



# 中华人民共和国国家标准

GB/T 4092.5—1992

---

## 程序设计语言 COBOL 相对 I-O 模块

Programming language COBOL  
Relative I-O module

1992-08-04 发布

1993-05-01 实施

---

国家技术监督局 发布

程序设计语言 COBOL  
相对 I-O 模块

GB/T 4092.5—1992

Programming language COBOL  
Relative I-O module

代替 GB 4092.5—1983

## 1 引言

### 1.1 功能

相对 I-O 模块提供以随机方式或顺序方式存取海量存储器上的文卷的记录的功能。相对文卷中的每一个记录都用一个大于零的整数唯一地标识,该整数指明文卷中记录的逻辑顺序位置。

### 1.2 级别特征

1 级相对 I-O 对文卷控制款、文卷描述款和 I-O CONTROL 段中的款提供局部功能。在过程部中,1 级相对 I-O 对 CLOSE、OPEN、READ、REWRITE、USE 和 WRITE 语句提供局部功能,而对 DELETE 提供完整功能。

2 级相对 I-O 对文卷控制款、文卷描述款及 I-OCNTR0L 段中各款提供了完整功能。在过程部中,2 级相对 I-O 对 CLOSE、DELETE、OPEN、READ、REWRITE、START、USE 和 WRITE 语句提供了完整功能。

### 1.3 语言概念

#### 1.3.1 组织

相对组织文卷是一个海量存储文卷。任何记录可以通过提供相对记录号的值来存储或检索。

这种文卷也可以认为是由一串区域组成。每一个区域能存放一个逻辑记录。每个区域由一个相对记录号命名。根据这个号存储和检索记录。例如,第 10 个记录是相对记录号为 10 且在第 10 个记录区域中的一个记录。不管第 1 到第 9 个记录区域是否已写过。

为了在相对文卷中更有效地存取记录,在媒体上为存储一个特定逻辑记录而保留的字符位置数可以不同于在程序中描述的那个记录的字符位置数。

#### 1.3.2 存取方式

对于相对组织,在其顺序存取方式中,记录存取的次序就是该文卷中所有现存记录的相对记录号的递增次序。只有当前存在于文卷中的记录是有效的。

在随机存取方式中,输入输出语句用于按程序员指定的次序存取记录。要存取所需的记录就把它的相对记录号存放在相对键数据项中。

#### 1.3.3 文卷位置指示符

文卷位置指示符是本标准中为了便于在某一输入输出操作中指明给定的文卷中要存取的下一个记录而使用的一个概念实体。文卷位置指示符概念对按输出或延伸方式打开的文卷是无意义的。文卷位

置指示符的值只受 **CLOSE**、**OPEN** 和 **READ** 语句的影响。

### 1.3.4 I-O 状态

**I-O** 状态是一个两字符的概念实体。赋给它的值指明了 **CLOSE**、**DELETE**、**OPEN**、**READ**、**REWRITE** 或 **WRITE** 语句执行期间的状态,指明了与该 **I-O** 语句相联系的任一命令语句执行之前的状态,或任一可应用的 **USE AFTER STANDARD EXCEPTION** 过程执行之前的状态。通过使用该文卷的文卷控制款中的 **FILE STATUS** 子句,使 **I-O** 状态值对 **COBOL** 程序是可用的。

**I-O** 状态也决定是否执行一个可用的 **USE AFTER STANDARD EXCEPTION** 过程。如果不是包含在标题“成功的结束”下的任何条件,则这个过程可以根据其它地方所述的规则执行。如果是在标题“成功的结束”下列出的一个条件,则不执行这种过程(见 4.6 **USE** 语句)。

某些类的 **I-O** 状态值指出关键错误条件。它们是:任何以数字 3 或 4 和任何由实现者作为关键定义的以数字 9 开始的错误条件。如果输入输出操作的 **I-O** 状态值指出这种错误条件,那么实现者确定在执行任何可用的 **USE AFTER STANDARD EXCEPTION** 过程之后所要采取的动作;或者若没有可用的上述过程,则实现者确定输入输出控制系统错误标准处理之后采取什么动作。

**I-O** 状态根据输入输出操作完成的情况表示下列条件之一:

- (1) 成功的结束。成功地执行输入输出语句。
- (2) 到末端。作为末端条件的结果,顺序的 **READ** 语句执行不成功。
- (3) 无效键。作为一个无效条件的结果,输入输出语句执行不成功。
- (4) 永久性错误。由于发生了阻止文卷的进一步处理的错误,使输入输出语句执行不成功,且执行指定的例外过程。除非调用实现者定义的改正永久性错误条件的技术,否则在余下的输入输出操作中该条件一直有效。
- (5) 逻辑错误。作为一种不适当的对文卷执行的输入输出操作结果或作为违反用户定义的限制的结果,输入输出语句执行不成功。
- (6) 实现者定义。由于满足了实现者指定的条件,输入输出语句执行不成功。

下面列出上述条件的 **I-O** 状态值,这些条件是在对一相对文卷进行输入输出操作之后产生的。如果一个以上的值适用,由实现者决定哪一个可用的值放入 **I-O** 状态。

#### (1) 成功的结束

- a. **I-O** 状态=00。输入输出语句执行成功,对有关输入输出操作没有进一步的信息可用。
- b. **I-O** 状态=04。 **READ** 语句执行成功,但被处理的记录长度与相应文卷的固有文卷属性不一致。
- c. **I-O** 状态=05。成功执行了 **OPEN** 语句,但在执行 **OPEN** 时,引用的任选文卷没有出现。如果打开方式是 **I-O** 或延伸方式,则文卷已被建立。

#### (2) 不成功结束的末端条件

- a. **I-O** 状态=10。试图执行一个顺序 **READ** 语句,且文卷中不存在下一个逻辑记录。因为:
  - 1) 已到文卷末尾;或

b. **I-O** 状态=14。对一个相对文卷试图执行一个顺序的 **READ** 语句,而相对记录号中有效数字的位数大于为该文卷描述的相对键数据项的长度。

#### (3) 不成功结束的无效键条件

- a. **I-O** 状态=22。试图写一个在相对文卷上建立一个重复键的记录。
- b. **I-O** 状态=23。此条件的存在是因为:
  - 1) 试图随机地存取一个文卷中不存在的记录;或

c. I-O 状态=24。试图写到相对文卷定义的界限之外。实现者规定定义界限的方法,或者试图对相对文卷执行顺序的 **WRITE** 语句且相对记录号中有效数字的位数大于为文卷描述的相对键数据项的长度。

(4) 不成功结束的永久性错误条件

a. I-O 状态=30。存在一个永久性错误并且没有和输入输出操作有关的进一步可用信息。

b. I-O 状态=35。存在一个永久性错误,因为试图对一个不出现的 **DELETE** 文卷执行带有 **INPUT**, **I-O** **REWRITE** 短语的 **OPEN** 语句。

c. I-O 状态=37。存在一个永久性错误,因为试图执行一个 **OPEN** 语句且那个文卷不支持 **OPEN** 语句中规定的打开方式。可能的出错情形是:

1) 规定了 **OUTPUT** 短语,但文卷不支持写操作。

2) 规定了 **I-O** 短语,但文卷不支持以 **I-O** 方式打开的相对文卷所允许的输入输出操作。

3) 规定了 **INPUT** 短语,但文卷不支持读操作。

e. I-O 状态=39。**OPEN** 语句执行不成功,因为固有文卷属性和在程序中为文卷规定的属性之间相矛盾。

(5) 不成功结束的逻辑错误条件

a. I-O 状态=41。试图对已打开的文卷执行 **OPEN** 语句。

b. I-O 状态=42。试图对处于非打开状态的文卷执行 **CLOSE** 语句。

c. I-O 状态=43。在顺序存取方式下,在执行 **DELETE** 或 **REWRITE** 语句之前所执行的最后一次输入输出语句不是成功的 **READ** 语句。

d. I-O 状态=44。违反边界,因为:

1) 试图写或重写比有关文卷名规定的 **RECORD IS VARYING** 子句所允许的最大记录要大,或比最小记录要小的记录。

2) 在 1 级中试图重写一个记录到相对文卷中且此记录与被代替的记录长度不一致。

e. I-O 状态=46。试图对一个以输入方式或 **I-O** 方式打开的文卷执行顺序的 **READ** 语句且还未建立有效的下一个记录。因为:

2) 先前的 **READ** 语句执行不成功但未引起末端条件,或

3) 先前的 **READ** 语句引起一个末端条件。

f. I-O 状态=47。试图对一个不是以输入或 **I-O** 方式打开的文卷执行一个 **READ**

g. I-O 状态=48。试图对一个不是以 **I-O**,输出 **WRITE** 方式打开的文卷执行 **WRITE** 语句。

h. I-O 状态=49。对一个不是以 **I-O** 方式打开的文卷试图执行 **DELETE** 或 **REWRITE** 语句。

(6) 实现者定义的不成功结束条件

a. I-O 状态=9X。存在一个实现者定义的条件。这个条件不能与 **I-O** 状态值 00 到 49 的任何一个条件重复。X 的值由实现者定义。

### 1.3.5 无效键条件

**INVALID KEY** 条件可能作为 **READ**, **WRITE**, **REWRITE** 或 **DELETE** 语句的执行结果出现。当无效键条件出现时,识别出条件的输入输出语句的执行是不成功的且文卷不受影响(见 4.3 **DELETE** 语句; 4.5 **READ** 语句; 4.6 **REWRITE** 语句; 4.7 **START** 语句; 4.9 **WRITE** 语句)。

在执行一个输入输出语句规定的输入输出操作之后如果出现无效键条件,按下面给出的次序进行

处理:

(1) 把值置入有关语句文卷连接区的 **I-O** 状态,以指明一个无效键条件(见 1.3.4I-O 状态)。

(2) 若 **INVALID KEY** 短语在产生该条件的语句中指明,则将控制转移到 **INVALID KEY** 命令语句,对该文卷指明的任何 **USE AFTER EXCEPTION** 过程都不执行。然后根据命令语句规定的每个语句的规则继续执行。如果一个分支过程或引起显式控制转移的条件语句执行时,则根据那个语句的规则控制转移;否则,直到完成执行 **INVALID KEY** 短语中规定的命令语句,控制转移到输入输出语句的结束处且忽略 **NOT INVALID KEY** 短语(如果已指明的话)。

(3) 如果在输入输出语句中未规定 **INVALID KEY** 短语,一个 **USE AFTER EXCEPTION** 过程必须与文卷连接区相联系,执行那个过程并且根据 **USE** 语句的规则进行控制转移。如果规定了 **NOT INVALID** 短语则忽略它(见 4.8USE 语句)。

如果在执行了由输入输出语句规定的输入输出操作之后不存在无效键条件,则忽略已规定的 **INVALID KEY** 短语。与语句有关的文卷连接区的 **I-O** 状态更新且发生下列动作:

(1) 如果存在一个非有效键条件的例外条件,在执行任何一个与文卷连接符有关的 **USE AFTER EXCEPTION** 过程之后,按 **USE** 语句的规则进行控制转移(见 4.8USE 语句)。

(2) 如果不存在例外条件,控制转移到输入输出语句的结束处或转到在 **NOT INVALID KEY** 短语中规定的(如果已指明的话)命令语句。在后一种情况,根据在那个命令语句中规定的每一个语句的规则继续执行。如果执行一个分支过程或执行引起显式控制转移的条件语句,则根据那个语句的规则进行控制转移;否则,直到完成执行 **NOT INVALID KEY** 短语中的命令语句,控制才转移到输入输出语句的结束处。

### 1.3.6 末端条件

末端条件可作为 **READ** 语句执行的结果。产生该条件的细节,见 4.5READ 语句。

### 1.3.7 文卷属性冲突条件

执行 **OPEN**、**REWRITE** 或 **WRITE** 语句可能出现文卷属性冲突条件。当发生文卷属性冲突条件时,识别出此条件的输入输出语句执行是不成功的,文卷不受影响(见 4.4OPEN 语句;4.6REWRITE 语句;4.9WRITE 语句)。

当识别到文卷属性冲突条件时,发生下列次序的动作:

(1) 对与文卷名有关的 **I-O** 状态赋值以指出文卷属性冲突条件(见 1.3.4I-O 状态)。

(2) 执行与文卷名有关的 **USE AFTER EXCEPTION** 过程(如果有)。

## 2 相对 I-O 模块的环境部

### 2.1 输入-输出节

与输入输出节有关的信息见顺序 **I-O** 模块的 2.1。

### 2.2 FILE-CONTROL 段

与 **FILE-CONTROL** 段与有关的信息见顺序 **I-O** 模块的 2.2。

### 2.3 文卷控制款

#### 2.3.1 功能

文卷控制款说明一个相对文卷的有关物理属性。

#### 2.3.2 一般格式

```

SELECT OPTIONAL 文卷名 1
      ASSIGN TO { 实现名 1 } ...
               { 字值 1 }
RESERVE 整数 1 AREA
                  AREAS

```

[ORGANIZATION IS]RELATIVE

[ACCESS MODE IS {SEQUENTIAL [RELATIVE KEY IS 数据名 1]  
 {RANDOM }  
 {INDEXED }RELATIVE KEY IS 数据名 1 } ]

[FILE STATUS IS 数据名 2].

### 2.3.3 语法规则

(1) **SELECT** 子句必须在文卷控制款中首先指出。接在 **SELECT** 子句后的子句可按任意次序出现。

(2) 数据部描述的每一个文卷必须且只能在 **FILE-CONTROL** 段命名一次,在 **SELECT** 子句中指出的每个文卷必须在同一程序的数据部中有一个文卷描述款。

(3) 字值 1 必须是一个非数值字值且不能是象征常量。实现名 1 允许的内容的意义及规则及字值 1 的值由实现者来定义。

### 2.3.4 一般规则

(1) 如果文卷名 1 引用的文卷连接符是外部文卷连接符(见程序间通信模块 4.5**EXTERNAL** 子句),则运行单位中引用此文卷连接符的所有文卷控制款必须:

a. 对 **ASSIGN** 子句中实现名 1 或字值 1 有一致的说明。实现者对实现名 1 或字值 1 指定一致规则。

b. 对 **ASSIGN** 子句中实现名 1 或字值 1 有一致的说明。实现者对实现名 1 或字值 1 指定一致规则。

c. 对 **ASSIGN** 子句中实现名 1 或字值 1 有一致的说明。实现者对实现名 1 或字值 1 指定一致规则。

d. 同样的组织。

e. 同样的存取方式。

f. 在 **RELATIVE KEY** 短语中的数据名 1 有同样的外部数据项。

(2) 外部媒体上的数据项使用本源字符集。

(3) **OPTIONAL** 短语仅用于以输入、I-O 或扩展方式打开的文卷。对目标程序每一次运行并不都要用到的文卷来说,该短语是需要的。

(4) **ASSIGN** 子句指明由文卷名 1 引用的文卷到由实现名 1 或字值 1 引用的存储媒体之间的联系。

(5) 相对 I-O 模块的 **RESERVE** 子句与顺序 I-O 模块的 **RESERVE** 子句是一样的。因此对 **RESERVE** 子句的规定见顺序 I-O 模块中的 2.9。

(6) 相对 I-O 模块的 **FILE STATUS** 子句与顺序 I-O 模块的 **FILE STATUS** 子句是一样的。因此,有关 **FILE STATUS** 子句有关内容见顺序 I-O 模块。相对文卷的 **FILE STATUS** 子句数据项的内容见 1.3.4I-O 状态。

(7) **ACCESS MODE** 子句和 **ORGANIZATION IS RELATIVE** 子句在下面列出。

## 2.4 ACCESS MODE 子句

### 2.4.1 功能

**ACCESS MODE** 子句规定了文卷中记录的存取次序。

### 2.4.2 一般格式

ACCESS MODE IS {SEQUENTIAL [RELATIVE KEY IS 数据名 1]  
 {RANDOM }  
 {INDEXED }RELATIVE KEY IS 数据名 1 }

### 2.4.3 语法规则

- (1) 数据名 1 可以受限。
- (2) 数据名 1 必须引用一个其描述不含 **PICTURE** 符号 ‘P’ 的无正负号整数数据项。
- (3) 数据名 1 不能在有关文卷名的记录描述款中定义。
- (4) **ACCESS MODE IS RANDOM** 子句不能指定为 **SORT** 或 **MERGE** 语句中的 **USING** 或 **GIVING** 短语中指定的文卷名。

(5) 如果相对文卷由一个 **START** 语句引用,则必须为那个文卷规定 **ACCESS MODE** 子句内的 **RELATIVE KEY** 短语。

#### 2.4.4 一般规则

- (1) 如果未指明 **ACCESS MODE** 子句,则假定为顺序存取。
- (2) 如果存取方式是顺序的,则文卷中的记录按照文卷组织指定的顺序存取。对相对文卷,这种顺序是文卷中已存在记录的相对记录号的升序。
- (3) 如果存取方式是随机的,相对文卷的相对键数据项的值指明了要存取的记录。

(4) 如果存取方式是顺序的,则文卷中的记录按照顺序和随机存取。

- (5) 所有在相对文卷中存储的记录由相对记录号唯一地标识。给出一个记录的相对记录号规定了在文卷中记录的逻辑次序位置。第一个逻辑记录有相对记录号 1,后继的逻辑记录有相对记录号 2、3、4、...

- (6) 由数据名 1 规定的用于在用户和海量存储控制系统(MSCS)之间对相对记录号进行通信。

- (7) 与执行输入输出语句有关的相对键数据项是在 **ACCESS MODE** 子句中由数据名 1 引用的数据项。

- (8) 如果有关的文卷连接符是外部文卷连接符,则与文卷连接符有关的运行单位中的每一个文卷控制款必须规定相同的存取方式。另外,数据名 1 必须引用一个外部数据项目在每一种情况下与每个有关的文卷控制款中的 **RELATIVE KEY** 短语必须引用相同的外部数据项。

### 2.5 ORGANIZATION IS RELATIVE 子句

#### 2.5.1 功能

**ORGANIZATION IS RELATIVE** 子句规定文卷的逻辑结构是相对组织。

#### 2.5.2 一般格式

**[ORGANIZATION IS]RELATIVE**

#### 2.5.3 一般规则

- (1) **ORGANIZATION IS RELATIVE** 子句指明文卷的逻辑结构是相对组织。文卷组织在文卷创建时建立并且以后不能改变。

- (2) 相对组织是一种永久性的逻辑文卷结构,其每个记录由一个大于零的整数值唯一地标识,这个整数值规定文卷中记录的逻辑次序位置。

### 2.6 I-O-CONTROL 段

#### 2.6.1 功能

**I-O-CONTROL** 段指出建立重运行的点和不同文卷所共享的存储区域。在标准 **COBOL** 的这一版本中视 **I-O-CONTROL** 段中的 **RERUN** 子句是过时成分,因为在标准 **COBOL** 的以后的修改版中要把它删掉。

#### 2.6.2 一般格式

**I-O-CONTROL**

$$\left[ \left[ \text{RERUN ON} \left\{ \begin{array}{l} \text{文卷名 1} \\ \text{实现名 1} \end{array} \right\} \text{EVERY} \left\{ \begin{array}{l} \text{整数 1} \text{ RECORDS OF 文卷名 2} \\ \text{整数 2} \text{ CLOCK-UNITS} \\ \text{条件名 1} \end{array} \right\} \right] \dots \right]$$

[SAME  AREA FOR 文卷名 3

{文卷名 4}...{文卷名 4}]

### 2.6.3 一般规则

(1) 相对 I-O 模块的 RERUN 子句是顺序 I-O 模块的 RERUN 子句的子集。因此, RERUN 子句的规定见顺序 I-O 模块的 2.12 RERUN 子句。

(2) 相对 I-O 模块的 SAME 子句同顺序 I-O 模块的 SAME 子句是一样的。因此, 有关 SAME 子句的规定见顺序 I-O 的 2.13 SAME 子句。

## 3 相对 I-O 模块的数据部

### 3.1 文卷节

文卷节的有关信息见顺序 I-O 模块的 3.1 文卷节。

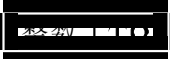
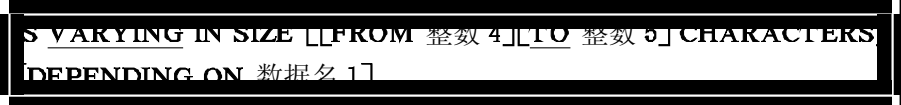
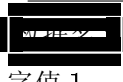
### 3.2 文卷描述款

#### 3.2.1 功能

文卷描述款提供关于一个相对文卷的物理结构、标识和记录名的信息。

#### 3.2.2 一般格式

FD 文卷名 1

[BLOCK CONTAINS  整数 2 {RECORDS  
CHARACTERS} ][RECORD]  
(CONTAINS 整数 3 CHARACTERS  
  
CONTAINS 整数 6 TO 整数 7 CHARACTERS  
[LABEL {RECORD IS } {STANDARD  
RECORDS ARE } {OMITTED } ]  
[VALUE OF { 实现名 1 IS  } ... ]  
[DATA {RECORD IS } {RECORDS ARE } { 数据名 3 } ... ]

#### 3.2.3 语法规则

(1) 层指示符 FD 标识文卷描述款的开始, 而且它必须位于文卷名 1 的前面。

(2) 接在文卷名 1 之后的各个子句出现的次序是任意的。

(3) 在文卷描述款的后面必须接有一个或多个记录描述款。

#### 3.2.4 一般规则

(1) 文卷描述款使文卷名 1 与一个文卷连接符联系起来。

(2) 相对 I-O 模块的 BLOCK CONTAINS 子句与顺序 I-O 模块的 BLOCK CONTAINS 子句相同。因此有关 BLOCK CONTAINS 子句见顺序 I-O 模块。

(3) 相对 I-O 模块的 DATA RECORDS 子句与顺序 I-O 模块的 DATA REOCRDS 子句相同。因此有关 DATA RECORDS 子句的规定见顺序 I-O 模块。在标准 COBOL 的这一版本中视 DATA RECORDS 子句是过时成分, 因为在标准 COBOL 的以后的修改版中要把它删掉。

(4) 相对 I-O 模块的 LABEL RECORDS 子句与顺序 I-O 模块的 LABEL RECORDS 子句相同。因此, 有关 LABEL RECORDS 子句的规定见顺序 I-O 模块。在标准 COBOL 的这一版本中视 LABEL RECORDS 子句是过时成分, 因为在标准 COBOL 的以后的修改版中要把它删掉。



- (5) 相对 I-O 模块的 **RECORD** 子句与顺序 I-O 模块的 **RECORD** 子句是相同的。因此,有关 **RECORD** 子句的规定见顺序 I-O 模块。
- (6) 相对 I-O 模块的 **VALUE OF** 子句与顺序 I-O 模块的 **VALUE OF** 子句是相同的。因此有关 **VALUE OF** 子句的规定见顺序 I-O 模块。在标准 **COBOL** 的这一版本中视 **VALUE OF** 子句是过时成分。因为在标准 **COBOL** 的以后的修改版中要把它删掉。

4 相对 I-O 模块的过程部

4.1 一般描述

当 **COBOL** 源程序中出现相对 I-O 模块的 **USE** 语句时,过程部包含申述过程。以下指出的是当 **USE** 语句出现时过程部的一般格式。

**PROCEDURE DIVISION.**

**DECLARATIVES.**

{节名 **SECTION.**

**USE** 语句.

[段名.[句子]...]}...

**END DECLARATIVES.**

{节名 **SECTION.**

[段名.

[句子]...]}...

4.2 **CLOSE** 语句

4.2.1 功能

**CLOSE** 语句终止 [ ] 文卷的处理。

4.2.2 一般格式

**CLOSE**{ 文卷名 1 [ ] ...

4.2.3 语法规则

(1) **CLOSE** 语句中引用的文卷无须全都具有相同的组织或存取方式。

4.2.4 一般规则

(1) 只能对处于打开方式的文卷执行 **CLOSE** 语句。

(2) 相对文卷属于非顺序的单或多-卷或单位的文卷类型。对于这类文卷,每个 **CLOSE** 语句执行的结果可综述成表 1。

表 1

<b>CLOSE</b> 语句格式	文卷类型:非顺序的单或多一卷或单位
<b>CLOSE</b>	A
<b>CLOSE WITH LOCK</b>	A,B

下面给出表中符号的定义,这是根据文卷是输入文卷、输出文卷还是输入-输出文卷,而分别给出相应的定义,否则一个定义可适用于输入,输出和输入-输出文卷。

A. 关闭文卷

输入文卷和输入-输出文卷(顺序存取方式):

若文卷定位于它的末端用指出该文卷有标号记录,则根据实现者的标准标号约定处理标号。当指明标号记录但又未出现或当未指明标号记录而又出现时,则 **CLOSE** 语句的行为是无定义的。系统执行由

实现者指定的关闭操作。若文卷定位在它的末端,且它指明文卷标号记录,则不进行标号处理,而执行由实现者指明的其它关闭操作。若文卷未定位在它的末端,则执行由实现者指明的关闭操作,但是不进行结束标号处理。

输入文卷和输入-输出文卷(随机存取方式);

输出文卷(随机或顺序存取方式);

若指明文卷有标号记录,则根据实现者的标准标号约定处理标号。当指明有标号记录但未出现或未指明标号记录但又出现时,CLOSE 语句的行为是无定义的。系统执行由实现者指明的关闭操作。若未指明文卷标号记录,则不进行标号处理,但执行由实现者指明的其它关闭操作。

#### B. 文卷锁

给文卷加锁,以保证在运行单位执行期间不能再次打开这个文卷。

(3) 执行 CLOSE 语句引起与文卷名 1 有关的 I-O 状态值的更新(见 1.3.4I-O 状态)。

(5) CLOSE 语句成功执行以后,和文卷名 1 相关联的记录区不再可用。CLOSE 语句执行不成功,则使得记录区的可用性未定义。

(6) 跟着执行成功的 CLOSE 语句,文卷离开打开方式,且文卷不再与文卷连接符有关。

(7) 如果在一个 CLOSE 语句中指定的文卷名 1 多于一个,则执行这种 CLOSE 语句的结果,就如同对 CLOSE 语句中的每个文卷名 1 以同样的次序分开写出的一串 CLOSE 语句一样。

### 4.3 DELETE 语句

#### 4.3.1 功能

DELETE 语句从一个海量存储文卷中逻辑地抹去一个记录。

#### 4.3.2 一般格式

DELETE 文卷名 1 RECORD

[INVALID KEY 命令语句 1]

[NOT INVALID KEY 命令语句 2]

[END-DELETE]

#### 4.3.3 语法规则

(1) 对于引用顺序存取方式的文卷的 DELETE 语句,不能指明 INVALID KEY 短语和 NOT INVALID KEY 短语。

(2) 对于引用非顺序存取方式的且未指明可用 USE AFTER STANDARD EXCEPTION 过程的文卷的 DELETE 语句,则必须指明 INVALID KEY 短语。

#### 4.3.4 一般规则

(1) 在执行该语句时,文卷名 1 引用的文卷必须是以 I-O 方式打开的海量存储文卷(见 4.4OPEN 语句)。

(2) 对顺序存取方式的文卷而言,在 DELETE 语句执行之前,对该文卷名 1 执行的最末一个输入-输出语句必须是已经成功执行的 READ 语句。MSCS 逻辑地从该文卷中删去由 READ 语句存取的记录。

(3) 对随机存取方式存取方式的相对文卷而言,MSCS 从该文卷中逻辑地删去与文卷名 1 相关联的 RELATIVE KEY 数据项的内容所标识的那个记录。若文卷中不含有由该键所指明的记录,就产生一个 INVALID KEY 条件(见 1.3.5 无效键条件)。

(4) 成功地执行过 DELETE 语句之后,指定的记录已经逻辑地从文卷中删去,且不再能被存取。

(5) DELETE 语句的执行不影响与文卷名 1 相关联的记录区的内容,也不影响与文卷名 1 有关的由在 RECORD 子句的 DEPENDING ON 短语中规定的名称引用的数据项的内容。

(6) 当前文卷位置指示符不受 **DELETE** 语句执行的影响。

(7) **DELETE** 语句的执行使得与文卷名 1 相关联的 **I-O** 状态数据项(如果有的话)的值被更新(见 1.3.4 **I-O** 状态)。

(8) 在成功或不成功执行 **DELETE** 操作以后,转移控制根据 **DELETE** 语句中任选的 **INVALID KEY** 和 **NOT INVALID KEY** 短语是否出现来决定(见 1.3.5 无效键条件)。

(9) **END-DELETE** 短语限定了 **DELETE** 语句的作用域(见预备知识 6.6.4.3 语句的作用域)。

#### 4.4 OPEN 语句

##### 4.4.1 功能

**OPEN** 语句初启文卷的处理。

##### 4.4.2 一般格式

```
OPEN { INPUT {文卷名 1}...  
      OUTPUT {文卷名 2}...  
      I-O {文卷名 3}...  
      EXTEND {文卷名 4}...  
      ... }
```

##### 4.4.3 语法规则

```
1. OPEN 语句由下列部分组成:  
2. 文卷名 1, 文卷名 2, 文卷名 3, 文卷名 4, ...
```

(2) 所有在 **OPEN** 语句中引用的文卷无须全都具有相同的组织或存取方式。

##### 4.4.4 一般规则

(1) **OPEN** 语句的成功执行确定文卷的可用性并使得文卷处于打开方式。成功地执行 **OPEN** 语句通过文卷连接符把文卷与文卷名联系起来。

如果一个文卷物理上存在且由输入输出控制系统识别,则此文卷是可用的。表 2 显示了打开可用与不可用文卷的结果。

表 2 一个文卷的可用性

	文卷可用	文卷不可用
输入	正常打开	打开不成功
输入(任选文卷)	正常打开	正常打开;第一次读产生末端条件或无效键条件
<b>I-O</b>	正常打开	打开不成功
<b>I-O</b> (任选文卷)	正常打开	打开使文卷产生
输入	正常打开; 文卷不含记录	打开使文卷产生
<b>EXTEND</b>	正常打开	打开不成功
<b>EXTEND</b> (任选文卷)	正常打开	打开使产生文卷

(2) **OPEN** 语句的成功执行使得相关联的记录区对程序是可用的。如果与文卷名有关的文卷连接符是一个外部文卷连接符,对运行单位只有一个与文卷连接符相关的记录区。

(3) 当一个文卷不处于打开方式时,任何显式或隐式地引用该文卷的语句都不能执行,除非带有 **USING** 或 **GIVING** 短语的 **MERGE** 语句、**OPEN** 语句或带有 **USING** 或 **GIVING** 短语的 **SORT** 语句。

(4) 在任何可允许的输入-输出语句执行之前,**OPEN** 语句必须首先成功地执行。在表 3 中,交叉点上的“X”指出按所在行给出存取方式使用的语句可以在列的顶部给出的文卷打开方式下使用。

表 3 允许的语句

文卷存取方式	语句	打开方式			
		输入	输出	输入-输出	扩展
顺序	READ	×		×	
	WRITE		×		×
	REWRITE			×	
	START	×		×	
	DELETE			×	
随机	READ	×		×	
	WRITE		×	×	
	REWRITE			×	
	START				
	DELETE			×	
动态	READ	×		×	
	WRITE		×	×	
	REWRITE			×	
	START	×		×	
	DELETE			×	

(5) 一个文卷可以在同一个运行单位中用 **INPUT**、**OUTPUT** 和 **I-O** 短语来打开。在对文卷初次执行 **OPEN** 语句后,对同一个文卷每当要执行下一个 **OPEN** 语句,必须对该文卷先执行一个 **CLOSE** 语句。

(6) **OPEN** 语句的执行并不获取或者释放第一个数据记录。

(7) 若对文卷指明有标号记录,则文卷开始的标号处理如下:

a. 当指出 **INPUT** 短语时,**OPEN** 语句的执行将引起根据实现者规定的输入标号核对的约定核对标号。

b. 当指出 **OUTPUT** 短语时,**OPEN** 语句的执行将引起根据实现者规定的输入标号的书写约定来写标号。

当指出有标号记录而又未出现或未指出标号记录却又出现了标号时,**OPEN** 的行为是未定义的。

(8) 如果在执行 **OPEN** 语句期间,发生文卷属性冲突条件,则 **OPEN** 语句执行不成功(见 1.3.7 文卷属性冲突条件)。

(9) 如果用 **INPUT** 短语打开的文卷是一个不存在的往远文卷,则 **OPEN** 语句设置文卷位置指示符为 1 以指明任选输入文卷不存在。

(10) 对于用 **INPUT** 或 **I-O** 短语打开的文卷,**OPEN** 语句把文卷位置指示符置为 1。

(11) 当规定 **EXTEND** 短语时, **OPEN** 语句将该文卷定位于紧接在那个文卷最后一个逻辑记录之后。相对文卷的最后一个逻辑记录是现存的具有最大相对记录号的记录。

(12) 当规定了 **EXTEND** 短语并且 **LABEL RECORDS** 子句指出标号记录存在时, **OPEN** 语句的执行包括下列步骤:

- a. 仅在单卷或单单位文卷的情况下处理开始的文卷标号。
- b. 处理最后一个存在的卷或单位上的开始的卷或单位标号, 如同文卷是用 **INPUT** 短语打开一样。
- c. 处理现存的结束文卷标号, 如同用 **INPUT** 短语打开文卷一样。然后删除这些标号。
- d. 然后进行处理如同文卷已用 **OUTPUT** 短语打开一样。

(13) 带有 **I-O** 短语的 **OPEN** 语句必须引用一个支持输入输出操作的文卷, 这些操作应对以 **I-O** 方式打开的相对文卷是允许的。带有 **I-O** 短语的 **OPEN** 语句执行后, 将引用的文卷置于允许输入和输出操作的打开方式之下。

(14) 当指出 **I-O** 短语, 且 **LABEL RECORDS** 子句指出有标号记录, 则 **OPEN** 语句的执行包含下列各步:

- a. 根据实现者规定的输入输出标号核对的约定核对标号。
- b. 根据实现者规定的输入输出标号的书写约定写新标号。

(15) 对一个不可用的任选文卷, 带有 **I-O** 短语的 **OPEN** 语句的成功执行建立了该文卷。这种建立了发生如同依序执行下列语句:

**OPEN OUTPUT** 文卷名。

**CLOSE** 文卷名。

上述语句执行后跟着执行源程序中指明的 **OPEN** 语句。

带有 **OUTPUT** 短语的 **OPEN** 语句成功执行之后, 便建立一个文卷。但此时该文卷中还不含数据记录。

(16) 执行 **OPEN** 语句引起与文卷名相关的 **I-O** 状态值的更新(见 1.3.4 **I-O** 状态)。

(17) 如果在一个 **OPEN** 语句中指定的文卷名多于一个, 则执行此 **OPEN** 语句的结果就如同对 **OPEN** 语句中每一个文卷名以同样顺序分开写出的一串 **OPEN** 语句一样。

(18) 一个文卷的最大和最小记录长度在文卷创建时建立且以后不能改变。

## 4.5 READ 语句

### 4.5.1 功能

对顺序存取方式而言, **READ** 语句使文卷中的下一逻辑记录成为可用。对随机存取方式而言, **READ** 语句使得海量存储文卷中的一个指定记录成为可用。

### 4.5.2 一般格式

格式 1:

**READ** 文卷名 1 **RECORD** [**INTO** 标识符 1] [**AT END** 命令语句 1]  
[**NOT AT END** 命令语句 2] [**END-READ**]

格式 2:

**READ** 文卷名 1 **RECORD** [**INTO** 标识符 1] [**INVALID KEY** 命令语句 3]  
[**NOT INVALID KEY** 命令语句 4] [**END-READ**]

### 4.5.3 语法规则

(1) 与标识符 1 相关联的存储区和与文卷名 1 相关联的记录区不能是同一个存储区。

(2) 格式 1 的 **READ** 语句须用于所有顺序存取方式的文卷。

(4) 当以随机方式检索记录时,对处于随机存取方式 **INDEXED** 的文卷,使用格式 2 的 **READ** 语句。

(5) 若对文卷名 1 未指出可应用的 **USE AFTER STANDARD EXCEPTION** 过程,则必须指出 **INVALID KEY** 短语或 **AT END** 短语。

#### 4.5.4 一般规则

(1) 在 **READ** 语句执行时,由文卷名 1 引用的文卷必须已经用 **INPUT** 或 **I-O** 方式打开(见 4.4**OPEN** 语句)。

(3) **READ** 语句的执行,使得和文卷名 1 相关联的 **FILE STATUS** 数据项的值被更新(见 1.3.4**I-O** 状态)。

(4) 在开始执行格式 1 的 **READ** 语句时对文卷位置指示符的赋值,根据下列规则用它来确定成为有用的记录。在相对文卷中记录的比较与相对键号有关。

- a. 如果文卷位置指示符指明没有建立下一个有效记录,则 **READ** 语句的执行不成功。
- b. 如果文卷位置指示符指明 **INDEXED** 则依照一般规则 10 来执行下去。
- c. 如果文卷位置指示符由先前的 **OPEN** 或 **START** 语句建立,则选取文卷中存在的其相对记录号大于或等于文卷位置指示符值的第一个记录。
- d. 如果由先前的 **READ** 语句建立文卷位置指示符,则选取文卷中存在的其相对记录号大于所选的文卷位置指示符值的第一个记录。

如果找到一个记录满足上述规则,则它在与文卷名 1 有关的记录区中成为可用的,除非为文卷名 1 规定 **RELATIVE KEY** 短语且所选记录相对记录号的有效数字位数大于相对键数据项的长度,在这种情况下,文卷位置指示符用来指明这个条件且按一般规则 10 规定的过程执行。

如果没有找到满足上述规则的记录,文卷位置指示符用来指明下一个逻辑记录不存在且按一般规则 10 规定的过程执行。

如果一个记录成为可用的,文卷位置指示符用来指向该可用记录的相对记录号。

(5) 如果不考虑处理时间和存取时间重叠的方式,则 **READ** 语句的概念是不变的。即任何下一个 **READ** 语句执行之前或命令语句 2,命令语句 4 执行之前,(如果已指明的话)记录对目标程序是可用的。如果既无命令语句 2 也无命令语句 4,则在 **READ** 语句之后的任一语句执行以前,记录对目标程序也是可用的。

(6) 当文卷的逻辑记录由多个记录描述款来描述时,这些记录自动地共享同一个存储区,这等价于对该区域的隐含的重定义。在 **READ** 语句执行完毕时,处于当前数据记录范围之外的任何数据项的内容是无定义的。

(7) 可以在一个 **READ** 语句中规定 **INTO** 短语:

- a. 如果只有一个记录描述属于文卷描述款,或
- b. 如果与文卷名 1 有关的所有记录名和由标识符 1 引用的数据项描述了一个组项或一个字符型初等项

(8) 执行带有 **INTO** 短语的 **READ** 语句的结果相当于按规定的次序应用下列规则:

- a. 执行不带 **INTO** 短语的相同的 **READ** 语句。
- b. 当前记录从记录区传送到由标识符 1 指定的区域中,这种传送根据不带 **CORRESPONDING** 短语的 **MOVE** 语句的规则进行。当前记录的长度由 **RECORD** 子句规定的规则决定。如果 **READ** 语句执行不成功,则这种隐含的 **MOVE** 语句便不发生。和标识符 1 相关联的任何下标在记录读入以后以及即将传送到数据项之前立即求值。记录在记录区和由标识符 1 引用的数据项中都是可用的。

(9) 在执行格式 2 的 **READ** 语句时,如果文卷位置指示符指出往远输入文卷不存在,则存在无效键条件,且 **READ** 语句的执行是不成功的(见 1.3.5 无效键条件)。

(10) 对于格式 1 的 **READ** 语句,如果文卷位置指示符指出下一个逻辑记录不存在,或相对记录号的有效数字位数大于相对键数据项的长度,则按规定的次序执行下列操作:

a. 把赋给文卷位置指示符的值放入与文卷名 1 有关的 **I-O** 状态以指出末端条件(见 1.3.4I-O 状态)。

b. 如果在引起该条件的语句中规定了 **AT END** 短语,控制转移到 **AT END** 短语中的命令语句 1。任何与文卷名 1 有关的 **USE AFTER STANDARD EXCEPTION** 过程不执行。

c. 如果没有指明 **AT END** 短语,一个 **USE AFTER STANDARD EXCEPTION** 过程必须与文卷名 1 相联系,并执行那个过程。从那个过程返回后就转到 **READ** 语句末端之后的下一个可执行语句。

当末端条件发生时,**READ** 语句的执行是不成功的。

(11) 在执行 **READ** 语句期间,如果既没有末端条件也没有无效键条件发生,则跳过 **AT END** 短语或 **INVALID KEY** 短语(如果已指明的话),且执行下列动作:

a. 设置文卷位置指示符且更新与文卷名 1 有关的 **I-O** 状态。

b. 如果存在一个非末端或无效键条件的例外条件,则在执行对文卷名 1 可用的任何 **USE AFTER EXCEPTION** 过程后,根据 **USE** 语句的规则进行控制转移(见 4.8USE 语句)。

c. 如果不存在例外条件,在记录区中的记录可用,且由于执行 **INTO** 短语而导致隐式传送控制转移到 **READ** 语句的结束处或命令语句 2(如果已指明的话),在后一种情况,根据在命令语句 2 中规定的每个语句的规则继续执行。如果执行一个引起显式控制转移的过程分支或条件语句,则根据那个语句的规则进行控制转移;否则,直到执行完命令语句 2,控制转移到 **READ** 语句的结束处。

(12) 在执行 **READ** 语句不成功的情况下,有关记录区的内容无定义,且对文卷位置指示符赋值以指出未建立有效的下一个记录。

(13) 对一个指定为动态存取方式的相对文卷,带有指明 **NEXT** 短语的格式 1 的 **READ** 语句从文卷中检索出下一个逻辑记录。

(14) 对相对文卷,如果对文卷名 1 规定 **RELATIVE KEY** 短语,则根据 **MOVE** 语句的规则执行格式 1 的 **READ** 语句把可用记录的相对记录号送给相对键数据项(见核心模块 6.19MOVE 语句)。

(15) 对相对文卷,执行格式 2 的 **READ** 语句使文卷位置指示符的值赋定,该值包含在文卷的在 **RELATIVE KEY** 短语所引用的数据项中,且相对记录号与文卷位置指示符一样的记录在与文卷名 1 有关的记录区中成为可用的。若该文卷中不包含这样的记录,就存在一个 **INVALID KEY** 条件,且 **READ** 语句的执行是不成功的(见 1.3.5 无效键条件)。

(16) 如果读入的记录的字付位置数小于由文卷名 1 的记录描述款规定的最小长度,则最后读入的有效字符右边的部分记录区是无定义的。如果读入记录的字符位置数大于由文卷名 1 的记录描述款规定的最大长度,则记录按最大长度从右边截断。无论在那一种情况下,**READ** 语句是成功的,且设置一个 **I-O** 状态以指出发生记录长度冲突(见 1.3.4I-O 状态)。

(17) **END-READ** 短语限定 **READ** 语句的作用域(见预备知识 6.6.4.3 语句的作用域)。

## 4.6 REWRITE 语句

### 4.6.1 功能

**REWRITE** 语句逻辑地替换海量存储文卷中的一个记录。

### 4.6.2 一般格式

**REWRITE** 记录名 1 [**FROM** 标识符 1] [**INVALID KEY** 命令语句 1]

[**NOT INVALID KEY** 命令语句 2] [**END-REWRITE**]

### 4.6.3 语法规则

- (1) 记录名 1 和标识符 1 一定不能指称同一个存储区。
- (2) 记录名 1 是数据部文卷节中的一个逻辑记录的名,它可以受限。
- (3) 对于引用顺序存取方式的相对文卷,**REWRITE** 语句不能指明 **INVALID KEY** 短语和 **NOT INVALID KEY** 短语。

(4) 对于随机存取方式的相对文卷,且未指出适当的 **USE AFTER STANDAD EXCEPTION** 过程时,则在 **REWRITE** 语句中必须指出 **INVALID KEY** 短语。

#### 4.6.4 一般规则

(1) 在执行该语句时,与记录名 1 相关联的文卷必须是海量存储文卷,且以 **I-O** 方式打开(见 4.4 **OPEN** 语句)。

(2) 对顺序存取方式的文卷,在 **REWRITE** 语句执行之前,相关联的文卷的最末一个执行的输入-输出语句必须是一个成功执行的 **READ** 语句。**MSCS** 逻辑地替换由该 **READ** 语句存取的记录。

(3) 在 1 级中,由记录名 1 引用的记录中的字符位置数必须等于替换的记录的字符位置数。

(4) 成功执行 **REWRITE** 语句所释放的逻辑记录在记录区中不再可用,除非相应的文卷在 **SAME RECORD AREA** 子句中指明外。该逻辑记录对程序仍然是可用的,但它不作为与相关输出文卷同一 **SAME RECORD AREA** 子句中出现的其他文卷的一个记录。此外,它对于和记录名 1 相关联的文卷也是可用的。

(5) 带有 **FROM** 短语的 **REWRITE** 语句的执行等价于按顺序执行下列语句:

a. 根据 **MOVE** 语句规定的规则执行语句。

**MOVE** 标识符 1 **TO** 记录名 1

b. 执行不带 **FROM** 短语的相同的 **REWRITE** 语句。

(6) 在执行完 **REWRITE** 语句之后,由标识符 1 引用的区域中的信息是可用的,即使由记录名 1 引用的区域中的信息是不可用的。

(7) **REWRITE** 语句的执行不影响文卷位置指示符。

(8) **REWRITE** 语句的执行引起更新和记录名 1 有关的文卷的 **I-O** 状态的值(见 1.3.4 **I-O** 状态)。

(9) 执行 **REWRITE** 语句则释放一个逻辑记录给操作系统。

(10) 成功或不成功执行 **REWRITE** 语句操作之后的控制转移取决于在 **REWRITE** 语句中是否出现任选的 **INVALID KEY** 和 **NOT INVALID KEY** 短语(见 1.3.5 无效键条件)。

(11) **END-REWRITE** 短语限定 **REWRITE** 语句的作用域(见 GB/T 4092.1 中 6.6.4.3 语句的作用域)。

(12) 由记录名 1 引用的记录中字符位置数必须不大于最大字符位置数也不能小于最小字符位置数,字符位置数是同记录名 1 对应的文卷名的相关 **RECORD IS VARYING** 子句所允许的范围。无论在哪一种情况下,执行 **REWRITE** 语句是不成功的,不发生更新操作,记录区的内容不受影响且与记录名 1 有关的文卷的 **I-O** 状态被置一个值以指明条件的原因(见 1.3.4 **I-O** 状态)。

(13) 对于以随机存取方式存取的文卷,**MSCS** 逻辑上替换由该文卷相关联的 **RELATIVE KEY** 数据项的内容所指出的记录。若该文卷不包含由该键指出的记录,则存在一个 **INVALID KEY** 条件。当识别出该无效键条件时,**REWRITE** 语句的执行是不成功的,此时不进行更新操作,也不影响记录区中的数据。且与记录名 1 有关的文卷名的 **I-O** 状态被置一个值以指出条件的原因(见 1.3.4 **I-O** 状态)。



## 4.7 START 语句

### 4.7.1 功能

为了顺序地检索后继记录, **START** 语句为相对文卷内的逻辑定位提供一个基点。

### 4.7.2 一般格式

<b>START</b> 文卷名 1	[ <b>KEY</b> ]	<b>IS EQUAL TO</b>	] 数据名 1
		<b>IS =</b>	
		<b>IS GREATER THAN</b>	
		<b>IS &gt;</b>	
		<b>IS NOT LESS THAN</b>	
		<b>IS NOT &lt;</b>	
		<b>IS GREATER THAN OR EQUAL TO</b>	
		<b>IS &gt;=</b>	

[**INVALID KEY** 命令语句 1]

[**NOT INVALID KEY** 命令语句 2]

[**END-START**]

### 4.7.3 语法规则

- (1) 文卷名 1 必须是顺序存取方式或动态存取方式的文卷的名。
- (2) 数据名 1 可以受限。
- (3) 若对文卷名 1 未指出可应用的 **USE AFTER STANDARD EXCEPTION** 过程, 则必须指出 **INVALID KEY** 短语。

(4) 若有数据名 1, 则它必须是相关文卷控制款的 **ACCESS MODE** 子句中的 **RELATIVE KEY** 短语中所指出的数据项。

### 4.7.4 一般规则

(1) 在 **START** 语句执行时, 以文卷名 1 命名的文卷必须以输入或 **I-O** 方式打开(见 4.4 **OPEN** 语句)。

(2) 若未指出 **KEY** 短语, 则隐含指出关系运算符 '**IS EQUAL TO**'。

(3) **START** 语句的执行既不更改记录区的内容也不更改与文卷名 1 有关的 **RECORD** 子句的 **DEPENDING ON** 短语规定的的数据名所引用的数据项内容。

(4) **KEY** 短语中的关系运算符指出的比较种类在由文卷名 1 引用的文卷的相关联记录中的键和按一般规则 10 中指出的数据项之间进行。应用数值比较规则(见核心模块 6.3.1.1.1 数值运算对象比较)。

a. 文卷位置指示符定位于文卷中其键满足比较关系的第一个逻辑记录的相对记录号。

b. 若文卷中的任何记录都不满足这个比较关系, 则存在 **INVALID KEY** 条件且 **START** 语句的执行是不成功的。

(5) **START** 语句的执行引起更新相关联的文卷名 1 的 **I-O** 状态数据项(如果有的话)的值(见 1.3.4 **I-O** 状态)。

(6) 在执行 **START** 语句时, 如果文卷位置指示符指出任选输入文卷不存在, 那么存在无效键条件而且 **START** 语句的执行是不成功的。

(7) 成功或不成功执行 **START** 语句之后的控制转移是根据在 **START** 语句中是否出现任选的 **INVALID KEY** 短语和 **NOT INVALID KEY** 短语来决定的(见 1.3.5 无效键条件)。

(8) 执行不成功的 **START** 语句之后, 置文卷位置指示符以指出没有建立下一个有效的记录。

(9) **END-START** 短语限定 **START** 语句的作用域(见预备知识 6.6.4.3 语句的作用域)。

(10) 一般规则 4 描述的比较使用与文卷名 1 相关的 **ACCESS MODE** 子句的 **RELATIVE KEY**

### 4.8.1 功能

### 4.8.2 一般格式

USE AFTER STANDARD { EXCEPTION }  
ERROR

{ 文卷名 1 }  
INPUT  
PROCEDURE ON { OUTPUT }  
 I-O  
 { EXTENT }

(1) 出现 **USE** 语句时,它必须紧接在过程部的申述节的节首之后,并且 **USE** 语句出现其中的句子只包含一个语句。该节的其余部分必须由 0 个、1 个或多个用于定义要使用的过程的过程段组成。

(3) 在 **USE** 语句中文卷名 1 的出现不能同时要求执行一个以上的 **USE** 过程。

(5) 在 **USE** 语句中隐式地或显式地引用的文卷无须全都具有相同的组织或存取方式。

(6) INPUT、OUTPUT、I/O [REDACTED] 短语中的任何一个在给定的过程部的申述部分只能指明一次。

(1) 一个 **COBOL** 源程序中可以包括申述过程,不管此程序包含另一个程序或被包含在另一个程序中。程序执行过程中,如果申述前的 **USE** 语句中描述的条件发生,则调用该申述。在一分别编译的程序执行时,如果申述前的 **USE** 语句中描述的条件发生,则该分别编译的程序中只有一个申述被调用。这里的分别编译的程序包含引起限定条件的语句。如果分别编译的程序中不存在限定申述,则不执行申述。

(3) 与 **USE** 语句有关的过程名可以在不同的申述节中引用,或在仅有一个 **PERFORM** 语句的非申述过程中引用。

(5) 输入输出操作执行不成功时执行标准输入输出例外例行程序, 执行完毕再由输入输出控制系统执行与 **USE** 语句相关的过程。除非 **AT END** 或 **INVALID KEY** 短语处于优先地位, 何时执行这些过程的规则如下:

- b.** 如果指明了 **INPUT**, 对以输入方式打开或正在以输入方式打开的文卷当有关 **USE** 语句中描述的条件发生时执行有关过程, 对由指定同一条件的另一个 **USE** 语句中的文卷名 1 引用的那些文卷除外。

c. 如果指明了 **OUTPUT**, 对以输出方式打开或正以输出方式打开的文卷, 当有关 **USE** 语句中描述的条件发生时执行有关的过程, 对由指定同一条件的另一个 **USE** 语句中的文卷名 1 引用的那些文卷除外。

- d.** 如果指明了 I-O, 对以 I-O 方式打开或正以 I-O 方式打开的文卷, 当有关 **USE** 语句中描述的条

件发生时执行有关的过程,对由指定同一条件的另一个 **USE** 语句中的文卷名 1 引用的那些文卷除外。

e. 如果指明 **EXTEND**,对以 **EXTEND** 方式打开或正以 **EXTEND** 方式打开的文卷,当有关 **USE** 语句中描述的条件发生时执行有关的过程,对由指定同一条件的另一个 **USE** 语句中的文卷名 1 引用的那些文卷除外。

(6) 在执行了 **USE** 过程之后,控制转移到输入输出控制系统中的调用例行程序。如果 **I-O** 状态值没有指出关键性的输入输出错误,输入输出控制系统控制返回到引起执行例外情况的输入输出语句之后的下一可执行语句。如果 **I-O** 状态确实指出了关键性错误,则由实现者来决定采取什么动作(见 1.3.4 **I-O** 状态)。

(7) 在 **USE** 过程中,一定不能执行这样的语句,该语句引起执行一个先前被调用而至今尚未返回调用例行程序的 **USE** 过程。

#### 4.9 **WRITE** 语句

##### 4.9.1 功能

**WRITE** 语句把一个逻辑记录释放到输出文卷或输入-输出文卷上去。

##### 4.9.2 一般格式

**WRITE** 记录名 1 [**FROM** 标识符 1] [**INVALID KEY** 命令语句 1]  
[**NOT INVALID KEY** 命令语句 2]  
[**END-WRITE**]

##### 4.9.3 语法规则

- (1) 记录名 1 和标识符 1 不能引用同一存储区。
- (2) 记录名 1 是数据部文卷节中的逻辑记录的名字,它可以受限。
- (3) 若未对相关的文卷指出可应用的 **USE** 过程,则必须指出 **INVALID KEY** 短语。

##### 4.9.4 一般规则

(1) 在该语句执行时相关的文卷必须已按 **OUTPUT**、**I-O** 或 **INVALID** 方式打开(见 4.4 **OPEN** 语句)。

(2) 执行 **WRITE** 语句所释放的逻辑记录在记录区中不再可用。但与记录名 1 有关的文卷名在 **SAME RECORD AREA** 子句中指明时为例外。该逻辑记录对该程序还是可用的,不过这时它是作为与相关输出文卷同一个 **SAME RECORD AREA** 子句中出现的其它文卷的一个记录。此外,它对和记录名 1 相关的文卷也是可用的。

(3) 带有 **FROM** 短语的 **WRITE** 语句的执行结果等价于:

a. 根据对 **MOVE** 语句所指出的规则执行

**MOVE** 标识符 1 **TO** 记录名 1

b. 其后接着执行同一个不带 **FROM** 短语的 **WRITE** 语句。

(4) **WRITE** 语句执行完成后,尽管记录名 1 引用的区域中的信息可能是不可用的,而由标识符 1 引用的区域中的信息却是可用的。

(5) **WRITE** 语句的执行不影响文卷位置指示符。

(6) **WRITE** 语句的执行引起更新和记录名 1 有关的文卷的 **I-O** 状态的值(见 1.3.4 **I-O** 状态)。

(7) **WRITE** 语句的执行释放一个逻辑记录给操作系统。

(8) 由记录名 1 引用的记录中的字符位置数必须不大于由同记录名 1 有关的文卷名的有关 **RECORD IS VARYING** 子句来指明的最大字符位置数也不小于最小字符位置数,无论哪一种情况 **WRITE** 语句的执行是不成功的且不发生 **WRITE** 操作,记录区的内容不受影响且与记录名 1 有关的文卷的 **I-O** 状态被置一个值以指出引起此条件的原由(见 1.3.4 **I-O** 状态)。

(9) 在执行带有 **NOT INVALID KEY** 短语的 **WRITE** 语句期间,如果无效键条件不发生,在如下适当的时候控制转移到命令语句 2:

a. 如果 **WRITE** 语句的执行是成功的则在写过记录并更新了与记录名 1 有关的文卷名的 **I-O** 状态之后。

b. 如果执行 **WRITE** 语句不成功且其原因不是无效键条件,则在更新了与记录名 1 有关的文卷名的 **I-O** 状态且在执行了可应用到有关记录名 1 的文卷名上由 **USE AFTER STANDARD EXCEPTION PROCEDURE** 语句指定的过程(如果有的话)之后。

(10) **END-WRITE** 短语限定 **WRITE** 语句的作用域(见 GB/T 4092.1 中 6.6.4.3 语句的作用域)。

(11) 当一个相对文卷按输出方式打开时,记录可以按下列情形之一存入该文卷。

a. 若存取方式是顺序的,则 **WRITE** 语句释放一个记录到海量存储控制系统。第一个记录具有相对记录号 1,而后继释放的记录具有记录号 2、3、4、…。若为与文卷名 1 相关的文卷名规定了 **RELATIVE KEY** 短语,则刚刚发放的记录的相对记录号在 **WRITE** 语句的执行期间由 **MSCS** 存入相对键数据项中,且根据 **MOVE** 语句的规则进行(见核心模块 6.19 **MOVE** 语句)。

b. 若存取方式是随机的,则在执行 **WRITE** 语句之前,相对键数据项的值必须在程序中使用相对记录号加以初始化,使其与记录域中的记录联系起来。然后执行 **WRITE** 语句把该记录释放给 **MSCS**。

(12) 当一个相对文卷以扩展方式打开时,记录被插入该文卷。释放到海量存储控制系统的第一个记录有一个相对记录号,它大于现存于此文卷中的最大相对记录号。释放到海量存储控制系统中的后续记录相继具有更高的相对记录号。如果对与记录名 1 相关的文卷名规定 **RELATIVE KEY** 短语,则在执行 **WRITE** 语句期间根据 **MOVE** 语句的规则,由海量存储控制系统把释放记录的相对记录号送入相对键数据项(见 GB/T 4092.2 中 6.19 **MOVE** 语句)。

(13) 当一个文卷以 **I-O** 方式打开,且存取方式是随机的,则把记录插入到相关的文卷中去。相对键数据项的值必须在程序中使用相对记录号初始化,使其与记录域中的记录联系起来。然后执行 **WRITE** 语句把该记录发送给 **MSCS**。

(14) 在下列情形, **INVALID KEY** 条件成立。

- a. 若存取方式是随机的,并且相对键数据项指出一个在该文卷中已存在的记录;或者
- b. 企图写一个记录到该文卷的外部定义的边界之外;或者
- c. 当相对记录号中有效数字的位数大于文卷中描述的相对键数据项的长度时。

(15) 当识别出 **INVALID KEY** 条件,则 **WRITE** 语句的执行是不成功的。但记录区的内容不受影响。对与记录名 1 相关联的文卷名的 **I-O** 状态置一个值,以表示该条件的发生原因。程序的执行根据无效键条件的规则进行(见 1.3.4 **I-O** 状态和 1.3.5 无效键条件)。

#### 附加说明:

本标准由中华人民共和国机械电子工业部提出。

本标准由南京大学负责起草。

本标准主要起草人钱树人、王静英、冯惠、段祥。

本标准由 1983 年 12 月首次发布,1992 年 8 月第一次修订。