以下操作：

```
*************************************************************************
*     主処理
*************************************************************************
  主処理                                  SECTION.
  主処理－ＳＴＡＲＴ.
    1.编辑处理
    2.出力出力
    3.读取入力文件的下一件目
    …その他処理内容…
  主処理－ＥＸＩＴ.
      EXIT.
```

③ “終了处理” 一般作以下操作：

```
*************************************************************************
*     終了処理                                                          *
*************************************************************************
  終了処理                                SECTION.
  終了処理－ＳＴＡＲＴ.
    1.DB 关闭
    2.文件关闭
    3.件数信息的出力
    4.结束信息的出力
    …その他処理内容…
  終了処理－ＥＸＩＴ.
      EXIT.
```

④ “エラー判定処理”模块：根据错误代码显示出错信息及异常退出程序。

⑤ 其他模块的格式与这些基本相同。

12. 文件操作：

① 环境部的输入输出节的定义：

“U01”表示存取名，这个名称在执行配置文件《COBOL85.CBR》中与一个物理存在的文件名相对应。输入文件对应的存取名是U01、U02、U03等，输出文件对应的存取名是U11、U12、U13等。

“入力ファイル”和“出力ファイル”等表示在程序中引用的文件名称。

“W－状態”表示每次文件操作后的状态，其中ZERO表示正常。

“LINE　SEQUENTIAL”表示这个文件是行顺序文件，即每行一个纪录文件。

```
*****************************************************************
*    ＥＮＶＩＲＯＮＭＥＮＴ          ＤＩＶＩＳＩＯＮ          *
*****************************************************************
 ENVIRONMENT                          DIVISION.
*
 INPUT-OUTPUT                         SECTION.
*
 FILE-CONTROL.
     SELECT     入力ファイル          ASSIGN     TO      U01
     FILE       STATUS      IS        W－状態
     ORGANIZATION           IS        LINE       SEQUENTIAL.
*
     SELECT     出力ファイル          ASSIGN     TO      U11
     FILE       STATUS      IS        W－状態
     ORGANIZATION           IS        LINE       SEQUENTIAL.
```

② 数据部的文件节的定义：

```
*****************************************************************
*    ＤＡＴＡ                    ＤＩＶＩＳＩＯＮ              *
*****************************************************************
 DATA                                 DIVISION.
*
 FILE                                 SECTION.
*---------------------------------------------------------------*
*    入力ファイル                                               *
*---------------------------------------------------------------*
 FD  入力ファイル
     LABEL      RECORD      IS        STANDARD
     BLOCK      CONTAINS    0         RECORDS.
*
 01  入力１－レコード.
     COPY       CISUF470    REPLACING    ==()==     BY   ==入力－==.
*---------------------------------------------------------------*
*    出力ファイル                                               *
*---------------------------------------------------------------*
 FD  出力ファイル
     LABEL      RECORD      IS        STANDARD
     BLOCK      CONTAINS    0         RECORDS.
*
 01  出力－レコード.
     COPY       CISUF470    REPLACING    ==()==     BY   ==出力－==.
```

③ 文件的打开:

输入文件的打开是“OPEN　INPUT　入力ファイル.”;

输出文件的打开是“OPEN　OUTPUT　出力ファイル.”。

对文件的打开要做返回值判断，例如：

```
     OPEN  INPUT    入力ファイル.
*
*--< ファイルオープンの状態判定 >
     EVALUATE  Ｗ－状態
        WHEN  ZERO
           CONTINUE
        WHEN  OTHER
*--<       ファイルオープンエラー >
           MOVE      -1                  TO  Ｗ－エラーコード
           PERFORM  エラー処理
     END-EVALUATE.
```

④ 文件的读入: 对文件的读入要做返回值判断，例如：

```
     READ  入力ファイル
        AT END
           MOVE  "Y"                     TO  Ｗ－終了－フラグ
           GO    TO    ファイル読込処理－ＥＸＩＴ
     END-READ.
*
*--< ファイル読込状態判定 >
     EVALUATE  Ｗ－状態
        WHEN  ZERO
*--<       ファイル入力件数を加算 >
           COMPUTE  Ｗ－入力件数  =  Ｗ－入力件数  +  1
        WHEN  OTHER
*--<       ファイル読込エラー >
           MOVE      -3                  TO  Ｗ－エラーコード
           PERFORM  エラー処理
     END-EVALUATE.
```

⑤ 文件的输出: 对文件的输出要做返回值判断，例如：

```
     WRITE  出力－レコード.
*
*--< ファイル出力の状態判定 >
     EVALUATE  Ｗ－状態
        WHEN  ZERO
*--<       ファイル出力件数の加算 >
           COMPUTE  Ｗ－出力件数  =  Ｗ－出力件数  +  1
        WHEN  OTHER
*--<       ファイル出力エラー >
           MOVE      -6                  TO  Ｗ－エラーコード
           PERFORM  エラー処理
     END-EVALUATE.
```

⑥ 文件的关闭：对文件的关闭不用做返回值判断，例如：

```
CLOSE  入力ファイル.
CLOSE  出力ファイル.
```

13. 访问数据库方法：使用嵌入的 SQL 语句，说明如下：

① 在 SQL 文中使用的变量必须先定义

② COBOL 中的变量名不能有下划线，而 SQL 中的变量名不能有中划线（包括全角和半角）。

③ 在 COBOL 中的 SQL 语句必须以 EXEC SQL 开始，以 END-EXEC 结束。

④ SQL 中的变量名在“EXEC SQL BEGIN DECLARE SECTION END-EXEC.”
和“EXEC SQL END DECLARE SECTION END-EXEC.”之间定义。

⑤ DB 的 COPY 句用“INCLUDE”命令引入。

⑥ “END-EXEC”要上下对齐。

⑦ “SQLCOM.CBL”和“SQLCA.CBL”是连接 ORACLE 用的 COPY 句，在每本 DB 程序中都要引用。

```
*-------------------------------------------------------------*
*--< ホスト変数定義エリア >
*-------------------------------------------------------------*
     EXEC  SQL  BEGIN    DECLARE      SECTION  END-EXEC.
*
01  ＷＳ－ホスト変数.
    03  ＷＳ－取引先コード            PIC  X(08) VALUE "S2917080".
*
*--< ＯＲＡＣＬＥ共通変数 >
     EXEC  SQL  INCLUDE  SQLCOM.CBL            END-EXEC.
*
*--< ＯＲＡＣＬＥ ＳＱＬ実行情報（SQL）>
     EXEC  SQL  INCLUDE  SQLCA.COB             END-EXEC.
*
*--< 転リース提携先コード変換テーブル >
     EXEC  SQL  INCLUDE  IKOTBL004.CBL         END-EXEC.
*
*--< 納付先コード変換マスタ >
     EXEC  SQL  INCLUDE  D082NFH.CBL           END-EXEC.
*
     EXEC  SQL  END      DECLARE      SECTION  END-EXEC.
```

⑧ PCO 文件是操作数据库的源程序，需要预编译成 COB 文件（COBOL 的源程序），然后再编译和连接成可执行的程序文件。

⑨ 预编译文件 PCO 经过 ORACLE 预编译后把 DB 的 COPY 句的内容插入到 COBOL 程序中。

⑩ 纪录数据库操作返回状态的变量有 SQLSTATE（X(05)）、SQLCODE（S9(9)）、SQLERRMC（X(70)），其中，这 3 个变量需要在 SQLCA.COB 中定义。现在我们采用 SQLCODE 判断 SQL 操作的返回值，值为 ZERO 表示正常，100 表示数据没有检索到，或者游标读取结束。（其他值可以参考 Oracle 返回值说明文件）

在数据库操作的 SQL 语句中，COBOL 的变量前面要加字符‘:’。

14. 连结数据库的方法

① 定义取得访问数据库的变量的共通函数：

```
     03  COBC0001                       PIC  X(08) VALUE "COBC0001".
```

② 定义共通函数用的 COPY 句：COPY 句中的“PARA-DBSTRING”是本地访问 Oracle 数据库的服务名、“PARA-USERNAME”是访问 Oracle 数据库的用户名、“PARA-PASSWORD”是密码。

```
*-----------------------------------------------------------------*
*    ＩＮＩファイル読込サブルーチン用パラメタ領域                 *
*-----------------------------------------------------------------*
 01  PARA-AREA.
     COPY CPBC0001.
```

③ 连接 Oracle 数据库：首先调用共通函数“COBC0001”取得本地服务名、用户名和密码，然后根据这些信息连接数据库服务器。因为 COBOL 变量不能在 SQL 语句中使用，因此必须付值给在 SQL 中定义的变量“DB-STRING”、“ USERNAME”、“ PASSWD”，这三个变量在 COPY 句 SQLCOM.COB 中定义。

```
*--< ＩＮＩファイル読込サブルーチン呼び出し >
     CALL COBC0001                  USING  PARA-AREA.
*
     MOVE  PARA-DBSTRING              TO  DB-STRING.
     MOVE  PARA-USERNAME              TO  USERNAME.
     MOVE  PARA-PASSWORD              TO  PASSWD.
*-----------------------------------------------------------------*
*    開始接続                                                     *
*-----------------------------------------------------------------*
     IF  DB-STRING  =  SPACE
        EXEC  SQL
           CONNECT  :USERNAME  IDENTIFIED  BY  :PASSWD
        END-EXEC
     ELSE
        EXEC  SQL
           CONNECT  :USERNAME  IDENTIFIED  BY  :PASSWD
             USING  :DB-STRING
        END-EXEC
     END-IF.
*-----------------------------------------------------------------*
*    接続状態判定                                                 *
*-----------------------------------------------------------------*
     EVALUATE  SQLCODE
        WHEN  定数－ＳＱＬＯＫ
*--<       接続正常 >
           CONTINUE
        WHEN  OTHER
*--<       接続エラー >
           MOVE     -10              TO  Ｗ－エラーコード
           PERFORM  エラー処理
     END-EVALUATE.
```

15. 数据库查询语句

```
     EXEC SQL
        SELECT   新取引先コード
                ,枝番
          INTO :ＩＫＯ００１－新取引先コード
              ,:ＩＫＯ００１－枝番
          FROM  IKOTBL001
         WHERE  取引先コード      = :ＷＳ－原債務者コード
           AND  目的ＣＤ          = '2'
     END-EXEC.
*
*
*----------------------------------------------------------------*
*    検索処理を確認                                              *
*----------------------------------------------------------------*
     EVALUATE  SQLCODE
        WHEN   定数－ＳＱＬＯＫ
*--<        正常の時 >
           CONTINUE
        WHEN   OTHER
*--<        取得しない、初期値をセット >
           MOVE  ZERO          TO  ＩＫＯ００１－新取引先コード
           MOVE  ZERO          TO  ＩＫＯ００１－枝番
           MOVE  -31           TO  Ｗ－エラーコード
           PERFORM   エラー処理
     END-EVALUATE.
```

16. 数据库更新语句

```
     EXEC  SQL
        UPDATE  D411STK_TBL
           SET
               自社売上元本        =  :Ｄ４１１－自社売上元本
              ,自社売上利息        =  :Ｄ４１１－自社売上利息
              ,自社売上元本残高    =  :Ｄ４１１－自社売上元本残高
              ,自社分支払利息適用 =  :Ｄ４１１－自社分支払利息適用
              ,自社分支払利息一般 =  :Ｄ４１１－自社分支払利息一般
*
          WHERE  (契約番号        =  :ＷＳ－契約番号     )
            AND  (再リース回数    =  :ＷＳ－再リース回数)
            AND  (契約種類        =  :ＷＳ－契約種類     )
            AND  (連番            =  :ＷＳ－連番         )
*
     END-EXEC.
*
*---------------------------------------------------------------*
*    ＤＢ更新処理を確認                                         *
*---------------------------------------------------------------*
     EVALUATE  SQLCODE
        WHEN   定数－ＳＱＬＯＫ
*--<        更新成功 >
           COMPUTE  Ｗ－更新件数  =  Ｗ－更新件数  +  1
        WHEN   定数－ＳＱＬＥＮＤ
           CONTINUE
        WHEN   OTHER
*--<        更新失敗 >
           MOVE     -40              TO  Ｗ－エラーコード
           PERFORM  エラー判定処理
     END-EVALUATE.
```

17. 数据库插入语句

```
     EXEC  SQL
        INSERT  INTO  D343HKY_TBL
           (契約番号
           ,再リース回数
           ,契約種類
           ,取引先コード
           ,支払先コード
           ,開始回
           ,開始年月
           ,終了回
           ,終了年月
           ,金額
           ,消費税
           ,登録担当者)
        VALUES
           (:Ｄ３４３－契約番号
           ,:Ｄ３４３－再リース回数
           ,:Ｄ３４３－契約種類
           ,:Ｄ３４３－取引先コード
           ,:Ｄ３４３－支払先コード
           ,:Ｄ３４３－開始回
           ,:Ｄ３４３－開始年月
           ,:Ｄ３４３－終了回
           ,:Ｄ３４３－終了年月
           ,:Ｄ３４３－金額
           ,:Ｄ３４３－消費税
           ,:Ｄ３４３－登録担当者)
     END-EXEC.
*
*-----------------------------------------------------------------*
*    ＤＢ追加処理を確認                                           *
*-----------------------------------------------------------------*
     EVALUATE  SQLCODE
        WHEN   定数－ＳＱＬＯＫ
*--<       追加成功 >
           COMPUTE  Ｗ－追加件数  =   Ｗ－追加件数  +  1
        WHEN   OTHER
*--<       追加失敗 >
           MOVE     -60              TO  Ｗ－エラーコード
           PERFORM  エラー処理
     END-EVALUATE.
```

18. 数据库删除语句

```
      EXEC  SQL
          DELETE
          FROM  M40MAB_TBL
          WHERE  (レコード区分  =  '2')    AND
                 (契約番号      =  :Ｍ０１－契約番号)
      END-EXEC.
*-----------------------------------------------------------------*
*     ＤＢ削除処理確認                                            *
*-----------------------------------------------------------------*
      EVALUATE  SQLCODE
         WHEN   定数－ＳＱＬＯＫ
*--<        追加成功 >
            COMPUTE  Ｗ－削除件数  =   Ｗ－削除件数  +  1
         WHEN   OTHER
*--<        追加失敗 >
            MOVE     -60               TO  Ｗ－エラーコード
            PERFORM  エラー処理
      END-EVALUATE.
```

19. 如果要对数据库表的多行记录进行读取，则需要定义游标(CURSOR)，分为以下四步操作：

```
DECLARE CUR1  CURSOR FOR        *（CURSOR 的定义，即宣言）
OPEN    CUR1                    *（CURSOR 的打开）
FETCH   CUR1                    *（CURSOR 的读取）
CLOSE   CUR1                    *（CURSOR 的关闭）
```

① CURSOR 的定义（即宣言)。不用判断返回值。

```
*-----------------------------------------------------------------*
*    カーソル宣言                                                 *
*-----------------------------------------------------------------*
     EXEC  SQL
        DECLARE  CUR1  CURSOR  FOR
           SELECT  LTRFCK.契約番号
                  ,LTRFCK.事業所コード
                  ,LTRFCK.実行日
                  ,LTRFCK.額面金額
                  ,LTRFCK.銀行コード
                  ,LTRFCK.原債務者コード
                  ,LTRFCK.ユーザーコード
             FROM  LTRFCK
                  ,LTRFCT
            WHERE  LTRFCK.契約番号  =  LTRFCT.契約番号
              AND  LTRFCT.利息期日 >= :ＷＳ－利息期日
     END-EXEC.
```

② CURSOR 的打开。

```
*-----------------------------------------------------------------*
*    カーソルオープン                                             *
*-----------------------------------------------------------------*
     EXEC  SQL
        OPEN  CUR1
     END-EXEC.
*
*-----------------------------------------------------------------*
*    カーソルオープンを確認                                       *
*-----------------------------------------------------------------*
     EVALUATE   SQLCODE
        WHEN    定数－ＳＱＬＯＫ
*--<        正常 >
           CONTINUE
        WHEN   OTHER
*--<        カーソルオープン失敗、プログラムが異常終了 >
           MOVE       -20              TO  Ｗ－エラーコード
           PERFORM    エラー処理
     END-EVALUATE.
```

③ CURSOR 的读取。

```
     EXEC  SQL
        FETCH  CUR1
           INTO
             :ＬＴＲＦＣＫ－契約番号
            ,:ＬＴＲＦＣＫ－事業所コード
            ,:ＬＴＲＦＣＫ－実行日
            ,:ＬＴＲＦＣＫ－額面金額
            ,:ＬＴＲＦＣＫ－銀行コード
            ,:ＬＴＲＦＣＫ－原債務者コード
            ,:ＬＴＲＦＣＫ－ユーザーコード
     END-EXEC.
*
*-----------------------------------------------------------------*
*    ファクタリング読込確認                                       *
*-----------------------------------------------------------------*
     EVALUATE  SQLCODE
        WHEN  定数－ＳＱＬＯＫ
*--<        正常 >
           COMPUTE   Ｗ－入力－件数  =  Ｗ－入力－件数  +  1
        WHEN  定数－ＳＱＬＥＮＤ
*--<        読込終了 >
           MOVE      "Y"             TO  Ｗ－終了－フラグ
        WHEN  OTHER
*--<        読込エラー >
           MOVE      -21             TO  Ｗ－エラーコード
           PERFORM   エラー処理
     END-EVALUATE.
```

④ CURSOR 的关闭。

```
*--< カーソルのクローズ >
     EXEC  SQL
        CLOSE  CUR1
     END-EXEC.
```

20. 数据库表结合操作：当多个表结合读取的时候，一般只读取条件都满足的数据行；但是在实际开发中还有这样的情况，即有一个表是主输入数据，其它表是辅助数据，要求主输入数据必须读入，而辅助数据可以没有，例如假设有 A 表和 B 表。

① A 表是主数据必须读入，B 表满足条件时正常读入；不满足条件时设置为初期值。

```
     EXEC  SQL
        DECLARE  CUR1  CURSOR  FOR
           SELECT  A.契約番号
                  ,A.期限前決裁日
                  ,NVL(B.ユーザーコード,'00000000')
                  ,NVL(B.額面金額,0)
           FROM   A
                 ,B
          WHERE   A.契約番号  =  B.契約番号(+)
     END-EXEC.
```

② B 表是主数据必须读入，A 表满足条件时正常读入；不满足条件时设置为初期值。

```
EXEC  SQL
   DECLARE  CUR1  CURSOR  FOR
      SELECT  NVL(A.契約番号,'000000000')
              ,NVL(A.期限前決裁日,'0000000')
              ,B.ユーザーコード
              ,B.額面金額
      FROM    A
             ,B
     WHERE    A.契約番号(+)  =  B.契約番号
END-EXEC.
```

21. 事物（TRANSACTION）操作：确认(COMMIT)或放弃(ROLLBACK)对数据库的修改。(在 CURSOR 关闭以后进行，另外只进行查询操作的程序不用这两个操作)

① 确认（即保存）对数据库的修改。(事物的提交，不用判断返回值)

```
*--< コミット処理 >
     EXEC  SQL
        COMMIT  WORK  RELEASE
     END-EXEC.
```

② 取消对数据库的修改。(事物的回退，不用判断返回值)

```
*--< ロールバック処理 >
     EXEC  SQL
         ROLLBACK  WORK  RELEASE
     END-EXEC.
```

22. 打印方法的实现：同普通 PS 文件的操作方法基本一样。

① 环境部的配置节的定义：

```
CONFIGURATION                          SECTION.
*
 SPECIAL-NAMES.
*--< 制御レコード >
     CTL   IS   PAGE-CNTL
```

② 环境部的输入输出节的定义：

```
 INPUT-OUTPUT                          SECTION.
 FILE-CONTROL.
     SELECT          出力ファイル       ASSIGN     TO    PRINTER
     FILE            STATUS      IS     Ｗ－状態
     ORGANIZATION    IS                 SEQUENTIAL.
```

③ 数据部的文件节的定义:（其中“行レコード”用于打印明细行用;“制御レコード”控制打印机的一些信息，在后面详细说明）

```
FD   出力ファイル.
01   行レコード                        PIC   N(136).
01   注釈レコード                      PIC   N(050).
01   制御レコード                      PIC   X(100).
```

④ 打印字体大小和打印位置的定义：

“PRINTING POSITION IS 3”：表示在当前行打印位置。

“MODE-2”：表示字体大小，有“MODE-2”、“ MODE-3”等。“MODE-2”比“MODE-2”字体大。

```
          05   Ｐ－前受区分              PRINTING  POSITION  IS  3
                                         PIC  N(06)  MODE-2.
```

⑤　数据部的工作节的定义。

```
*-----<Ｉ制御レコードのデータ宣言>
 01　Ｉ制御データ.
   03　レコード識別子　　PIC X(002) VALUE "I1".
   03　モード　　PIC X(001) VALUE "1".
   03　オーバレイ.
     05　オーバレイ名　　PIC X(004) VALUE SPACE.
     05　焼付け回数　　PIC 9(003) VALUE ZERO.
   03　複写数　　PIC 9(003) VALUE ZERO.
   03　ＦＣＢ名　　PIC X(004) VALUE SPACE.
   03　帳票定義体名　　PIC X(008) VALUE SPACE.
   03　　PIC X(030) VALUE SPACE.
   03　印刷形式　　PIC X(002) VALUE SPACE.
   03　用紙サイズ　　PIC X(003) VALUE SPACE.
   03　　PIC X(004) VALUE SPACE.
   03　印刷面　　PIC X(001) VALUE SPACE.
   03　印刷開始位置づけ面 PIC X(001) VALUE SPACE.
   03　印字禁止域　　PIC X(001) VALUE SPACE.
   03　綴じ代方向.
     05　ポート時表面　　PIC X(001) VALUE SPACE.
     05　ポート時裏面　　PIC X(001) VALUE SPACE.
     05　ランド時表面　　PIC X(001) VALUE SPACE.
     05　ランド時裏面　　PIC X(001) VALUE SPACE.
   03　綴じ代幅　　PIC X(004) VALUE SPACE.
   03　印字開始原点位置.
     05　表面Ｘ座標　　PIC X(004) VALUE SPACE.
     05　表面Ｙ座標　　PIC X(004) VALUE SPACE.
     05　裏面Ｘ座標　　PIC X(004) VALUE SPACE.
     05　裏面Ｙ座標　　PIC X(004) VALUE SPACE.
   03　文書情報　　PIC X(004) VALUE SPACE.
   03　予約域　　PIC X(005) VALUE SPACE.
```

⑥　打印的控制项目的编辑

```
*-------------------------------------------------------------*
*     制御レコード設定
*-------------------------------------------------------------*
*----<オーバレイ名”ＳＦ１９”（ＫＯＬ５ＳＦ１９．ＯＶＤ）設定>
      MOVE ″SF19″ TO  オーバレイ名.
*----<焼き付け回数１回設定>
      MOVE 1       TO  焼付け回数.
*----<複写枚数１枚設定>
      MOVE 1       TO  複写数.
*----<ＦＣＢ名”Ａ４Ｌ６”（ＦＣＢＡ４Ｌ６）設定>
      MOVE ″LPI6″ TO  ＦＣＢ名.
*----<印刷形式ランドスケープモード(横向き)設定>
      MOVE ″L″    TO  印刷形式.
*----<用紙サイズＢ４設定>
      MOVE ″B4″   TO  用紙サイズ.
*----<両面印刷設定>
      MOVE ″B″    TO  印刷面.
*----<表面位置づけ設定>
      MOVE ″F″    TO  印刷開始位置づけ面.
*----<ポートレートモード表面は左とじ>
      MOVE ″L″    TO  ポート時表面.
*----<ポートレートモード裏面は左とじ>
      MOVE ″L″    TO  ポート時裏面.
*----<ランドスケープモード表面は上とじ>
      MOVE ″U″    TO  ランド時表面.
*----<ランドスケープモード裏面は下とじ>
      MOVE ″D″    TO  ランド時裏面.
*----<文書情報名”ＤＯＣ１”(@ＣＢＲ__ＤｏｃｕｍｅｎｔＮａｍｅ__ＤＯＣ１）設定>
      MOVE ″DOC1″ TO 文書情報.


* Ｉ制御レコードを出力することによりページ属性を設定
      WRITE 制御レコード FROM Ｉ制御データ
                         AFTER ADVANCING PAGE-CNTL.
```

⑦ 打开操作：同 PS 文件的打开方法，“OPEN　OUTPUT　出力ファイル.”。

⑧ 输出操作：

```
*--< Ｉ制御レコードを出力することによりページ属性を設定 >
     WRITE 制御レコード              FROM   Ｉ制御データ
                                     AFTER  ADVANCING  PAGE-CNTL.

 (印刷) 出力１行
     WRITE   行レコード  AFTER  ADVANCING  1  LINES
     COMPUTE 行ー件数    =  行ー件数  +  1

 (印刷) 改ページ
     WRITE  行レコード  FROM  Ｐ－見出し  AFTER  ADVANCING  PAGE
     COMPUTE ページー件数     =  ページー件数  +  1
```

⑨ 文件的关闭：同 PS 文件的关闭方法。

23. 程序日志信息的显示方法：COPY 句是“CHOCO001”，共通函数是“CLOCO001”。信息显示在系统日志中。

① 共通函数的定义：

```
01  CALL-AREA.
*--< 共通ログサブルーチン >
     03  CLOCO001                      PIC  X(08) VALUE ″CLOCO001″.
```

② COPY 句的定义：

```
*--< 共通ログ用パラメータ >
 01  IF-CHOCO001.
     COPY  CHOCO001  REPLACING  ==()==  BY  ==共通ー==.
```

③ COPY 句项目的定义：

“共通ーｲﾍﾞﾝﾄ種別”：显示信息的种类，“1”表示错误（红色）；“2”表示警告（黄色）；“3”表示正常的信息（蓝色）；

“共通ーｿｰｽＩＤ”：显示数据库表名或者文件名 ID。

“共通ー復帰ｺｰﾄﾞ”：返回值，错误和警告的时候是 9；正常信息的时候是 0。

“共通ー処理識別”：当前进行的操作，例如打开文件是“OPEN”；数据库检索是“SELECT”；件数信息是“COUNT”，等等。

“共１ーその他ﾒｯｾｰｼﾞ”：详细的文字信息，例如数据库的出错信息；件数信息。

④ 开始信息的显示

```
*-----------------------------------------------------------------*
*    開始メッセージ出力                                           *
*-----------------------------------------------------------------*
     INITIALIZE                      IF-CHOCO001.
     MOVE  ″3″                       TO  共通ーイベント種別.
     MOVE  定数ープログラムＩＤ      TO  共通ーソースＩＤ.
     MOVE  ZERO                      TO  共通ー復帰コード.
     MOVE  ″START″                   TO  共通ー処理識別.
     MOVE  定数ープログラム名        TO  共通ーその他メッセージ.
     CALL  CLOCO001                USING  IF-CHOCO001.
```

⑤ 入出力件数表示

```
*-------------------------------------------------------------*
*    ファクタリング明細読込件数                               *
*-------------------------------------------------------------*
     INITIALIZE                      IF-CHOCO001.
     MOVE  "3"                       TO  共通－イベント種別.
     MOVE  定数－プログラムＩＤ      TO  共通－ソースＩＤ.
     MOVE  ZERO                      TO  共通－復帰コード.
     MOVE  "LTRFCT"                  TO  共通－処理テーブルＩＤ.
     MOVE  "COUNT"                   TO  共通－処理識別.
     MOVE  Ｗ－入力－件数            TO  共通－データ内容.
     MOVE  "ファクタリング明細読込件数"
                                     TO  共１－その他メッセージ.
     CALL  CLOCO001                USING  IF-CHOCO001.
*
*-------------------------------------------------------------*
*    受取手形出力件数                                         *
*-------------------------------------------------------------*
     INITIALIZE                      IF-CHOCO001.
     MOVE  "3"                       TO  共通－イベント種別.
     MOVE  定数－プログラムＩＤ      TO  共通－ソースＩＤ.
     MOVE  ZERO                      TO  共通－復帰コード.
     MOVE  "D642UTG"                 TO  共通－処理テーブルＩＤ.
     MOVE  "COUNT"                   TO  共通－処理識別.
     MOVE  Ｗ－出力－件数            TO  共通－データ内容.
     MOVE  "受取手形出力件数"        TO  共通－その他メッセージ.
     CALL  CLOCO001                USING  IF-CHOCO001.
```

⑥ 结束信息的显示

```
*-------------------------------------------------------------*
*    終了メッセージ 出力                                      *
*-------------------------------------------------------------*
     INITIALIZE                      IF-CHOCO001.
     MOVE  "3"                       TO  共通－イベント種別.
     MOVE  定数－プログラムＩＤ      TO  共通－ソースＩＤ.
     MOVE  ZERO                      TO  共通－復帰コード.
     MOVE  "END"                     TO  共通－処理識別.
     MOVE  定数－プログラム名        TO  共通－その他メッセージ.
     CALL  CLOCO001                USING  IF-CHOCO001.
```

24. 程序中变量名的定义方法

① ＷＯＲＫ－エリア的定义部分的变量名以“Ｗ－”开始。

② SQL 的变量的定义以“ＷＳ－”开始。

③ 定量部分的变量的定义以“定数－”开始。

25. 错误处理：

① 在各个出错处理部分设置出错代码，然后调用错误处理模块，显示错误信息。

```
MOVE     -1                TO  Ｗ－エラーコード
PERFORM  エラー処理
```

② 错误处理模块：

对于要异常退出的错误，才作显示错误信息、事务回退(注意没有数据库操作的程序不用调用这个函数)、显示件数和结束信息、设置错误返回码；

而不异常退出的错误只显示错误信息即可。

```
*****************************************************************
*    エラー判定処理                                             *
*****************************************************************
 エラー判定処理                        SECTION.
 エラー判定処理－ＳＴＡＲＴ.
*
     MOVE  "Y"                         TO  Ｗ－異常終了－フラグ.
     INITIALIZE                        IF-CHOCO001.
*
     EVALUATE  Ｗ－エラーコード
        WHEN  -1
*--<        コードマスタオープンエラー>
           MOVE  "1"                   TO  共通－イベント種別
           MOVE  定数－プログラムＩＤ  TO  共通－ソースＩＤ
           MOVE  "9"                   TO  共通－復帰コード
           MOVE  "FFUIS230"            TO  共通－処理テーブルＩＤ
           MOVE  "OPEN"                TO  共通－処理識別
           MOVE  Ｗ－状態              TO  共通－データ内容
           MOVE  "コードマスタオープンエラー"
                                       TO  共通－その他メッセージ
           CALL  CLOCO001          USING  IF-CHOCO001
        WHEN  -20
*--<        転リーステーブル読込失敗 >
           MOVE  "2"                   TO  共１－イベント種別
           MOVE  定数－プログラムＩＤ  TO  共１－ソースＩＤ
           MOVE  "9"                   TO  共１－復帰コード
           MOVE  "IKO004"              TO  共１－処理テーブルＩＤ
           MOVE  "SELECT"              TO  共１－処理識別
           MOVE  ＷＳ－取引先コード    TO  共１－キー情報
           MOVE  SQLCODE               TO  共１－データ内容
           MOVE  SQLERRMC              TO  共１－その他メッセージ
           CALL  CLOCO001          USING  IF-CHOCO001
           MOVE  "N"                   TO  Ｗ－異常終了－フラグ
        WHEN  OTHER
           MOVE  "N"                   TO  Ｗ－異常終了－フラグ
     END-EVALUATE.
```

接下一页…

```
*
     IF   Ｗ－異常終了－フラグ        =  "Y"
        PERFORM   ＤＢロールバック処理
*
        PERFORM   終了メッセージ出力処理
*
*--<     プログラム異常終了のリターンコード >
        MOVE   定数－異常状態              TO   PROGRAM-STATUS
*
        EXIT   PROGRAM
     END-IF.
```

26. 其它注意部分

① 变量的初始化。

```
     MOVE   SPACE                    TO   ＷＯＲＫ－エリア.
     INITIALIZE                           ＷＯＲＫ－エリア.
```

② 输出记录的初期化，先清空一个初期化用的纪录，然后在编辑每个输出记录前，把这个初期化用的纪录直接付值给输出记录即可。（初期化时一般初期化集团项目）

```
     MOVE   SPACE                    TO   初期化レコード.
     INITIALIZE                           初期化レコード.
.........
*--< 出力レコードの初期化 >
     MOVE   初期化レコード              TO   出力－レコード.
```

③ 输出记录在每回输出前要进行初始化。

④ 给变量付值时，空白字符用关键字‘SPACE’代替表示；数值型或者字符型 0 用关键字‘ZERO’ 代替表示。给 9(5)付 ZERO，值是 00000；给 X(5)付 ZERO，值是‘00000’； 给 S9(5)付 ZERO，值是 0。

⑤ N 型字符串付值时，如果前面有 NC 关键字，则后面的字符串中不能有半角的字符。例如：NC"前受情報（前受金）当月分作成処理"

⑥ 编写代码时，每行的变量间隔两个字符；对于“IF”、“EVALUATE”等带有多行语句，下一行语句要向右缩进 3 个字符。

⑦ 变量定义的“PIC”语句，和付值时的“TO”语句，要求对齐在 45 列，整体上要求整齐美观。

⑧ 每行代码（包括 COPY 句的代码）不要超过 73 列，否则编译的时候可能出现错误。

⑨ 件数等数值计算用 COMPUTE 语句，例如“COMPUTE Ｗ－入力件数 = Ｗ－入力件数 + 1”。

⑩ 对于某个变量有多个值的情况的判断，可以写成如下形式：

```
     IF   Ｄ３１－債務協調コード   =  (45  OR  55)
        MOVE  1                        TO   Ｄ４１２－購入区分
     ELSE
        MOVE  0                        TO   Ｄ４１２－購入区分
     END-IF.
```

27. 系统时间的取得

① 系统日期和时间的取得方法。其中“年月日”是六位，没有世纪，所以在编程的时候要注意加上

2 位的世纪；“年日”是 2 位的年，加上从 1 月 1 日开始到今天经过的天数；另外这些变量也可以定义成 X 型。

```
  01  年月日     PIC 9(6).
  01  年日       PIC 9(5).
  01  曜日       PIC 9(1).
  01  時刻       PIC 9(8).
 PROCEDURE DIVISION.
     ACCEPT   年月日     FROM DATE.
     ACCEPT   年日       FROM DAY.
     ACCEPT   曜日       FROM DAY-OF-WEEK.
     ACCEPT   時刻       FROM TIME.
```

② 取系统时间的操作一般在“初期处理”模块中进行。

③ 编程的时候要注意模块的划分：因为大的，复杂的代码不容易看懂，如果划分成小的模块有助于理解和维护；另外有些在多个地方调用的代码也要作为一个模块，例如文件读入模块等。

④ 模块名最好能反映模块的功能：例如“編集処理”模块是编辑数据，而“編集出力処理”表示不但有编辑数据的功能，还有出力数据的功能。

⑤ 编辑数据项时要在每个编辑项的上面加上项目编号的注释，这样不会漏掉编辑项，而且有利于评审和维护。

⑥ 对文件操作和数据库操作的返回值判定一般用“EVALUATE”语句。

```
*-----------------------------------------------------------------*
*     追加処理を確認                                              *
*-----------------------------------------------------------------*
     EVALUATE  SQLCODE
        WHEN   定数－ＳＱＬＯＫ
*--<          追加成功 >
           COMPUTE  Ｗ－出力－件数  =  Ｗ－出力－件数  +  1
        WHEN   OTHER
*--<          追加処理エラー >
           MOVE     -40                TO  Ｗ－エラーコード
           PERFORM  エラー判定処理
     END-EVALUATE.
```

⑦ 对于大的复杂的程序，可以编辑一部分就开始编译，避免最后程序不好编译和调试。