



# 中华人民共和国国家标准

GB/T 4092.6—1992

---

## 程序设计语言 COBOL 索引 I-O 模块

Programming language COBOL  
Indexed 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  
索引 I-O 模块

GB/T 4092.6—1992

Programming language COBOL  
Indexed I-O module

代替 GB 4092.6—83

## 1 引言

### 1.1 功能

索引 I-O 模块提供以随机方式或顺序方式存取海量存储文卷中记录的功能。索引文卷中的每个记录是由记录里一个或多个键的值来唯一标识的。

### 1.2 级别特征

1 级索引 I-O 对文卷控制款、文卷描述款及 I-O CONTROL 段中的各段提供局部功能。在过程部中,1 级索引 I-O 对 CLOSE、OPEN、READ、REWRITE、USE 和 WRITE 语句提供局部功能,而对 DELETE 语句提供了完整功能。

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

### 1.3 语言概述

#### 1.3.1 组织

用索引方法组织的文卷是海量存储文卷,通过给出指定的记录键值可以存取文卷中任一记录。文卷的记录中定义的每个键数据项均与一个索引相联系。每个这种索引代表每个记录中相应键数据项的一组值。因此,每个索引是能够提供存取文卷中任一记录的机构。

每个索引文卷有一个主索引,它代表文卷中每个记录的主记录键。每个记录被插入到文卷中、或被改变、或从文卷中被删除完全依赖于主记录键的值。文卷中每个记录的主记录键必须是唯一的,而且在修改记录时不能被改变。主记录键在该文卷的文卷控制款的 RECORD KEY 子句中说明。

次记录键提供检索文卷记录的次路径。这种键在文卷控制款的 ALTERNATE RECORD KEY 子句中被命名。每个记录中特定的次记录键的值可以不唯一。当这些值可能不唯一时,必须在 ALTERNATE RECORD KEY 子句中使用 DUPLICATES 短语指明。

#### 1.3.2 存取方式

对索引组织,顺序存取的序是根据该文卷的对比序列的引用键值的递升次序。在文卷处理期间,与文卷相关联的任何键可以用作为引用键。在具有重复键引用值的一组记录里,记录的检索次序是记录被写到该组记录里的次序。START 语句可用在索引文卷里为一系列后继顺序检索建立开始点。

当用随机方式存取文卷时,输入-输出语句以程序员指定的次序存取记录。对于索引组织,程序员通

过把记录键的某一个值放入记录键或次记录键数据项来规定所要的记录。

对动态存取方式,程序员使用输入输出语句的适当格式,可以任意地由顺序存取方式改变成随机存取方式。

### 1.3.3 文卷位置指示符

文卷位置指示符是在这个标准中为简化在某个输入输出操作期间,从给定的文卷中存取下一个记录的确切说明而使用的一个概念实体。置文卷位置指示符的值仅受 **CLOSE**、**OPEN**、**READ** 和 **WRITE** 语句的影响。文卷位置指示符的概念对用输出方式打开的文卷没有意义。

### 1.3.4 I-O 状态

**I-O** 状态是一个两字符概念实体,它的值标志在 **CLOSE**、**DELETE**、**OPEN**、**READ**、**REWRITE**、**WRITE** 或 **WRITE** 语句执行期间输入输出操作的状态,并且标志与那个输入输出语句相关的任一命令语句执行之前或任一可应用的 **USE AFTER STANDARD EXCEPTION** 过程的执行以前输入输出操作的状态。通过在相应文卷的文卷控制款中使用 **FILE STATUS** 子句,使得 **I-O** 状态值对 **COBOL** 程序成为可用。

**I-O** 状态也用于决定一个可用的 **USE AFTER STANDARD EXCEPTION** 过程是否要执行。如果出现的条件不是标题为“成功的结束”之下包含的那些条件,则这样的过程可以依据别处指出的规则执行。若出现的条件是在标题为“成功的结束”之下包含的那些条件,则不执行这样的过程(见 4.8 **USE** 语句。)

某些 **I-O** 状态值指出关键错误条件。它们是: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。若对应输入输出操作的 **I-O** 状态值指出这样一个错误条件,则由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 **USE AFTER STANDARD EXCEPTION** 过程时,则实现者确定输入输出控制系统错误标准处理以后采取什么动作。

**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。输入输出语句执行成功。
- (2) 到末端。由于末端条件引起顺序 **READ** 语句执行不成功。
- (3) 无效键。由于无效键条件引起输入输出语句执行不成功。
- (4) 永久性错误。由于一个错误引起输入输出语句执行不成功,这个错误使文卷的处理不能继续进行。执行指定的例外过程,除非引用实现者定义的技术来改正永久性错误条件,否则永久性错误条件对该文卷的所有以后的输入输出操作具有影响。
- (5) 逻辑错误。由于在文卷上执行了不正确的输入输出操作顺序,或由于与用户定义的限制相冲突而引起的输入输出语句执行不成功。
- (6) 实现者定义。由于实现者指定的条件引起输入输出语句执行不成功。

下面给出值表,列出对应于在索引文卷上执行了输入输出操作且出现前面命名的条件时放入 **I-O** 状态中的值。若多于一个值可应用,那么由实现者确定将哪个可应用的值放到 **I-O** 状态中。

#### (1) 成功的结束

- a. **I-O** 状态=00。输入输出语句成功地执行,对有关输入输出操作没有进一步的信息可用。
- b. **I-O** 状态=02。输入输出语句成功地执行,但重复键被查出。

#### 1) 对 **READ** 语句,当前引用键的键值等于当前引用键里下个记录中相同键的值。

2) 对 **REWRITE** 或 **WRITE** 语句,刚写的记录至少有一个允许重复的次记录键建立了一个重复键值。

- c. **I-O** 状态=04。**READ** 语句成功地执行,但正被处理的记录的长度与相应文卷的定长文卷属性不一致。

d. I-O 状态=05。OPEN 语句成功地执行,但被引用的往还文卷在 OPEN 语句执行时并不存在如打开方式是 I-O 或扩充方式,则该文卷已被建立

(2) 不成功结束的末端条件

a. I-O 状态=10。试图执行顺序的 READ 语句,但由于下列原因文卷中不存在下一个逻辑记录:

1) 已经达到文卷的末端。

(3) 不成功结束的无效键条件

a. I-O 状态=21。对顺序存取的索引文卷存在一个顺序错误。对于那个文卷在成功执行 READ 语句和执行下一个 REWRITE 语句之间程序改变了主记录键的值,或违反了相继记录键的值是递升顺序这个要求(见 4.9WRITE 语句)。

b. I-O 状态=22。企图写或重写索引文卷的记录将导致出现重复主记录键,或没有 DUPLICATE 短语时出现重复次记录键。

c. I-O 状态=23。由于下述情况而出现这个条件:

1) 企图随机存取文卷中不存在的记录。

d. I-O 状态=24。试图写到索引文卷外部确定的边界之外。确定这些边界的方式由实现者指明。

(4) 不成功结束的永久性错误条件

a. I-O 状态=30。出现永久性错误并且没有和输入输出有关的进一步可用信息。

b. I-O 状态=35。因试图对尚不存在的文卷执行具有 INPUT、I-O 短语的 OPEN 语句,而引起永久性错误发生。

c. I-O 状态=37。因试图对一个并不支撑该 OPEN 语句中指定的打开方式的文卷执行 OPEN 语句,引起永久性错误发生。可能的违反是:

1) 说明了 OUTPUT 短语,但文卷不支持写操作。

2) 说明了 I-O 短语,但文卷不支持对索引文卷在以 I-O 方式打开时所允许的输入和输出操作。

3) 说明了 INPUT 短语,但文卷不支持读操作。

d. I-O 状态=38。由于试图对前面已经带锁关闭的文卷执行 OPEN 语句,而引起永久性错误的发生。

e. I-O 状态=39。由于在定长文卷属性与程序对那个文卷指明的属性之间检测到矛盾,OPEN 语句执行不成功。

(5) 不成功结束的逻辑错误条件

a. I-O 状态=41。试图对已处在打开方式的文卷执行 OPEN 语句。

b. I-O 状态=42。试图对不处于打开方式的文卷执行 CLOSE 语句。

c. I-O 状态=43。在顺序存取方式中,在 DELETE 或 REWRITE 语句执行之前,对文卷执行的最后一个输入输出语句不是一个成功地执行的 READ 语句。

d. I-O 状态=44。由于下述因素引起边界违反:

1) 试图与或里与比相关文卷名的 RECORD IS VARYING 子句所允许的最长记录长或比最短记录短的记录

2) 在 1 级中,试图对一个索引文卷写或重写一个与被替代的记录长度不同的记录。

e. I-O 状态=46。试图对以输入或 I-O 方式打开的文卷执行顺序 READ 语句,而且由于下述原因未建立有效的下个记录:

2) 先前的 **READ** 语句执行不成功,但未引起末端条件。

3) 先前的 **READ** 语句引起了末端条件。

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

g. I-O 状态=48。试图对不是以 I-O、输出方式打开的文卷执行 **WRITE** 语句。

h. I-O 状态=49。试图对不是以 I-O 方式打开的文卷执行 **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。存在由实现者定义的条件。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

**DELETE**、**READ**、**REWRITE** 或 **WRITE** 语句执行后,可能出现无效键条件。当无效键条件出现时,识别出该条件的输入输出语句的执行是不成功的,而且文卷不受影响(见 4.3 **DELETE** 语句;4.5 **READ** 语句;4.6 **REWRITE** 语句和 4.9 **WRITE** 语句)。

在输入输出语句中指定的输入输出操作执行以后,若出现无效键条件,则发生下列次序的动作:

(1) 与该语句相关联的文卷联接符的 I-O 状态被置为标志无效键条件的值(见 1.3.4 I-O 状态)。

(2) 若输入输出语句中指明了 **INVALID KEY** 短语,则不执行与该文卷联接符相关联的 **USE AFTER EXCEPTION** 过程,而控制转向 **INVALID KEY** 短语中说明的命令语句。然后按照那个命令语句中指定的每个语句的规则继续执行。若执行显式引起控制转移的过程分支或条件语句,则按照那个语句的相应规则转移;否则,在 **INVALID KEY** 短语中说明的命令语句执行完成后,控制转向输入输出语句的结束处,并且忽略 **NOT INVALID KEY** 短语(若已指定的话)。

(3) 若在输入输出语句中未指明 **INVALID KEY** 短语,则 **USE AFTER EXCEPTION** 过程必须与该文卷联接符相关联,而且执行该过程,然后按照 **USE** 语句的规则转移控制。忽略 **NOT INVALID KEY** 短语(若已指定的话)(见 4.8 **USE** 语句)。

由输入-输出语句指定的输入-输出操作执行以后,若无效键条件不出现,那么 **INVALID KEY** 短语被忽略(若已指定的话)。与该语句相关联的文卷联接符的 I-O 状态被修改,而且执行下列动作:

(1) 若出现非无效键条件的例外条件,则在与该文卷联接符相关联的任何 **USE AFTER EXCEPTION** 过程执行以后,按照 **USE** 语句的规则转移控制(见 4.8 **USE** 语句)。

(2) 若无例外条件出现,则控制转向输入输出语句结束处,或若指明 **NOT INVALID KEY** 短语时就转向由其说明的命令语句。在后一种情况,按照那个命令语句中说明的相应每个语句的规则继续执行。若执行引起显式控制转移的过程分支或条件语句,则按照那个语句相应规则转移;否则,在 **NOT INVALID KEY** 短语中说明的命令语句执行完后,控制转到输入输出语句结束处。

### 1.3.6 末端条件

执行 **READ** 语句后,末端条件可能发生(见 4.5 **READ** 语句)。

### 1.3.7 文卷属性冲突条件

执行 **OPEN**、**REWRITE** 或 **WRITE** 语句,可能产生文卷属性冲突条件。当文卷属性冲突条件出现时,识别出该条件的输入输出语句的执行是不成功的,而且文卷不受影响(见 4.4 **OPEN** 语句;4.6 **REWRITE** 语句和 4.9 **WRITE** 语句)。

当识别出文卷属性冲突条件时,发生下列次序的动作:

(1) 在与该文卷名相关联的 I-O 状态中放一个指示文卷属性冲突条件的值(见 1.3.4 I-O 状态)。

(2) 执行与该文卷名相关联的 **USE AFTER EXCEPTION** 过程(若有的话)。

## 2 索引 I-O 模块的环境部

### 2.1 输入输出节

有关输入输出节的信息见顺序 I-O 模块的 2.1 中的说明。

### 2.2 FILE-CONTROL 段

有关 **FILE-CONTROL** 段的信息见顺序 I-O 模块的 2.2 中的说明。

## 2.3 文卷控制款

### 2.3.1 功能

文卷控制款说明索引文卷的有关物理属性。

### 2.3.2 一般格式

**SELECT** [实现名 1] 文卷名 1

**ASSIGN TO** { 实现名 1  
字值 1 } ...

**RESERVE** 整数 1 **AREA**  
**AREAS**

[**ORGANIZATION IS**] **INDEXED** [**ACCESS MODE IS** { **SEQUENTIAL**  
**RANDOM**  
**DYNAMIC** } ]

**RECORD KEY IS** 数据名 1

[**FILE STATUS IS** 数据名 3]。

### 2.3.3 语法规则

(1) 在文卷控制款中必须首先指出 **SELECT** 子句。跟在 **SELECT** 子句后的那些子句可以按任何次序出现。

(2) 在数据部中描述的每个文卷必须作为 **FILE-CONTROL** 段中的文卷名被指明一次,且只能一次。在 **SELECT** 子句中指出的每个文卷名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,而且不能是象征常量。关于实现名 1 所允许的内容以及字值 1 的值,其含义和规则由实现者定义。

### 2.3.4 一般规则

(1) 若由文卷名 1 引用的文卷连接符是一个外部文卷连接符(见 GB 4092.12 中 4.5 **EXTERNAL** 子句),则运行单位中引用这文卷连接符的所有文卷控制款必须:

b. 对 **ASSIGN** 子句中实现名 1 或字值 1 有一致性的说明。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. 相关记录中具有同一相对位置的数据名 1 有相同数据描述款。

g. 数据名 2 有相同数据描述款,数据名 2 在相关的记录中具有同一相对位置,相同数量的次记录键和相同的 **DUPLICATES** 短语。

(2) 外部媒体上的数据项使用本源字符集。

(3) 对索引文卷,假定对比序列与本源字符集和关联。这是用于顺序处理文卷而引用的给定键值的序列。

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

(5) **ASSIGN** 子句指出由文卷名 1 所引用的文卷与由实现名 1 或字值 1 所引用的存储媒体之间的

联系。

(6) 索引 I-O 模块的 **RESERVE** 子句与顺序 I-O 模块的 **RESERVE** 子句相同。因此有关 **RESERVE** 子句的说明见顺序 I-O 模块的 2.9 **RESERVE** 子句。

(7) 索引 I-O 模块的 **FILE STATUS** 子句与顺序 I-O 模块的 **FILE STATUS** 子句相同。因此有关 **FILE STATUS** 子句的说明见顺序 I-O 模块 2.5 **FILE STATUS** 子句。与索引文卷的 **FILE STATUS** 子句相关联的数据项的内容见 1.3.4 I-O 状态。

(8) **ACCESS MODE** 子句 **SEQUENTIAL ORGANIZATION IS INDEXED** 子句和 **RECORD KEY** 子句见下文。

## 2.4 ACCESS MODE 子句

### 2.4.1 功能

**ACCESS MODE** 子句指出在文卷中存取记录的次序。

### 2.4.2 一般格式

**ACCESS MODE IS** { **SEQUENTIAL**  
**RANDOM**  
**DYNAMIC** }

### 2.4.3 语法规则

(1) 对 **SORT** 或 **MERGE** 语句的 **USING** 或 **GIVING** 短语中说明的文卷名不能指定 **ACCESS MODE IS RANDOM** 子句。

### 2.4.4 一般规则

(1) 如果未指出 **ACCESS MODE** 子句,则假定是顺序存取。

(2) 若存取方式是顺序的,文卷中的记录按文卷组织决定的顺序存取。对索引文卷,这个顺序是根据文卷的对比序列在给定引用键内的记录键值的递升次序。

(3) 如果存取方式是随机的,对索引文卷,记录键数据项的值表明要存取的记录。

(4) 若存取方式是随机的,则文卷中记录键值的顺序必须和存取顺序一致。

(5) 若相关联的文卷连接符是外部文卷连接符,则与那个文卷连接符相关联的运行单位中每个文卷控制款必须指定相同的存取方式。

## 2.5 ALTERNATE RECORD KEY 子句

### 2.5.1 功能

**ALTERNATE RECORD KEY** 子句指明一个次记录键,这个次记录键提供了在索引文卷中记录的次存取路径。

### 2.5.2 一般格式

**ALTERNATE RECORD KEY IS** 数据名 1 [**WITH DUPLICATES**]

### 2.5.3 语法规则

(1) 数据名 1 可以受限。

(2) 数据名 1 必须在与文卷名相关联的记录描述款中定义成字符类型的数据项,**ALTERNATE RECORD KEY** 子句从属于这个文卷名。

(3) 数据名 1 不能引用包含可变出现数据项的组项。

(4) 数据名 1 不能引用这样的—数据项,它的最左边字符位置和该文卷相关联的主记录键或任一其他次记录键的最左边的字符位置相对应。

(5) 若索引文卷包含变长记录,则每个次记录键必须包含在记录开始的 **x** 个字符位置中,**x** 等于对该文卷指定的最小记录长度(见 GB/T 4092.4 中 3.8 **RECORD** 子句)。

**2.5.4 一般规则**

- (1) **ALTERNATE RECORD KEY** 子句指出与这个子句相关联的文卷的次记录键。
- (2) 数据名 1 的数据描述以及其在记录里的相对位置必须与创建该文卷时所用的一样。该文卷次记录键数目也必须与创建该文卷时所用的一样。
- (3) **DUPLICATES** 短语指出相关联的次记录键的值可以在文卷中任何记录里重复。如果没有指出 **DUPLICATES** 短语,则相关联的次记录键的值必不能在文卷中任何记录中重复。
- (4) 如果文卷有多于一个记录描述款,则数据名 1 仅需在这些记录描述款之一中描述。在任一记录描述款中由数据名 1 引用的相同的字符位置隐含地被引用为那个文卷的所有其他记录描述款相对应的键。
- (5) 若相关联的文卷连接符是外部文卷连接符,那么在与那文卷连接符相关联的运行单位里,每个文卷控制款必须指出对数据名 1 有相同的数据描述款,相关记录里有同样的相对位置,同样数目的次记录键,和同样的 **DUPLICATES** 短语。

**2.6 ORGANIZATION IS INDEXED 子句****2.6.1 功能**

**ORGANIZATION IS INDEXED** 子句指明索引组织为文卷的逻辑结构。

**2.6.2 一般格式**

**[ORGANIZATION IS]INDEXED**

**2.6.3 一般规则**

- (1) **ORGANIZATION IS INDEXED** 子句指明索引组织为文卷的逻辑结构。文卷组织在文卷创建时建立,而且以后不能改变。
- (2) 索引组织是一种永久的逻辑文卷结构,用记录中一个或多个键值来标识该文卷中每个记录。

**2.7 RECORD KEY 子句****2.7.1 功能**

**RECORD KEY** 子句指出主记录键,该主记录键提供了在索引文卷中记录的存取路径。

**2.7.2 一般格式**

**RECORD KEY IS** 数据名 1

**2.7.3 语法规则**

- (1) 数据名 1 可以受限。
- (2) 数据名 1 必须引用与该文卷名相关联的记录描述款中的字符型数据项,**RECORD KEY** 子句从属于该文卷名。
- (3) 数据名 1 不能引用包含可变出现数据项的组项。

(4) 若索引文卷包含变长记录,则主记录键必须被包含在记录开始的  $X$  个字符位置里, $X$  等于  $X$  文卷指定的最小记录长度(见 GB/T 4092.4 中 3.8 RECORD 子句)。

**2.7.4 一般规则**

- (1) **RECORD KEY** 子句指出与该子句相关联的文卷的主记录键。主记录键的值在该文卷的诸记录中必须是唯一的。
- (2) 数据名 1 的数据描述和它在记录中的相对位置一样,必须与建立文卷时使用的相同。
- (3) 若文卷有多于一个记录描述款,数据名 1 仅需在这些记录描述款之一中描述。在任一记录描述款中由数据名 1 引用的相同的字符位置隐含地被引用为那个文卷的所有其他记录描述款对应的键。
- (4) 若相关联的文卷连接符是外部文卷连接符,那么在与那个文卷连接符相关联的运行单位里,所有文卷描述款必须对在相关记录里具有同样相对位置的数据名 1 指出相同的数据描述款。

**2.8 I-O CONTROL 段**



### 2.8.1 功能

**I-O CONTROL** 段指出了建立的重运行点和不同文卷共享的存储区。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。

### 2.8.2 一般格式

**I-O-CONTROL.**

[**RERUN ON** { 文卷名 1 } **EVERY**  
{ 整数 1 **RECORDS OF** 文卷名 2 }  
{ 整数 2 **CLOCK-UNITS** }  
条件名 1 ] ...

[**SAME** **AREA FOR** 文卷名 3 { 文卷名 4 } ... ] ... . ]

### 2.8.3 一般规则

(1) 有关索引 **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** 子句的说明见顺序 **I-O** 模块的 2.12 **RERUN** 子句。

(2) 有关索引 **I-O** 模块的 **SAME** 子句与顺序 **I-O** 模块的 **SAME** 子句相同。因此有关 **SAME** 子句的说明见顺序 **I-O** 模块的 2.13 **SAME** 子句。

## 3 索引 **I-O** 模块的数据部

### 3.1 文卷节

有关文卷节的信息见顺序 **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索引文卷的物理结构、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** } ]  
[**CONTAINS** 整数 3 **CHARACTERS**  
**RECORDS VARYING IN SIZE** [[**FROM** 整数 4 ] [**TO** 整数 5 ]  
**CHARACTERS** ] [**DEPENDING ON** 数据名 1 ]  
[**CONTAINS** 整数 6 **TO** 整数 7 **CHARACTERS** ]  
[**LABEL** { **RECORD IS** } { **STANDARD** }  
{ **RECORDS ARE** } { **OMITTED** } ]  
[**VALUE OF** { 实现名 1 **IS** } { 字值 1 } ] ...  
[**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,它必须位于文卷名 1 的前面。

(2) 跟在文卷名 1 后面的那些子句可以以任意次序出现。

(3) 文卷描述款之后必须跟有一个或多个记录描述款。

#### 3.2.4 一般规则

(1) 文卷描述款把文卷名 1 与文卷连接符联系起来。

(2) 索引 I-O 模块的 **BLOCK CONTAINS** 子句与顺序 I-O 模块的 **BLOCK CONTAINS** 子句相同。因此有关于 **BLOCK CONTAINS** 子句的说明见顺序 I-O 模块的 3.3 **BLOCKCONTAINS** 子句。

(3) 索引 I-O 模块的 **DATA RECORDS** 子句与顺序 I-O 模块的 **DATA RECORDS** 子句相同。因此,有关 **DATA RECORDS** 子句的说明见顺序 I-O 模块 3.5 **DATA RECORDS** 子句。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) 索引 I-O 模块的 **LABEL RECOREDS** 子句与顺序 I-O 模块的 **LABEL RECORDS** 子句相同。因此有关 **LABEL RECORDS** 子句的说明见顺序 I-O 模块中 3.6 **LABEL RECORDS** 子句。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) 索引 I-O 模块的 **RECORD** 子句与顺序 I-O 模块的 **RECORD** 子句相同。因此有关 **RECORD** 子句的说明见顺序 I-O 模块中 3.8 **RECORD** 子句。

(6) 索引 I-O 模块的 **VALUE OF** 子句与顺序 I-O 模块的 **VALUE OF** 子句相同。因此有关 **VALUE OF** 子句的说明见顺序 I-O 模块中 3.9 **VALUE OF** 子句。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 索引 I-O 模块的过程部

### 4.1 一般描述

当在 COBOL 源程序中存在索引 I-O 模块的 **USE** 语句时,过程部包含中述过程。下面给出当 **USE** 语句存在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** 语句中引用的文卷不需要都有相同的组织或存取方式。

#### 4.2.4 一般规则

(1) **CLOSE** 语句只能对处于打开方式的文卷执行。

(2) 索引文卷属于非顺序的单或多-卷或单位的文卷类型,对于这类文卷的各种 **CLOSE** 语句的执行结果概括成下表 1。

表 1

CLOSE 语句 格式	文卷类型：非顺序的单或多-卷或单位
CLOSE	A
CLOSE WITH LOCK	A,B

下面给出表中符号的定义。此处,根据文卷是输入,输出或输入输出文卷,而分别给出了相应的定义;否则,一种定义可适用于输入,输出以及输入输出。

A. 关闭文卷

输入文卷和输入输出文卷(顺序存取方式)

如果文卷定位在它的末端并且对该文卷指出有标号记录,则按照实现者的标准标号约定处理标号。当指出有标号记录但又未出现,或当未指出标号记录而又出现时,则 **CLOSE** 语句的行为是无定义的,但执行由实现者指定的关闭操作。如果文卷定位在末端而对该文卷没有指明标号记录,则不进行标号处理,但执行由实现者指出的其他关闭操作。如果文卷不定位在它的末端,则执行由实现者指出的关闭操作,但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(随机存取或顺序存取方式);

如果对文卷指出了标号记录,则按照实现者的标准标号约定处理这些标号。当指出了标号记录而未出现,或未指出标号记录而又出现时,**CLOSE** 语句的行为是无定义的,但执行由实现者指出的关闭操作。如果对该文卷未指出标号记录,则不进行标号处理,但执行由实现者指出的其他关闭操作。

B. 文卷锁

该文卷被锁而且在这个运行单位的这次执行期间不能再打开

(3) 执行 **CLOSE** 语句将改变与文卷名 1 相关联的 I-O 状态值(见 1.3.4I-O 状态)。

(5) 在 **CLOSE** 语句执行成功之后,与文卷名 1 相关联的记录区就不再可用。如 **CLOSE** 语句执行不成功,则记录区的可用性是未定义的。

(6) 在 **CLOSE** 语句执行成功之后,从打开方式中除去这个文卷,而且该文卷不再与该文卷连接符相关联。

(7) 若在 **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** 语句中每个文卷名 1 以同样的顺序分开写出的一串 **CLOSE** 语句一样。

4.3 DELETE 语句

4.3.1 功能

**DELETE** 语句从海量存储文卷中逻辑地除去一个记录。

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文卷的 **DELETE** 语句,不能指出 **INVALID KEY** 和 **NOT INVALID KEY** 短语。

(2) 对于引用非顺序存取方式且未指出可应用 **USE AFTERSTANDARD 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) 在执行这个语句时,由文卷名 1 引用的文卷必须是海量存储文卷且必须已用 **I-O** 方式打开(见 4.4 **OPEN** 语句)。

(2) 对顺序存取方式的文卷而言,在 **DELETE** 语句执行之前,对文卷名 1 最近执行的输入输出语句必须是成功执行的 **READ** 语句。海量存储控制系统(MSCS)逻辑地从该文卷中除去由 **READ** 语句存取的记录。

(3) 对随机存取方式的文卷而言,MSCS 从该文卷中逻辑地除去由与文卷名 1 相关联的主记录键数据项的内容标识的那个记录。如果该文卷不包含由该键指出的记录,则就产生一个 **INVALID KEY** 条件(见 1.3.5 无效键条件)。

(4) 成功地执行 **DELETE** 语句之后,指定的记录已被逻辑地从文卷中删去,并且不能再被存取。

(5) **DELETE** 语句的执行,不影响记录区的内容 或与文卷名 1 相关联的 **RECORD** 子句内 **DEPENDING ON** 短语中规定的数据名所引用数据项的内容

(6) 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** 状态值被更新(见 1.3.4 **I-O** 状态)。

(8) 在 **DELETE** 操作的成功或不成功的执行之后,控制转移依赖于 **DELETE** 语句中是否存在 **INVALID KEY** 和 **NOT INVALID KEY** 短语(见 1.3.5 无效键条件)。

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

#### 4.4 **OPEN** 语句

##### 4.4.1 功能

**OPEN** 语句对文卷的处理进行初始化。

##### 4.4.2 一般格式

```

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

```

##### 4.4.3 语法规则

(2) 所有在 **OPEN** 语句中引用的文卷不要求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,使文卷处在打开方式,并且通过文卷连接符把该文卷与文卷名联系起来。

若文卷物理地存在且被输入输出控制系统识别,则该文卷是可用的。表 2 给出打开可用文卷和不可用文卷的结果。

## 卷 2 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
I-O	正常打开	打开不成功
I-O(任选文卷)	正常打开	打开使文卷产生
输出	正常打开,文卷不包含记录	打开使文卷产生
EXTEND	正常打开	打开不成功
EXTEND(任选文卷)	正常打开	打开使文卷产生

(2) **OPEN** 语句的成功执行,使得相关记录区可用于程序。若与文卷名相关联的文卷连接符是外部文卷连接符,则该运行单位仅有一个记录区与这文卷连接符相关。

(3) 当文卷未打开时,不能显式或隐式地执行引用该文卷的语句,带有 **USING** 或 **GIVING** 短语的 **MERGE** 语句、**OPEN** 语句,或带有 **USING** 或 **GIVING** 短语的 **SORT** 语句除外。

(4) 在执行任何允许的输入输出语句之前,必须首先成功地执行 **OPEN** 语句。在表 3 中,在交叉点上×指出可允许的语句。这些语句按所在行左边指出的存取方式使用,且可以和索引文卷组织以及在列的顶部给出的打开方式一起使用。

表 3 可允许语句

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

(5) 一个文卷可以在同一个运行单位中用 **INPUT**、**OUTPUT** 和 **I-O** 短语打开。一个文

卷的 **OPEN** 语句初次执行之后,对该文卷再执行 **OPEN** 语句之前,必须对该文卷先执行一个不带 **LOCK** 短语的 **CLOSE** 语句。

(6) **OPEN** 语句的执行并不获得或释放第一个数据记录。

(7) 如果对文卷指出有标号记录,则文卷开始的标号处理如下:

- a. 当指出 **INPUT** 短语时,**OPEN** 语句的执行将根据实现者规定的输入标号核对的约定核对标号。
- b. 当执行 **OUTPUT** 短语时,**OPEN** 语句的执行将根据实现者规定的输出标号的约定写标号。

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** 语句的行为是未定义的。

(8) 在 **OPEN** 语句的执行期间,若出现文卷属性冲突条件,那么 **OPEN** 语句的执行是不成功的(见 1.3.7 文卷属性冲突条件)。

(9) 若带有 **INPUT** 短语打开的文卷是向不存在的任选文卷,那么 **OPEN** 语句用文卷位置指示符指明该任选输入文卷尚不存在。

(10) 对于带有 **INPUT** 或 **I-O** 短语打开的文卷,把文卷位置指示符置为具有与文卷相关的对比序列中最低起始位置的字符,而且把主记录键建立为引用键。

(11) 指出 **EXTEND** 短语时,**OPEN** 语句定位文卷于该文卷的最后逻辑记录的紧后面。对索引文卷,最后逻辑记录是当前存在的具有最高主键值的记录。

(12) 在指出 **EXTEND** 短语且 **LABEL RECORD** 子句指出有标号记录时,**OPEN** 语句的执行含有如下几步:

- a. 在单卷或单单位文卷情况,仅处理开始文卷标号。
- b. 处理最后存在的卷或单位上的开始卷或单位标号,如同该文卷已用 **INPUT** 短语打开了一样。
- c. 处理现存的末尾文卷标号,如同文卷正在用 **INPUT** 短语打开一样。然后这些标号被清除。
- d. 接下来的处理过程如同文卷已经用 **OUTPUT** 短语打开那样。

(13) 带有 **I-O** 短语的 **OPEN** 语句必须引用支持输入和输出两种操作的文卷,这些操作对索引文卷以 **I-O** 方式打开时是允许的。执行具有 **I-O** 短语的 **OPEN** 语句使被引用的文卷对输入和输出操作均处于打开方式。

(14) 在指出 **I-O** 短语且 **LABEL RECORD** 子句指出有标号记录时,执行 **OPEN** 语句包括如下几步:

- 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. 根据实现者规定的写输入输出标号的约定写新标号。

(15) 对于非可用的任选文卷,成功地执行带有 **I-O** 短语的 **OPEN** 语句就建立了该文卷。这种建立采取的动作如同按序执行下列语句:

**OPEN OUTPUT** 文卷名。

**CLOSE** 文卷名。

这些语句之后执行源程序里指出的 **OPEN** 语句。

成功地执行带有 **OUTPUT** 短语的 **OPEN** 语句建立文卷。在文卷建立以后,该文卷还不包含有数据记录。

(16) **OPEN** 语句的执行使与该文卷名相关的 **I-O** 状态值被更新(见 1.3.4 **I-O** 状态)。

(17) 若在一个 **OPEN** 语句中指定的文卷名多于一个,则执行这种 **OPEN** 语句的结果就如同对 **OPEN** 语句中的每个文卷名以同样的顺序分开写出的一串 **OPEN** 语句一样。

(18) 文卷的最小和最大记录长度在文卷创建时建立,而且以后不能改变。

## 4.5 READ 语句

### 4.5.1 功能

对于顺序存取方式, **READ** 语句使文卷中的下一个逻辑记录成为可用。对随机存取方式, **READ** 语句使海量存储文卷中的一个指定的记录成为可用。

### 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 相关的存储区和与文卷名 1 相关的记录区不能是同一存储区。

(2) 数据名 1 必须是被指明为与文卷名 1 相关的记录键数据项的名。

(3) 数据名 1 可以受限。

(4) 对于所有顺序存取方式的文卷必须使用格式 1。

(6) 对随机存取方式的文卷, 在随机地检索记录时, 使用格式 2。

(7) 如果未对文卷名 1 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) 在执行这个语句时, 由文卷名 1 引用的文卷必须已用输入或 **I-O** 方式打开(见 4.4 **OPEN** 语句)。

(3) **READ** 语句的执行引起与文卷名 1 相关的 **I-O** 状态值被更新(见 1.3.4 **I-O** 状态)。

(4) 在格式 1 **READ** 语句开始执行时, 利用设置文卷位置指示符使记录成为可用是按如下规则确定。索引文卷中关于记录的比较涉及当前引用键的值。对索引文卷, 按文卷的对比序列进行比较。

a. 若文卷位置指示符指出还没有建立好有效的下一个记录, 则 **READ** 语句的执行是不成功的。

c. 若文卷位置指示符由前面的 **OPEN** 语句建立, 则选择文卷中记录键值大于或等于文卷位置指示符的第一个现存记录。

d. 若文卷位置指示符由前面的 **READ** 语句建立, 则选择文卷中记录键值大于文卷位置指示符的第一个现存记录。

e. 若文卷位置指示符由前面的 **READ** 语句建立, 并且当前引用的键允许重复, 则或者选择文卷中记录键值等于文卷位置指示符的第一个现存记录且该记录在这组重复键记录中的逻辑位置是紧跟在前面 **READ** 语句使其可用的那个记录之后; 或者选择文卷中记录键值大于文卷位置指示符的第一个现存记录。

若找到满足上述规则的记录,则使它在与文卷名 1 相关的记录区中成为可用的。

若未找到满足上述规则的记录,则用文卷位置指示符指示下个逻辑记录不存在,而且象一般规则 10 中指出的那样进行处理。

若某记录成为可用的,则置文卷位置指示符为该记录的当前引用键的值。

(5) 无论存取时间和处理时间使用何种重叠方法,READ 语句的概念不变;若指出命令语句 2 或命令语句 4,则在其执行之前,记录对目标程序是可用的,或者若既没有指出命令语句 2 又没有指出命令语句 4,则在 READ 语句之后的任何语句执行之前,记录对目标程序是可用的。

(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域,这等价于对该区域的隐含重定义。在 READ 语句执行完毕时,对于当前数据记录范围之外的任何数据项的内容是不确定的。

(7) 在 READ 语句中可以指出 INTO 短语:

a. 若仅当一个记录描述属于这个文卷描述款时,或  
b. 若与文卷名 1 相关的全部记录名以及由标识符 1 引用的数据项描述一个组项或字符型初等项时。

(8) 执行带有 INTO 短语的 READ 语句,其结果等价于按序应用下面规则:

a. 执行不带 INTO 短语的同一 READ 语句。  
b. 把当前记录从记录区传送到标识符 1 指出的区域中,这种传送按照无 CORRESPONDING 短语的 MOVE 语句的规则进行。当前记录的长度由 RECORD 子句指出的规则确定。  
若 READ 语句执行不成功,这种隐含的 MOVE 语句就不发生。与标识符 1 相关的任何下标在记录被读入之后以及即将传送到数据项之前计算。该记录在记录区以及标识符 1 引用的数据项里两处均为可用。

(9) 在格式 2 READ 语句执行时,若文卷位置指示符指出任意输入文卷不存在,则无效键条件产生并且 READ 语句的执行是不成功的(见 1.3.5 无效键条件)。

(10) 对格式 1 READ 语句,若文卷位置指示符指出下个逻辑记录不存在,或任选输入文卷不存在,则跟着执行下列动作:

a. 从文卷位置指示符所置的值中导出一个值放到与文卷名 1 相关的 I-O 状态中以指出末端条件(见 1.3.4 I-O 状态)。  
b. 若引起该条件的语句中指明了 AT END 短语,则控制转到 AT END 短语中的命令语句 1。任何与文卷名 1 相关的 USE AFTER STANDARD EXCEPTION 过程均不执行。  
c. 若未指明 AT END 短语,则必须有与文卷名 1 相关的 USE AFTER STANDARD EXCEPTION 过程,而且该过程被执行。从该过程返回后,控制转到 READ 语句结束处之后的下一个可执行语句。

当末端条件出现时,READ 语句的执行认为是不成功的。

(11) 若在 READ 语句执行期间,既没有产生末端条件又没有产生无效键条件,则若指明有 AT END 短语或 INVALID KEY 短语就忽略它,并且执行下述动作:

a. 文卷位置指示符被置值,并且与文卷名 1 相关的 I-O 状态被更新。  
b. 若出现一个不是末端条件又不是无效键条件的例外条件,则在执行对文卷名 1 可应用的任何 USE AFTER EXCEPTION 过程之后,按 USE 语句的规则进行控制转移(见 4.8 USE 语句)。  
c. 若无例外条件出现,则记录在记录区中是可用的,而且执行 INTO 短语引起的隐式传送。控制转到 READ 语句的结束处或若指出命令语句 2 就转向它。在后一种情况,按照命令语句 2 中指出的相应各语句的规则继续执行。若执行明显引起控制转移的过程分支或条件语句,则按那个语句的规则进行转移;否则在命令语句 2 执行完后,控制转到 READ 语句结束处。



(12) 若 **READ** 语句执行不成功,则相关联的记录区中的内容是不确定的,对索引文卷而言,引用键是不确定的,并且文卷位置指示符指出没有已建立好的下一个有效记录。

(13) 对于指出为动态存取方式的索引文卷,带有 **NEXT** 短语的格式 1 的 **READ** 语句,从文卷中检索下一个逻辑记录。

(14) 对顺序存取的索引文卷,在引用的次记录键中有相同重复值的记录成为可用,它是依照执行 **WRITE** 语句释放的次序,或者执行建立重复值的 **REWRITE** 语句所释放记录的次序使用。

(15) 对于索引文卷,如果在格式 2 的 **READ** 语句中指出了 **KEY** 短语,则数据名 1 就作为这一检索的引用键而建立。如果指出动态存取方式,引用这个键也用于对该文卷的格式 1 **READ** 语句的后继执行进行检索,直到为该文卷建立不同的引用键时为止。

(16) 对索引文卷,则主记录键作为这一语句的引用键而建立。如果指明动态存取方式,引用的这个键同样用于该文卷的格式 1 的 **READ** 语句的后继执行进行检索,直到为该文卷建立不同的引用键时为止。

(17) 对索引文卷,格式 2 **READ** 语句的执行把文卷位置指示符置为引用键的值。这个值与该文卷存储记录中相应数据项的值进行比较,直到具有相等值的第一个记录被找到为止。在具有重复值的次记录键情况,找到的第一个记录是释放给海量存储控制系统(MSCS)的重复序列中的第一个记录。这样找到的记录在文卷名 1 相关的记录区里是可用的。如果没有可以如此标识的记录,则无效键条件就发生,并且 **READ** 语句的执行是不成功的(见 1.3.5 无效键条件)。

(18) 若被读记录中字符位置数小于相应文卷名 1 记录描述款指出的最小长度,则记录区中所读的最后一个有效字符右边的那部分是未定义的。若被读记录中字符位置数大于相应文卷名 1 记录描述款指出的最大长度,记录被截断于最大长度的右边。这两种情况之任何一种,**READ** 语句是成功的,但 I-O 状态被置为记录长度出现了矛盾(见 1.3.4 I-O 状态)。

(19) **END-READ** 短语限定 **READ** 语句的作用域(见 GB/T 4092.1 中 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存在的记录。

### 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 不能引用同一个存储区。

(2) 记录名 1 是数据部文卷节中逻辑记录名并且可以限定。

(3) 如果对相关文卷名没有指出可应用的 **USE AFTER STANDARD EXCEPTION** 过程,则必须指出 **INVALID KEY** 和 **NOT INVALID KEY** 短语。

### 4.6.4 一般规则

(1) 由记录名 1 相关的文卷名所引用的文卷必须是海量存储文卷,并且在该语句执行时必须已用 I-O 方式打开(见 4.4 **OPEN** 语句)。

(2) 对于用顺序存取方式的文卷,在 **REWRITE** 语句执行之前,对相关文卷的最后一个输入输出语句必须是成功执行的 **READ** 语句。**MSCS** 逻辑地替换 **READ** 语句存取过的记录。

(3) 对 1 级,由记录名 1 引用的记录中的字符数必须等于被替换的记录中字符数。

(4) 在成功地执行 **REWRITE** 语句后,所释放的逻辑记录在记录区中不再可用,除非相应的文卷名在 **SAME RECORD AREA** 子句中指明外。该逻辑记录对程序仍然是可用的,但它是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,它对与记录名 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序执行如下语句。

a. 按 **MOVE** 语句指定的规则执行语句:

**MOVE** 标识符 1 **TO** 记录名 1

b. 无 **FROM** 短语的同一个 **REWRITE** 语句。

(6) **REWRITE** 语句执行完成以后,标识符 1 所引用的区中的信息是可用的。记录名 1 所引用的区中的信息不可用。

(7) **REWRITE** 语句的执行不影响文卷位置指示符。

(8) **REWRITE** 语句的执行引起与记录名 1 相关文卷名的 **I-O** 状态值被更新(见 1.3.4 **I-O** 状态)。

(9) **REWRITE** 语句的执行把逻辑记录释放给操作系统。

(10) **REWRITE** 语句成功或不成功地执行之后,控制转移依赖于 **REWRITE** 语句中是否存在任选的 **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(见预备知识 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不能大于 **RECORD IS VARYING** 子句所允许的最大字符位置数或小于最小字符位置数,该子句与记录名 1 相关的文卷名相关联。不论两种情况的哪一种,如果 **REWRITE** 语句的执行是不成功的,更新的操作不进行,记录区的内容不受影响而与记录名 1 相关的文卷的 **I-O** 状态指出所发生的条件(见 1.3.4 **I-O** 状态)。

(13) 对于顺序存取方式的文卷,被替换的记录由包含在主记录键中的值指出。当执行 **REWRITE** 语句时,包含在被替换之记录的主记录键数据项的值必须等于上次读入的文卷中记录的主记录键的值。

(14) 对随机存取方式的文卷,被替换的记录由主记录键数据项指出。

(15) 对具有次记录键的记录,**REWRITE** 语句的执行如下:

a. 当特定次记录键的值不改变时,在那个键是引用键的时候,检索的次序保持不变。

b. 当特定次记录键的值改变时,在那个键是引用键时,那个记录后继的检索次序可能被改变。当允许重复键值时,记录逻辑地被定位在一组重复记录的最后位置,这组重复记录包含与记录中被替换的次记录键值相同的次记录键值。

(16) 无效键条件在下列几种情况下存在:

a. 文卷的存取方式是顺序的,且被替换之记录的主记录键的值不等于上次读入的记录的主记录键的值,或

b. 文卷是随机的存取方式,并且被替换之记录的主记录键值不等于现存在文卷中的任何记录的主记录键值,或

c. 当不允许重复次记录键时,被替换记录的次记录键值等于已存于文卷中记录的对应数据项值。

(17) 识别出无效键条件时,**REWRITE** 语句的执行是不成功的,不执行更新操作,记录区的内容不受影响而与记录名 1 相关联的文卷名的 **I-O** 状态指出所发生的条件(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** 语句为索引文卷中的逻辑定位提供一个基点。

### 4.7.2 一般格式

**START** 文卷名 1 **KEY IS EQUAL TO**  
**IS EQUAL TO**  
**IS =**  
**IS GREATER THAN**  
**IS >**  
**IS NOT LESS THAN**  
**IS NOT <**  
**IS GREATER THAN OR EQUAL TO**  
**IS >=**

数据名 1

[**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 必须是顺序或动态存取方式的文卷名。

(2) 数据名 1 可以限定。

(3) 如果未对文卷名 1 指出可应用的 **USE AFTER STANDARD EXCEPTION** 过程, 则必须指出 **INVALID KEY** 短语。

(4) 如果指出 **KEY** 短语, 则数据名 1 必须引用:

a. 指定为与文卷名 1 相关的记录键的数据项(见 2.5 **ALTERNATE RECORD KEY** 子句和 2.7 **RECORD KEY** 子句), 或者引用

b. 任何这样的字符型数据项, 其在文卷的记录中最左字符位置与文卷名 1 相关记录键的最左字符位置相对应, 而且其长度不大于那个记录键的长度。

### 4.7.4 一般规则

(1) 在 **START** 语句执行时, 以文卷名 1 引用的文卷必须以 **INPUT** 或 **I-O** 方式打开(见 4.4 中的 **OPEN** 语句)。

(2) 如果未指出 **KEY** 短语, 则隐含着关系运算符“**IS EQUAL TO**”的存在。

(3) **START** 语句的执行不改变记录区的内容, 也不改变数据名引用的数据项的内容, 该数据名在文卷名 1 相关联 **RECORD** 子句的 **DEPENDING ON** 短语中指出。

(4) **KEY** 短语中的关系运算符指出的比较类型, 出现在由文卷名 1 引用的文卷中的相关联记录的键和按一般规则 11 和 12 指出的数据项之间。比较是对引用的升序键按文卷对比序列进行。如果运算对象的大小不等, 则在较长的右边截去超过部分, 使它的长度等于短的长度。所有其他的非数值比较规则均适用(见 GB/T 4092.2 中 6.3.1.1.1 非数值运算对象的比较)。

a. 置文卷位置指示符为其键满足比较的第一个逻辑记录中引用键的值。

b. 如果文卷中所有记录均不满足比较要求, 则无效键条件产生, 而且 **START** 语句的执行是不成功的。

(5) **START** 语句的执行使得与文卷名 1 相关联的 **I-O** 状态值被更新(见 1.3.4 **I-O** 状态)。

(6) 在 **START** 语句执行时, 如果文卷位置指示符指出任选输入文卷不存在, 那么无效键条件出现而且 **START** 语句的执行是不成功的。



(3) 与 **USE** 语句相关的过程名可以在不同的申述节中,或在仅有 **PERFORM** 语句的非申述过程所引用。

(4) 当明显指明文卷名 1 时,没有其他的 **USE** 语句可应用于文卷名 1。

(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,此后再执行与 **USE** 语句相关的过程。除非一个 **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:

a. 若指出文卷名 1,则在 **USE** 语句中描述的条件出现时,执行相关的过程。

b. 若指出 **INPUT**,则在 **USE** 语句中描述的条件出现时,对任何以 **INPUT** 方式打开或正在打开的文卷执行相关的过程,对指定同样条件的另一 **USE** 语句中由文卷名 1 引用的那些文卷除外。

c. 若指出 **OUTPUT**,则在 **USE** 语句中描述的条件出现时,对任何以 **OUTPUT** 方式打开或正在打开的文卷执行相关的过程,对指定同样条件的另一 **USE** 语句中由文卷名 1 引用的文卷除外。

d. 若指出 **I-O**,则在 **USE** 语句中描述的条件出现时,对任何以 **I-O** 方式打开或正在打开的文卷执行相关的过程,对指定同样条件的另一 **USE** 语句中由文卷名 1 引用的文卷除外。

e. 若指出 **EXTEND**,则在 **USE** 语句中描述的条件出现时,对任何以 **EXTEND** 方式打开或正在打开的文卷执行相关的过程,对指定同样条件的另一 **USE** 语句中由文卷名 1 引用的文卷除外。

(6) **USE** 过程执行以后,控制转向输入输出控制系统中的调用程序。若 **I-O** 状态值不指出为严重的输入输出错误,则输入输出控制系统返回控制到其执行引起例外的输入输出语句后面的下一可执行语句。若 **I-O** 状态值指出一个严重错误,则由实现者确定要做什么动作(见 1.3.4 **I-O** 状态)。

(7) 在 **USE** 过程中,不能有任何这样的一种语句执行,它能使得先前已被调用,但其控制还未返回到调用程序的 **USE** 过程执行。

#### 4.9 **WRITE** 语句

##### 4.9.1 功能

**WRITE** 语句释放一个逻辑记录到输出文卷或输入输出文卷。

##### 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 不能引用同一个存储区。

(2) 记录名 1 是数据部文卷节中逻辑记录的名,并且可以受限。

(3) 如果没有为相关文卷名指明可应用的 **USE AFTER STANDARD EXCEPTION** 过程,则必须指出 **INVALID KEY** 短语。

##### 4.9.4 一般规则

(1) 在这个语句执行时,与记录名 1 相关联的文卷名引用的文卷必须已按 **OUTPUT**、**I-O**

**EXTEND** 方式打开(见 4.4 **OPEN** 语句)。

(2) 成功地执行 **WRITE** 语句所释放的逻辑记录在记录区中不再可用,除非与记录名 1 相关的文卷名在 **SAME RECORD AREA** 子句中指明。该逻辑记录对该程序也是可用的。但这时它是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。该逻辑记录对于和记录名 1 相关的文卷同样也是可用的。

(3) 带有 **FROM** 短语的 **WRITE** 语句执行结果等价于按序执行如下语句:

a. 根据 **MOVE** 语句指出的规则执行语句:

**MOVE** 标识符 1 **TO** 记录名 1

其后执行：

b. 无 **FROM** 短语的相同的 **WRITE** 语句。

(4) **WRITE** 语句执行完成之后,尽管由记录名 1 引用的存储区中的信息是不可用的,但由标识符 1 引用的这个域中的信息却是可用的。

(5) 文卷位置指示符不受 **WRITE** 语句执行的影响。

(6) **WRITE** 语句的执行使得与记录名 1 相关联的文卷名的 **I-O** 状态值被更新(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 所引用的记录中的字符位置数不能大于 **RECORD IS VARYING** 子句所允许的  
最大字符位置数或小于最小字符位置数,该子句与记录名 1 相关的文卷名相关联。不论两种情况的哪  
一种情况出现,**WRITE** 语句的执行是不成功的,**WRITE** 操作不进行,记录区的内容不受影响,而与记  
录名 1 相关联文卷的 **I-O** 状态指示着发生的条件(见 1.3.4 **I-O** 状态)。

(9) 在带有 **NOT INVALID KEY** 短语的 **WRITE** 语句执行期间,若无效键条件不出现,则在下述  
适当的时间控制转到命令语句 2:

a. 若 **WRITE** 语句的执行是成功,则在记录被写且修改与记录名 1 相关联文卷名的 **I-O** 状态以后。

b. 若由于非无效键条件的原因使 **WRITE** 语句的执行不成功,则在修改与记录名 1 相关联文卷名  
的 **I-O** 状态以后,若 **USE AFTER STANDARD EXCEPTION PROCEDURE** 语句指出了可应用于记  
录名 1 相关文卷名的过程,那么还要在执行这过程以后。

(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) **WRITE** 语句的执行导致释放记录域的内容。**MSCS** 按这样的方法利用记录键的内容,该记录  
键的后继存取可以基于这些指定的记录键中的任何一个进行。

(12) 主记录键的值在文卷的记录中必须是唯一的。

(13) 作为主记录键指明的数据项必须在 **WRITE** 语句执行之前由程序设置所要求的值。

(14) 若文卷以顺序存取方式打开,则记录必须按文卷对比序列以主记录键的递增次序释放给  
**MSCS**。

若文卷以扩充方式打开时,释放给 **MSCS** 的第一个记录的主记录键值必须大于该文卷中现存的最  
主记录键值。

(15) 若文卷以随机存取方式打开,则记录可以按程序指出的任何次序释放给 **MSCS**。

(16) 当索引文卷的文卷控制款中指出了 **ALTERNATE RECORD KEY** 子句时,仅当对那个  
数据项指出 **DUPLICATES** 短语时,次记录键的值才可以是不唯一的。在这种情况下,**MSCS** 提供存储  
记录,以便在顺序存取时那些记录的检索次序就是把它释放到 **MSCS** 上的次序。

(17) 在下列情况下存在 **INVALID KEY** 条件:

a. 当按输出或扩充方式打开的文卷指出顺序存取方式,而且主记录键的值不大于前一记录的主  
记录键的值,或

b. 当文卷按输出或输入输出方式打开,而且主记录键的值等于文卷中已经存在的主记录键的值,  
或

c. 当文卷按输出或输入输出方式打开,而且那些不允许重复的次记录键的值等于文卷中已经  
存在的记录的对应数据项,或

d. 当企图进行超出外部定义的文卷边界的写动作发生。

(18) 当识别出 **INVALID KEY** 条件使 **WRITE** 语句的执行不成功时,则记录区的内容不受影响,

并且与记录名 1 相关联的文卷名的 I-O 状态被置为指出引起这一条件的值。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。