第7章　ファイル入出力

指導目標

ファイルとは、記憶媒体上に書き出されたり、読み出されたりするレコードの集まりである。ＣＯＢＯＬでは、ファイル編成、ファイルの処理方法や手順を選択して利用できる。

本章では、索引編成ファイル、相対編成ファイルについて学習する。また、レコードの書換え、レコードの削除、レコードの位置づけを行う入出力操作の概念を理解させる。

索引ファイルでは、主レコードキー、副レコードキー、キーの文字の大小順序についての基礎知識とファイル処理方法の乱呼出し法や動的呼出し法の使い方を学習して、ファイル更新のレコードの変更、挿入（追加）、削除処理の特徴を学習する。

相対ファイルでは、相対レコード番号に関する基礎知識とファイル処理方法の乱呼出し法や動的呼出し法の使い方を学習する。

第５章と本章で入出力操作の概念と特徴を理解させて、順ファイル、索引ファイル、相対ファイルの使い分けができるように指導する。

本章で学習する入出力文は、次のとおり。

・ＯＰＥＮ文
・ＷＲＩＴＥ文
・ＲＥＡＤ文
・ＲＥＷＲＩＴＥ文
・ＤＥＬＥＴＥ文
・ＳＴＡＲＴ文
・ＣＬＯＳＥ文

内容のあらまし

内　　容	説　明	議　論	机上実習	計算機実習
索引ファイル	索引ﾌｧｲﾙの特徴を説明する。	順ﾌｧｲﾙとの構造の違いを議論する。		
索引ﾌｧｲﾙの定義	定義の全体像と索引編成を定義するSELECT句を説明する。	順ﾌｧｲﾙとの違いについて議論する。	ﾌｧｲﾙ領域の容量計算をｷｰの桁数を変えて演習する。	簡単なﾌﾟﾛｸﾞﾗﾑを作成して実行してみる
ＯＰＥＮ文	ｵｰﾌﾟﾝﾓｰﾄﾞと使用できる命令文の関係を説明する。			ｵｰﾌﾟﾝﾓｰﾄﾞに違反した命令文を実行してみる。
ＷＲＩＴＥ文	ﾚｺｰﾄﾞの書き出しと主ﾚｺｰﾄﾞｷｰの関係を説明する。			順呼出しと乱呼出しの書き出しを演習させる。
ＲＥＡＤ文	ﾚｺｰﾄﾞの取り出しについて乱呼出し法を説明して順編成との違いを指導する。	INVALIDとENDの違いについて議論する。		
ＲＥＷＲＩＴＥ文	ﾚｺｰﾄﾞの書き換えについて説明する。	WRITE文との違いについて議論する。		
ＤＥＬＥＴＥ文	ﾚｺｰﾄﾞの削除と主ﾚｺｰﾄﾞｷｰの関係について説明する。	順呼出しと乱呼出しの違いについて議論する。		全ﾚｺｰﾄﾞを削除したﾌｧｲﾙに使える命令文を演習させる
ＳＴＡＲＴ文	ﾌｧｲﾙ中の特定ﾚｺｰﾄﾞに位置づける方法から索引構造を説明する	順ﾌｧｲﾙ、相対ﾌｧｲﾙでの位置づけの概念を議論する。	ｷｰの桁数を変化させるﾃﾞｰﾀ記述のｺｰﾃﾞｨﾝｸﾞを実習する	
ＣＬＯＳＥ文	ﾌｧｲﾙの処理の終了の意味を説明する。			LOCKされたﾌｧｲﾙをOPENしてｼｽﾃﾑの動作を調べる。
相対ファイル	相対ﾌｧｲﾙの特徴を相対ﾚｺｰﾄﾞ番号を中心に説明する。 命令文については索引ﾌｧｲﾙと同様に説明する。	索引ﾌｧｲﾙと比較して相対ﾌｧｲﾙを利用する場合の注意点を明確にする。		索引ﾌｧｲﾙのﾚｺｰﾄﾞｷｰ(数字)を相対ﾚｺｰﾄﾞ番号にして処理速度を比較してみる。

応用編

第７章　ファイル入出力

７．１　索引ファイル

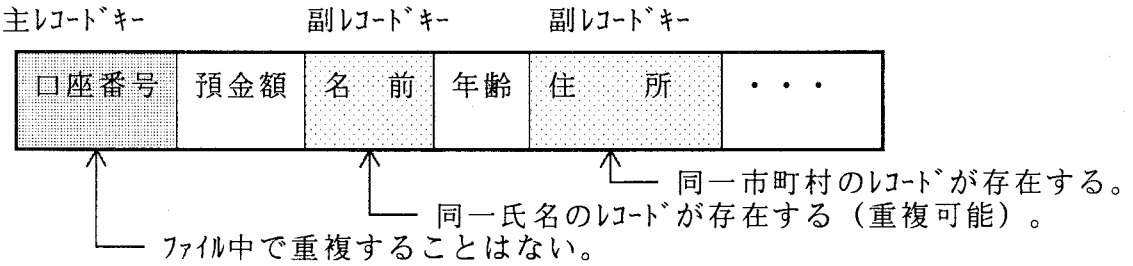
索引ファイル（indexed　file）は、レコード中の特定のデータ項目をキーとしてそのキーの値によってレコードをアクセスする大記憶ファイルである。

```
SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語
    ORGANIZATION  IS   INDEXED
```

レコード中のキーとなるデータ項目をレコードキー（record　key）という。レコードキーには「主レコードキー」と「副レコードキー」がある。副レコードキーはレコード中に複数指定できる。

```
    RECORD  KEY  IS  主ﾚｺｰﾄﾞｷｰ
    ALTERNATE  RECORD  KEY  IS  副ﾚｺｰﾄﾞｷｰ  DUPLICATES  ･･･
```

索引ファイル中のレコードは、主レコードキーを用いてレコードの挿入、削除、変更を行うので各レコードの主レコードキーの値はファイル中で一意である。
副レコードキーは、索引ファイルのレコードを検索する代替手段に用いる。この副レコードキーの値は、ファイル中で一意である必要はない。

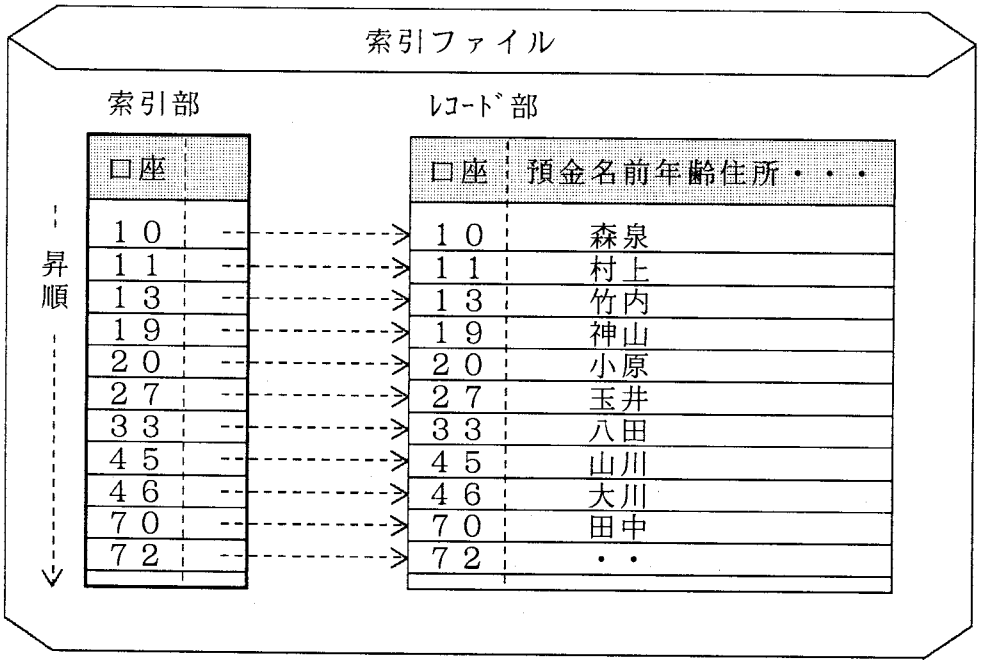


レコードキー（主、副）のデータ項目は英数字項目として定義しなければならない。
レコードキーのサイズと個数について特に規定されていないが、通常１～２５５バイトの範囲である。また、レコードには１個の主レコードキーと最大２５４個の副レコードキーが付けられる。

```
SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語1
    ORGANIZATION  IS  INDEXED
    RECORD  KEY  IS  口座番号
    ALTERNATE  RECORD  KEY  IS  名前  WITH  DUPLICATES
    ALTERNATE  RECORD  KEY  IS  住所  WITH  DUPLICATES.

01  索引ﾚｺｰﾄﾞ.
    03  口座番号.                                          ←主ﾚｺｰﾄﾞｷｰ
        05  口座        PICTURE  9(8)  USAGE  BINARY.      集団項目にすると
    03  預金額          PICTURE  S9(9).                    英数字となる
    03  名前            PICTURE  X(20).    ←副ﾚｺｰﾄﾞｷｰ
    03  年齢            PICTURE  9(3).
    03  住所            PICTURE  X(50).    ←副ﾚｺｰﾄﾞｷｰ
          :
```

７．１．１　索引ファイルの概念

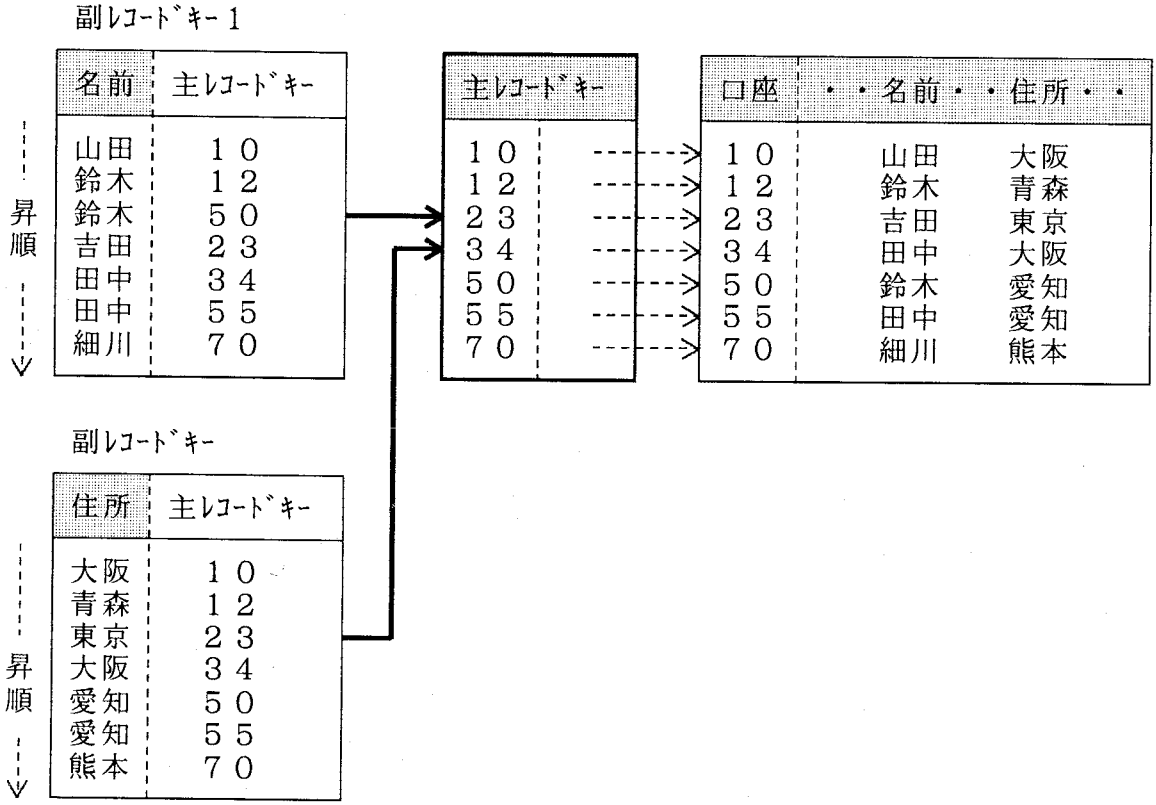
索引ファイルは、キー情報を管理する索引部とレコード部でファイルが構成されている。



注：現実の実現方式では、索引部が階層化され少ないアクセスでレコードを検索できる。
　　また、レコード部もレコードの挿入が効率的に行えるような構造になっている。

副レコードキーの概念

副レコードキーで検索要求がだされると副レコードキーを探し、一致した副レコードキーに付いている主レコードキーでレコードをアクセスする。



注：上記の例は概念であり、副レコードキーもコンピュータの大小順序に並べられる。

７．１．２　索引ファイルの定義

ＣＯＢＯＬプログラムにおける索引ファイル定義の全体像を解説する。

索引ファイル定義の概要

```
IDENTIFICATION  DIVISION.
PROGRAM-ID.   INDEX-F.

ENVIRONMENT  DIVISION.

INPUT-OUTPUT  SECTION.
FILE-CONTROL.
    SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語1                                    ①
            ORGANIZATION  IS  INDEXED                                        ②
            ACCESS  MODE  IS  呼出し法                                       ③
            RECORD  KEY  IS  主ﾚｺｰﾄﾞｷｰ                                      ④
            ALTERNATE  RECORD  KEY  IS  副ﾚｺｰﾄﾞｷｰ  WITH  DUPLICATES  ⑤
            FILE  STATUS  IS  ﾌｧｲﾙｽﾃｰﾀｽ.                                    ⑥

DATA  DIVISION.
FILE  SECTION.

FD    ﾌｧｲﾙ名
      BLOCK  CONTAINS  整数2  RECORDS                                        ⑦
      RECORD  CONTAINS  整数3  CHARACTERS.
01    ﾚｺｰﾄﾞ記述項.                                                           ⑧
      03  主ﾚｺｰﾄﾞｷｰ    PICTURE  X(ｻｲｽﾞ).                                   ⑨
      03  ﾃﾞｰﾀ名
          :
      03  副ﾚｺｰﾄﾞｷｰ    PICTURE  X(ｻｲｽﾞ).                                   ⑩
          :
      03  副ﾚｺｰﾄﾞｷｰ-2  PICTURE  X(ｻｲｽﾞ).
          :

WORKING-STORAGE  SECTION.
01    ﾌｧｲﾙｽﾃｰﾀｽ        PICTURE  X(2).


PROCEDURE  DIVISION.

      OPEN  入出力ﾓｰﾄﾞ  ﾌｧｲﾙ名                                                ⑪

      WRITE  ﾚｺｰﾄﾞ名  INVALID  KEY  ～                                       ⑫

      READ  ﾌｧｲﾙ名  AT  END／INVALID  KEY  ～

      REWRITE  ﾚｺｰﾄﾞ名  INVALID  KEY  ～

      START  ﾌｧｲﾙ名  KEY  IS  比較演算子  ﾃﾞｰﾀ名  INVALID  KEY  ～

      DELETE  ﾌｧｲﾙ名  INVALID  KEY  ～

      CLOSE  ﾌｧｲﾙ名.
```

索引ファイル定義の説明

①ファイル名
ＣＯＢＯＬプログラム上の論理ファイル名を定義する。

②ファイル編成
ファイル編成として索引ファイル（INDEXED）を定義する。

③呼出し方式
索引ファイルの呼出し方式を定義する。索引ファイルでは全ての方式をサポートしている。

ACCESS	MODE	IS	SEQUENTIAL	－	順呼出し法
ACCESS	MODE	IS	RANDOM	－	乱呼出し法
ACCESS	MODE	IS	DYNAMIC	－	動的呼出し法

④主レコードキー（必須）
索引ファイルはレコード中のキーの値によってレコードを取り出す大記憶ファイルであり、そのキーを主レコードキーとして定義する。
主レコードキーの値は、ファイル中で一意でなければならない。

⑤副レコードキー（任意）
副レコードキーは主レコードキーとは別の検索手段を提供する。
「WITH　DUPLICATES」を定義すると副レコードキーの値は、ファイル中に重複してもよい。また、複数のデータ項目を副レコードキーとして別々に定義できる。

⑥ファイルステータス
ＣＯＢＯＬでの入出力の実行結果（入出力状態）を返す領域（2ﾊﾞｲﾄ）を定義する。
索引ファイルでは、コード「０２」として「ＲＥＡＤの実行は成功したが重複キーが検出された」という状態があり、場合によっては入出力文の後でファイルステータスを調べる必要がある。

⑦ブロック、レコード
レコードとブロックの構成を定義する。

⑧レコード記述
入出力文で外部ファイルとのレコードのやりとりを行う領域と、そのデータ構造を定義する。

⑨主レコードキー
主レコードキーを「英数字項目」として定義する。
サイズは処理系にもよるが１～２５５バイトの範囲で指定できる。

⑩副レコードキー
副レコードキーを「英数字項目」として定義する。

⑪入出力モード
索引ファイルの処理モードを指定する。
・INPUT　－　入力ファイル
・OUTPUT　－　出力ファイル
・I-O　－　入出力両用ファイル
・EXTEND　－　拡張ファイル

⑫入出力文
入出力の動作を指定する。
OPEN文を最初に実行して、その後、入出力モードに対応した命令文を実行する。
最後はCLOSE文を実行する。

７．１．３　ＳＥＬＥＣＴ

索引ファイルに関する属性を定義する。

```
SELECT  [OPTIONAL]  ﾌｧｲﾙ名  ASSIGN  TO  作成者語1／定数1
       [RESERVE  整数1  AREA／AREAS]
       [ORGANIZATION  IS]  INDEXED
       [ACCESS  MODE  IS  SEQUENTIAL／RANDOM／DYNAMIC]
        RECORD  KEY  IS  ﾃﾞｰﾀ名1
       [ALTERNATE  RECORD  KEY  IS  ﾃﾞｰﾀ名2  [WITH  DUPLICATES] ] ･･･
       [FILE  STATUS  IS  ﾃﾞｰﾀ名3]
```

ＳＥＬＥＣＴ句は索引ファイルの呼出し法と、レコード中のレコードキー（主、副）の位置とサイズを定義する。

呼出し法（ACCESS　MODE）には３種類あり、索引ファイルでは全ての呼出し法が使える。

①順呼出し（SEQUENTIAL）
　レコードキーの値の文字（英数字）の大小順序（昇順）に従って呼び出される。
　索引ファイルの生成時に使用すると高速にファイルが生成できる。

②乱呼出し（RANDOM）
　レコード中のレコードキー（主、副）の値で呼び出すレコードが決まる。

③動的呼出し（DYNAMIC）
　レコードは乱呼出しと順呼出しを組み合わせてアクセスできる。

索引ファイルのファイルステータス

上位桁	下位桁	意味	
0	0	成功	情報なし
0	2		重複ｷｰがある （副ﾚｺｰﾄﾞｷｰのREAD、WRITE等）
0	4		ﾚｺｰﾄﾞ長がﾌｧｲﾙ属性と合わない
1	0	ﾌｧｲﾙの終了	順呼出しのREADでﾌｧｲﾙの終了になった
2	1	無効ｷｰ	順呼出しで順序誤りがある （例えば、WRITEでキーが昇順になっていない）
2	2		主ﾚｺｰﾄﾞｷｰと同じ値が既に存在している
2	3		ﾌｧｲﾙに存在しないﾚｺｰﾄﾞを呼び出そうとした
2	4		ﾌｧｲﾙの区域外に書き出そうとした
3	0	永続ｴﾗｰ	情報なし
3	5		OPEN I-O/INPUTでﾌｧｲﾙが存在しない
3	7		OPENﾓｰﾄﾞと物理ﾌｧｲﾙの属性が一致しない
3	8		CLOSE WITH LOCKのﾌｧｲﾙを再OPENした
3	9		物理ﾌｧｲﾙ属性とﾌｧｲﾙ定義との間で矛盾を検出
4	1	論理ｴﾗｰ	OPEN済みﾌｧｲﾙに再OPENを実行した
4	2		OPENされていないﾌｧｲﾙにCLOSEを実行した
4	3		順呼出しでDELETE/REWRITEの前がREADでない
4	4		区域外書き出し
4	6		終了したﾌｧｲﾙにREADを実行した
4	7		OUTPUTﾓｰﾄﾞのﾌｧｲﾙにREADを実行した
4	8		INPUTﾓｰﾄﾞのﾌｧｲﾙにWRITEを実行した
4	9		I-Oﾓｰﾄﾞ以外のﾌｧｲﾙにDELETE/REWRITEを実行した
9	x	その他ｴﾗｰ	作成者が定義する

7．1．4　ＯＰＥＮ

ＯＰＥＮ（開く）文は、ファイルの処理を行うための準備をする。

```
OPEN { INPUT    {ファイル名} ･･･ }
     { OUTPUT   {ファイル名} ･･･ } ･･･
     { I-O      {ファイル名} ･･･ }
     { EXTEND   {ファイル名} ･･･ }
```

索引ファイルのＯＰＥＮモードと入出力文の組合せは、次のとおり。

呼出し法	入出力文	ＯＰＥＮモード			
		INPUT	OUTPUT	I-O	EXTEND
順呼出し	READ	○	－	○	－
	WRITE	－	○	－	○
	REWRITE	－	－	○	－
	START	○	－	○	－
	DELETE	－	－	○	－
乱呼出し	READ	○	－	○	－
	WRITE	－	○	○	－
	REWRITE	－	－	○	－
	START	－	－	－	－
	DELETE	－	－	○	－
動的呼出し	READ	○	－	○	－
	WRITE	－	○	○	－
	REWRITE	－	－	○	－
	START	○	－	○	－
	DELETE	－	－	○	－

一つのファイルを同じ実行単位で、ＩＮＰＵＴ指定、ＯＵＴＰＵＴ指定、ＥＸＴＥＮＤ指定、Ｉ－Ｏ指定を書いて何回も開くことができる。ただし、２回目以降のＯＰＥＮを実行するためにはＬＯＣＫ指定のないＣＬＯＳＥを実行しておくこと。

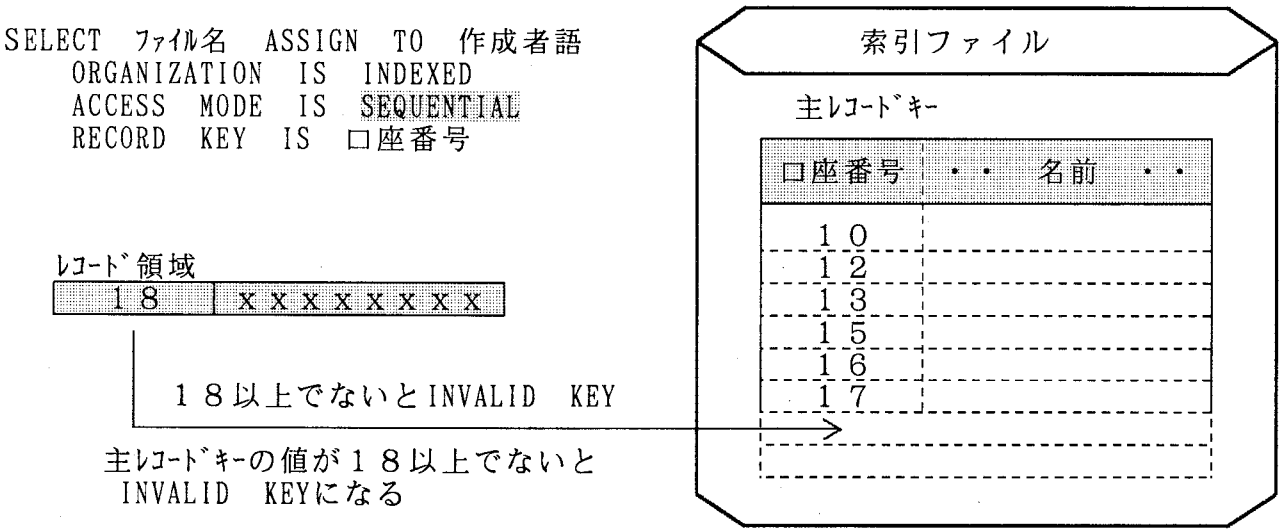
```
PROCEDURE  DIVISION.

    OPEN  OUTPUT  索引ファイル
          ･･･
    CLOSE  索引ファイル

    OPEN INPUT  索引ファイル
          ･･･
    CLOSE  索引ファイル

    OPEN  I-O  索引ファイル
          ･･･
    CLOSE  索引ファイル  WITH  LOCK

    OPEN  INPUT  索引ファイル          ←ｵｰﾌﾟﾝできない
```

７．１．５　ＷＲＩＴＥ

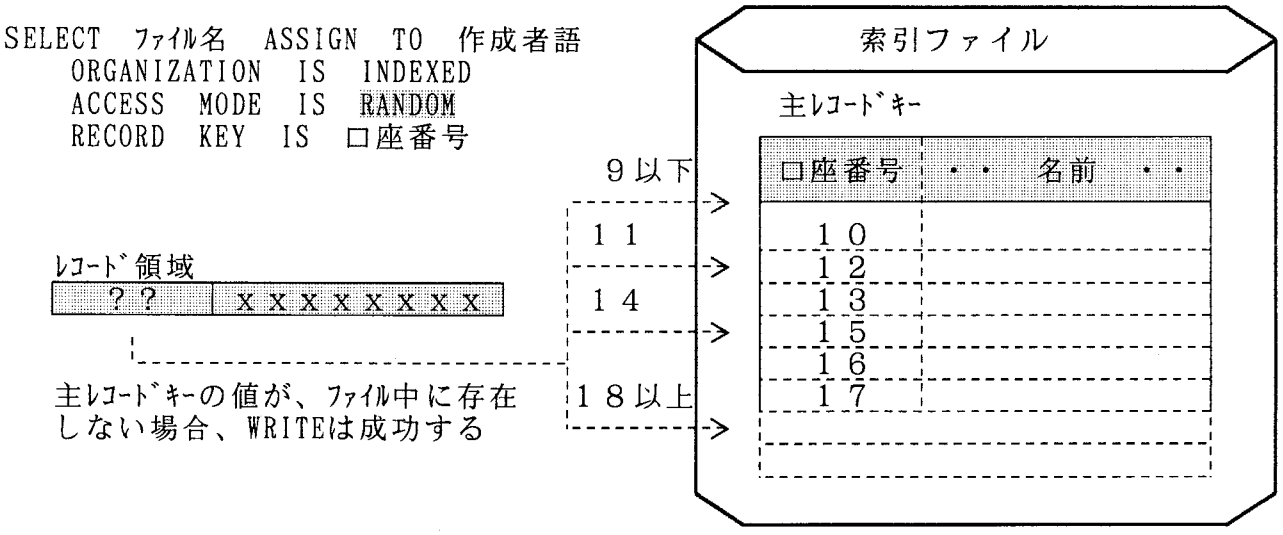
出力（OUTPUT）ファイル、入出力両用（I-O）ファイルにおいて、レコード中の主レコードキー（英数字項目）の値を索引としてレコードを書き出す。

```
WRITE  ﾚｺｰﾄﾞ名1  [FROM  一意名1]
       [INVALID  KEY  無条件文1]
       [NOT  INVALID  KEY  無条件文2]
 [END-WRITE]
```

順呼出し法（ACCESS　MODE　IS　SEQUENTIAL）では、主レコードキーの値が昇順になるように書き出さなければならない。直前の主レコードキーの値より大きくないと無効キー条件（INVALID　KEY）になる。



乱呼出し法（RANDOM）や動的呼出し法(DYNAMIC）では、任意の順序（値）で書き出せる。
また、主レコードキーの値と等しいレコードが既にファイル中にあると無効キー条件（INVALID　KEY）になる。



７．１．６　ＲＥＡＤ

順呼出し法では、次のレコードを取り出す。乱呼出し法では、主レコードキー項目の値で指定されたレコードを取り出す。

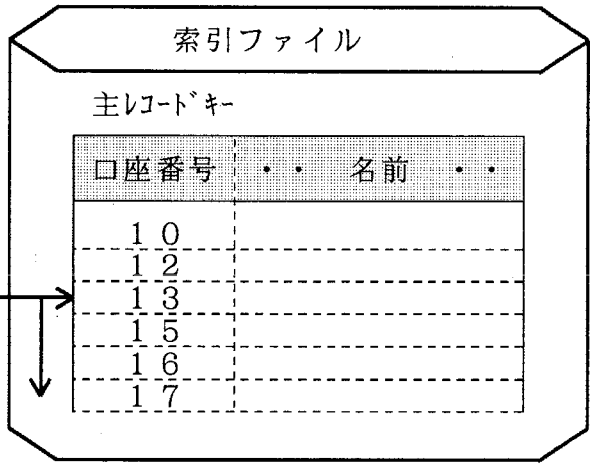
```
READ  ﾌｧｲﾙ名1  [NEXT]  RECORD  [INTO  一意名1]
      [AT  END  無条件文1]
      [NOT  AT  END  無条件文2]
 [END-READ]

READ  ﾌｧｲﾙ名1  RECORD  [INTO  一意名1]
      [KEY  IS  ﾃﾞｰﾀ名1]
      [INVALID  KEY  無条件文1]
      [NOT  INVALID  KEY  無条件文2]
 [END-READ]
```

順呼出し法（ACCESS　MODE　IS　SEQUENTIAL）では、「READ　ﾌｧｲﾙ名　AT　END　～」を使用する。レコードはファイルの最初から取り出す。
途中から取り出す場合は、ＳＴＡＲＴで指定のキーに位置づけてからＲＥＡＤを行う。

乱呼出し法（RANDOM）では、「READ　ﾌｧｲﾙ名　INVALID　KEY　～」を使用する。
取り出すレコードのレコードキー（主、副）の値は、事前にレコード領域のレコードキー項目に設定してからＲＥＡＤを実行する。

動的呼出し法（DYNAMIC）の場合は、「READ　ﾌｧｲﾙ名　INVALID　KEY　～」と「READ　ﾌｧｲﾙ名 NEXT　AT　END　～」を組み合わせて使用できる。

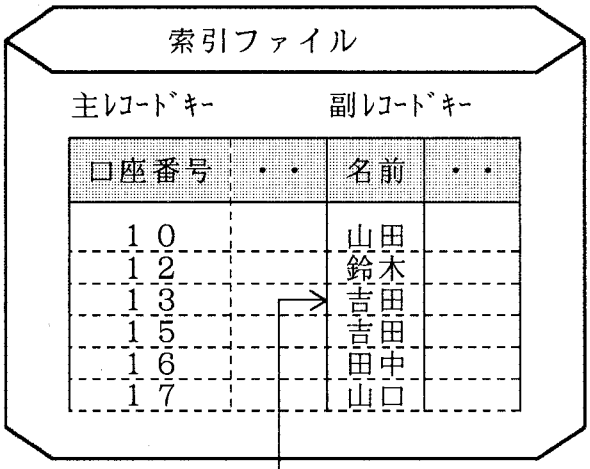
```
      SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語
          ORGANIZATION  IS  INDEXED
          ACCESS  MODE  IS  DYNAMIC
          RECORD  KEY   IS  口座番号
          FILE STATUS   IS  ﾌｧｲﾙｽﾃｰﾀｽ.
            :
FILE  SECTION.
FD  ﾌｧｲﾙ名  ～
01  ﾚｺｰﾄﾞ名.
    03  口座番号 PICTURE  X(2).
    03  ﾃﾞｰﾀ1    PICTURE  X(20).
          :

PROCEDURE  DIVISION.

    OPEN  INPUT  ﾌｧｲﾙ名.
RAN-READ.
    MOVE  "１３"  TO  ﾚｺｰﾄﾞｷｰ.
    READ  ﾌｧｲﾙ名  INVALID  KEY
       DISPLAY "指定のﾚｺｰﾄﾞｷｰがない"
          :
SEQ-READ.
    READ  ﾌｧｲﾙ名  NEXT  AT  END
         DISPLAY  "ﾌｧｲﾙの最後です"
           :
    IF  続行  GO TO  SEQ-READ.
      :
```



副レコードキーによるレコードの取り出しには、ＲＥＡＤの「ＫＥＹ」句を使う。

重複（等しい）する副レコードキーの値が存在する場合は「WITH DUPLICATES」句を指定する。重複する副レコードキーの存在の有無は、ＲＥＡＤ実行後にファイルステータスを調べることで判る。

```
    SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語
        ORGANIZATION  IS  INDEXED
        ACCESS  MODE  IS  RANDOM
        RECORD  KEY  IS  口座番号
        ALTERNATE RECORD  KEY  IS  名前  WITH  DUPLICATES
        FILE STATUS  IS  ﾌｧｲﾙｽﾃｰﾀｽ.
          :
FILE  SECTION.
FD  ﾌｧｲﾙ名  ～
01  ﾚｺｰﾄﾞ名.
   03  口座番号 PICTURE  X(2).
   03  ﾃﾞｰﾀ1    PICTURE  X(20).
   03  名前     PICTURE  X(10).

WORKING-STORAGE  SECTION.
01  ﾌｧｲﾙｽﾃｰﾀｽ   PICTURE  X(2).
```




```
PROCEDURE  DIVISION.
    :
    OPEN  I-O  ﾌｧｲﾙ名.

    MOVE  "吉田"  TO  名前
    READ  ﾌｧｲﾙ名  KEY  IS  名前
          INVALID  KEY  DISPLAY  "指定の名前が見つかりません"
    END-READ
    IF  ﾌｧｲﾙｽﾃｰﾀｽ  =  "02"  THEN
        DISPLAY "次のﾚｺｰﾄﾞにも同じ名前のｷｰがあります"
    END-IF
```

注：ファイルステータスの「０２」の意味は次のとおり。
「READ文で読み込んだレコードの参照キーの値が、その索引における次のレコードの参照キーと等しい」

7．1．7　REWRITE

入出力両用モード（I-0）でオープンされたファイルのレコードを書き換える。

```
REWRITE  ﾚｺｰﾄﾞ名1  [FROM  一意名1]
         [INVALID  KEY  無条件文1]
         [NOT  INVALID  KEY  無条件文2]
 [END-REWRITE]
```

順呼出し法（ACCESS　MODE　IS　SEQUENTIAL）では、最後にＲＥＡＤで読み込んだレコードが書き換えられる。

```
        SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語
            ORGANIZATION  IS  INDEXED
            ACCESS  MODE  IS  SEQUENTIAL
            RECORD  KEY  IS  ﾚｺｰﾄﾞｷｰ
            FILE STATUS  IS  ﾌｧｲﾙｽﾃｰﾀｽ.
              :
    FILE  SECTION.
    FD  ﾌｧｲﾙ名  ～
    01  ﾚｺｰﾄﾞ名.                              順呼出し法でﾚｺｰﾄﾞを書き換える場合
        03  ﾚｺｰﾄﾞｷｰ     PICTURE  X(10).      は、主ﾚｺｰﾄﾞｷｰを変更してはならない
              :                               副ﾚｺｰﾄﾞｷｰの値は変更できる
    PROCEDURE  DIVISION.
    DECLARATIVES.
    XXX  SECTION.  USE  AFTER  STANDARD  ERROR  ﾌｧｲﾙ名.
          IF  ﾌｧｲﾙｽﾃｰﾀｽ  =  "21"  OR  "41"  THEN
              DISPLAY  "REWRITEの前で主ﾚｺｰﾄﾞｷｰの値が変更された"
              DISPLAY  "または、REWRITEの前が成功したREADではない"
          END-IF.
    END  DECLARATIVES.
    AAA  SECTION.

        OPEN I-0  ﾌｧｲﾙ名.
    SEQ-REC.
        READ  ﾌｧｲﾙ名  AT  END  GO  TO  END-REC.      ﾚｺｰﾄﾞを書き換えする時
        IF  書換対象ﾚｺｰﾄﾞ  THEN                      はREWRITEの直前の文が
            MOVE  更新ﾃﾞｰﾀ  TO  ﾚｺｰﾄﾞ項目            成功したREADであること
            REWRITE  ﾚｺｰﾄﾞ名
        END-IF
        GO  TO  SEQ-REC.
    END-REC.
        CLOSE  ﾌｧｲﾙ名.
```

乱呼出し法（RANDOM）、動的呼出し法（DYNAMIC）では、主レコードキーを指定してファイル中のレコードを直接書き換えられる。

７．１．８　ＤＥＬＥＴＥ

入出力両用（I-0）モードでオープンされたファイルのレコードを取り除く（削除）。

```
DELETE  ﾌｧｲﾙ名1  RECORD
      [INVALID  KEY  無条件文1]
      [NOT  INVALID  KEY  無条件文2]
 [END-DELETE]
```

順呼出し法（ACCESS　MODE　IS　SEQUENTIAL）では、最後にＲＥＡＤで読み込んだレコードが取り除かれる。また、この時には「INVALID　KEY」、「NOT　INVALID　KEY」を指定してはいけない。

```
        SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語
              ORGANIZATION  IS  INDEXED
              ACCESS  MODE  IS  SEQUENTIAL
              RECORD  KEY  IS  ﾚｺｰﾄﾞｷｰ
              FILE STATUS  IS  ﾌｧｲﾙｽﾃｰﾀｽ.
               :
PROCEDURE  DIVISION.
DECLARATIVES.
XXX  SECTION.  USE  AFTER  STANDARD  ERROR  ﾌｧｲﾙ名.
      IF  ﾌｧｲﾙｽﾃｰﾀｽ  =  "41" | OR  "21" | THEN
          DISPLAY  "DELETEの前が成功したREADではない"
      END-IF.
END  DECLARATIVES.
AAA  SECTION.

    OPEN I-0  ﾌｧｲﾙ名.
SEQ-REC.
    READ  ﾌｧｲﾙ名  AT  END  GO  TO  END-REC.
    IF  削除対象ﾚｺｰﾄﾞ  THEN
        DELETE  ﾌｧｲﾙ名
    END-IF
    GO  TO  SEQ-REC.
END-REC.
    CLOSE  ﾌｧｲﾙ名.
```

ﾚｺｰﾄﾞを削除する場合は
DELETEの直前の命令文が
成功したREADであること

乱呼出し法（RANDOM）、動的呼出し法（DYNAMIC）では、主レコードキーを指定してファイル中のレコードを直接取り除くことができる。

```
        SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語
            ORGANIZATION  IS  INDEXED
            ACCESS  MODE  IS  RANDOM
            RECORD  KEY  IS  ﾚｺｰﾄﾞｷｰ
            FILE STATUS  IS  ﾌｧｲﾙｽﾃｰﾀｽ.
              :
PROCEDURE  DIVISION.
RAN.  SECTION.

      OPEN I-0  ﾌｧｲﾙ名.
          :
      MOVE  削除ｷｰ  TO  ﾚｺｰﾄﾞｷｰ
      DELETE  ﾌｧｲﾙ名  INVALID  KEY
            DISPLAY  "指定のﾚｺｰﾄﾞｷｰは存在しません"
      END-DELETE
```

7．1．9　START

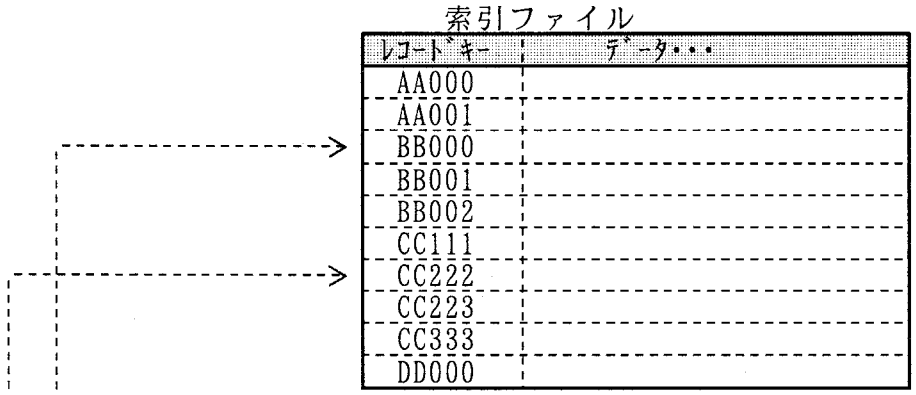
順呼出し法、動的呼出し法で主レコードキーの値とその比較条件を「ＫＥＹ」句で指定して、ファイル中の特定レコードに位置づける。
レコードの取り出しは、ＲＥＡＤ　ＮＥＸＴを使用する。

```
START  ﾌｧｲﾙ名1  [KEY { IS EQUAL TO
                    | IS =
                    | IS GREATER THAN
                    | IS >
                    | IS NOT LESS THAN
                    | IS NOT <
                    | IS GREATER THAN OR EQUAL TO
                    | IS >= } ﾃﾞｰﾀ名1]
       [INVALID KEY 無条件文1]
       [NOT INVALID KEY 無条件文2]
 [END-START]
```

ＫＥＹ句を省略すると「IS　EQUAL　TO」が指定されたものとみなす。
ＫＥＹ句のデータ名１は、ＳＥＬＥＣＴ句で主レコードキー、または、副レコードキーと定義したデータ項目を指定する。

```
       SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語
           ORGANIZATION  IS  INDEXED
           ACCESS  MODE  IS  DYNAMIC
           RECORD  KEY  IS  ﾚｺｰﾄﾞｷｰ
           FILE STATUS  IS  ﾌｧｲﾙｽﾃｰﾀｽ.
             :
FILE  SECTION.
FD  ﾌｧｲﾙ名  ～
01  ﾚｺｰﾄﾞ名.
    03  ﾚｺｰﾄﾞｷｰ    PICTURE  X(10).
          :
PROCEDURE  DIVISION.
AAA.  SECTION.

      OPEN I-O  ﾌｧｲﾙ名.
          :
      MOVE  "参照ｷｰ"  TO  ﾚｺｰﾄﾞｷｰ          ← ｷｰを指定して
      START  ﾌｧｲﾙ名                          ← STARTを実行
        INVALID  KEY
            DISPLAY  "指定のﾚｺｰﾄﾞｷｰに位置づけられません"
            MOVE  "失敗"  TO  ﾌﾗｸﾞ
        NOT  INVALID  KEY
            MOVE  "成功"  TO  ﾌﾗｸﾞ
      END-START.
      IF  ﾌﾗｸﾞ  =  "成功"  THEN
          READ  ﾌｧｲﾙ名  NEXT  RECORD  AT  END  ～   ← ﾚｺｰﾄﾞの取り出しはREAD
      END-IF.
```

ＲＥＡＤでは、レコードキーの値として全部を指定しないといけないが、ＳＴＡＲＴのＫＥＹ句ではレコードキーの部分文字列（項目の先頭から）を指定できる。
ＫＥＹ句のデータ項目はレコードキーの左端の文字位置に等しく、サイズはレコードキー（主、副）以下であること。




```
    SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語
        ORGANIZATION  IS  INDEXED
        ACCESS  MODE  IS  SEQUENTIAL
        RECORD  KEY  IS  ﾚｺｰﾄﾞｷｰ
        FILE STATUS  IS  ﾌｧｲﾙｽﾃｰﾀｽ.
          :
FILE  SECTION.
FD  ﾌｧｲﾙ名  ～
01  ﾚｺｰﾄﾞ名.
    03  ﾚｺｰﾄﾞｷｰ      PICTURE  X(05).
    03  ｷｰｻｲｽﾞ01   REDEFINES  ﾚｺｰﾄﾞｷｰ.
        05  ﾚｺｰﾄﾞｷｰ01  PICTURE  X(01).
        05  FILLER      PICTURE  X(04).
    03  ｷｰｻｲｽﾞ03   REDEFINES  ﾚｺｰﾄﾞｷｰ.
        05  ﾚｺｰﾄﾞｷｰ03  PICTURE  X(03).
        05  FILLER      PICTURE  X(02).
          :

PROCEDURE  DIVISION.
AAA  SECTION.
      :
    OPEN  INPUT  ﾌｧｲﾙ名.
    MOVE  "B"  TO  ﾚｺｰﾄﾞｷｰ                          ①ﾚｺｰﾄﾞｷｰB????に位置
    START  ﾌｧｲﾙ名  KEY  IS  EQUAL  TO  ﾚｺｰﾄﾞｷｰ01    づける
        INVALID  KEY
            DISPLAY  "グループ（B????）は存在しません"
    END-START
      :
    MOVE  "CC2"  TO  ﾚｺｰﾄﾞｷｰ                         ②ﾚｺｰﾄﾞｷｰCC2??に位置
    START  ﾌｧｲﾙ名  KEY  IS  EQUAL  TO  ﾚｺｰﾄﾞｷｰ03    づける
        INVALID  KEY
            DISPLAY  "グループ（CC2??）は存在しません"
    END-START
      :
```

REDEFINESでﾚｺｰﾄﾞｷｰを再定義して必要なキーサイズのﾃﾞｰﾀ項目を定義する

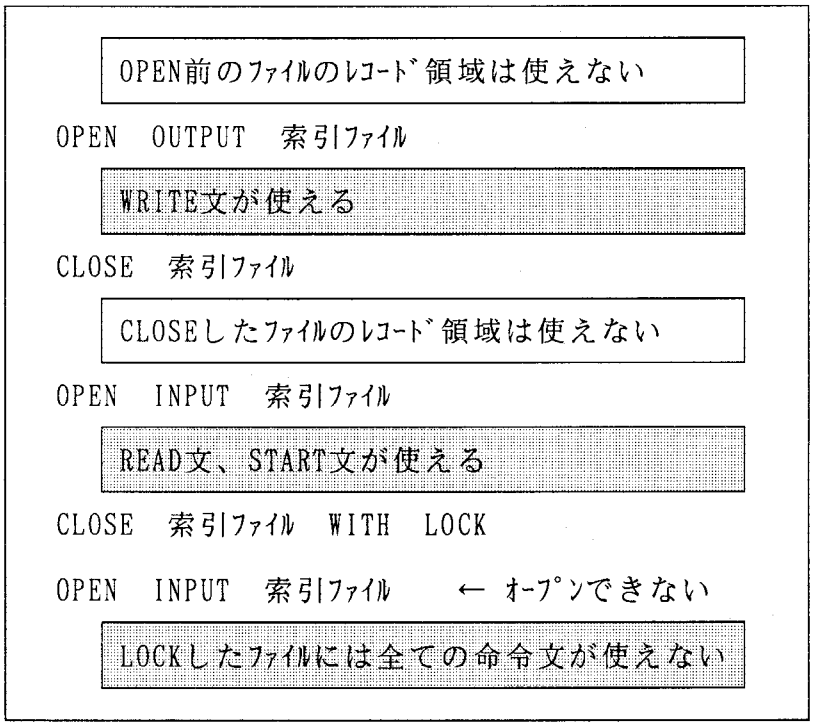
７．１．１０　ＣＬＯＳＥ

ＣＬＯＳＥ（閉じる）文は、ファイルの処理を終了させる。

```
CLOSE   {ﾌｧｲﾙ名1   [WITH  LOCK] } ･･･
```

必要ならば施錠（LOCK）して、同一プログラム（実行単位）内で再びＯＰＥＮすることを不可能にすることもできる。

ＣＬＯＳＥが成功すると、そのファイル名のレコード領域のデータ項目は参照できなくなる。



７．１．１１　索引ファイルの作成

索引ファイルの作成（生成）は、ＷＲＩＴＥ文でレコードをファイルに追加する。
作成の手順は、次の通り。

①ファイルをＯＰＥＮ　ＯＵＴＰＵＴ、または、Ｉ－Ｏでオープンする。
②レコードをＷＲＩＴＥで追加する。
③ファイルをＣＬＯＳＥする。

また、呼出し法でＷＲＩＴＥ文を分類すると、次のとおり。

①順呼出し法（ACCESS　MODE　IS　SEQUENTIAL）
②乱呼出し法、動的呼出し法（ACCESS　MODE　IS　RANDOM／DYNAMIC）

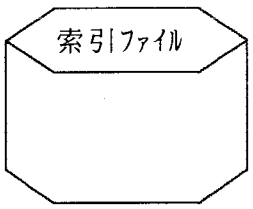
索引ファイルのＯＰＥＮモードと入出力文の組合せは、次のとおり。

呼出し法	入出力文	ＯＰＥＮモード			
		INPUT	OUTPUT	I-O	EXTEND
順呼出し	READ	○	－	○	－
	WRITE	－	○	－	○
	REWRITE	－	－	○	－
	START	○	－	○	－
	DELETE	－	－	○	－
乱呼出し	READ	○	－	○	－
	WRITE	－	○	○	－
	REWRITE	－	－	○	－
	START	－	－	－	－
	DELETE	－	－	○	－
動的呼出し	READ	○	－	○	－
	WRITE	－	○	○	－
	REWRITE	－	－	○	－
	START	○	－	○	－
	DELETE	－	－	○	－

通常、索引ファイルの最初の生成は、順呼出し法でレコードを作成する。そして、生成されたファイルへのレコードの追加は乱呼出し法、動的呼出し法で行う。

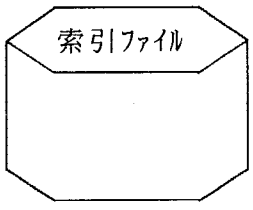
①ステップ１

```
SELECT  ﾌｧｲﾙ名  ASSIGN  TO  ～
    ORGANIZATION  IS  INDEXED
    ACCESS  MODE IS  SEQUENTIAL

OPEN  OUTPUT  ﾌｧｲﾙ名
WRITE  ﾚｺｰﾄﾞ名  INVALID  KEY  ～
```



②ステップ２

```
SELECT  ﾌｧｲﾙ名  ASSIGN  TO  ～
    ORGANIZATION  IS  INDEXED
    ACCESS  MODE IS  RANDOM

OPEN  I-O     ﾌｧｲﾙ名
WRITE  ﾚｺｰﾄﾞ名  INVALID  KEY  ～
```



７．１．１２　索引ファイルの読み出し

索引ファイルでは、読み出すレコードの位置を示す概念上のポインターとして「ファイル位置指示子（file position indicator）」がある。
ファイル位置指示子は、ＯＰＥＮ、ＲＥＡＤ、ＳＴＡＲＴで設定され、ＣＬＯＳＥで不定になる。また、ＷＲＩＴＥ、ＲＥＷＲＩＴＥ、ＤＥＬＥＴＥでは影響されない。
ファイル位置指示子の概念は、出力モード（OUTPUT）や拡張モード（EXTEND）で開いたファイルでは意味をもたない。

命　　令	ファイル位置指示子の内容
ＯＰＥＮ	ﾌｧｲﾙ内の最初のﾚｺｰﾄﾞを指す。
ＲＥＡＤ	ﾌｧｲﾙ位置指示子が指すﾚｺｰﾄﾞを取り出し、ﾌｧｲﾙ位置指示子は次のﾚｺｰﾄﾞを指す。
ＳＴＡＲＴ	ﾌｧｲﾙ位置指示子は指定されたﾚｺｰﾄﾞｷｰ位置のﾚｺｰﾄﾞを指す。
ＣＬＯＳＥ	ﾌｧｲﾙ位置指示子は不定になる。
ＷＲＩＴＥ ＲＥＷＲＩＴＥ ＤＥＬＥＴＥ	ファイル位置指示子は変化しない。

索引ファイルのレコードキーには、主レコードキーと副レコードキーがあり、ファイル位置指示子は、命令文で指定されたレコードキー（主または副）により設定される。
ＯＰＥＮ文のファイル位置指示子は主レコードキーが使用され、最初のレコードは主レコードキーの最小の値が指すレコードである。

７．１．１３　索引ファイルの追加と更新

レコードの更新は、ＲＥＷＲＩＴＥを使用して主レコードキーを除いたレコードの内容を書き換える。
副レコードキーの値は変更できる。また、副レコードキーが同じ値を許される（WITH DUPLICATES）場合で副レコードキーと同じ値が既に存在する時、同じ値の副レコードキーの最後に位置づけられる。

７．２　相対ファイル

相対ファイル（relative　file）は、相対レコード番号（relative　record　number）によりアクセスする大記憶ファイルである。

```
SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語
    ORGANIZATION  IS   RELATIVE
```

レコードをアクセスする相対レコード番号は符号なし整数項目として、そのファイルのレコード領域以外に定義する。

```
    RELATIVE  KEY  IS  ﾃﾞｰﾀ名    ←相対ﾚｺｰﾄﾞ番号を指定するデータ項目
```

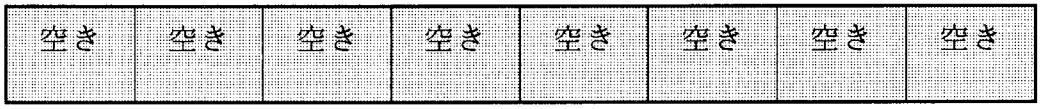
最初のレコードの相対レコード番号は１であり、続くレコードは２、３、・・となる。

```
SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語1
    ORGANIZATION  IS  RELATIVE
    RELATIVE  KEY  IS  相対ﾚｺｰﾄﾞ番号.

FILE  SECTION.
FD  ﾌｧｲﾙ名  ～

01  相対ﾚｺｰﾄﾞ.                                          ﾚｺｰﾄﾞ領域以外に定
    03  口座番号.                                       義する
        05  口座       PICTURE  9(8)  USAGE  BINARY.
    03  預金額         PICTURE  S9(9).
    03  名前           PICTURE  X(20).
    03  年齢           PICTURE  9(3).
    03  住所           PICTURE  X(50).
          :
WORKING-STORAGE  SECTION.

01  相対ﾚｺｰﾄﾞ番号      PICTURE  9(8)  USAGE  BINARY.   ←符号無し
                                                        整数項目
```

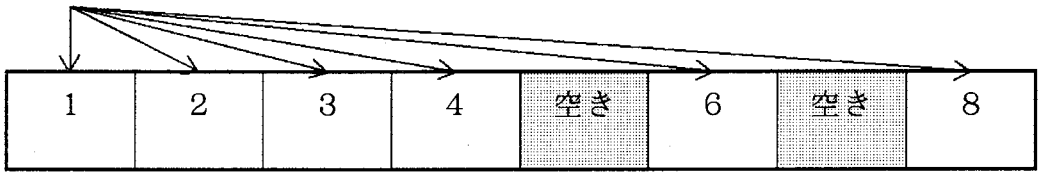
７．２．１　相対ファイルの概念

相対ファイルは、大記憶装置上に作成することができ、相対レコード番号により識別されるレコードで編成される。
相対ファイル中の各レコード領域は、相対レコード番号をもち、レコードが書かれているか否かにかかわらずレコード領域が確保される。

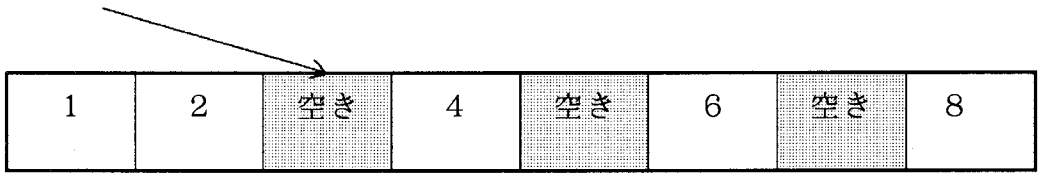
OPEN　OUTPUT　相対ﾌｧｲﾙ



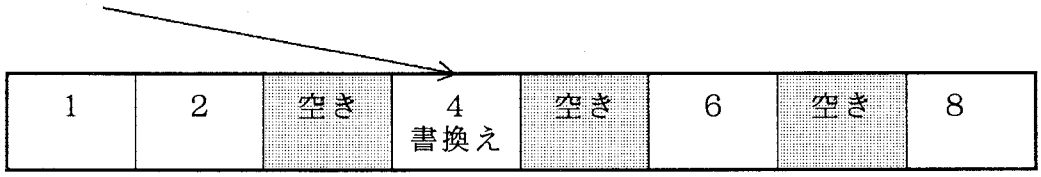
WRITE　ﾚｺｰﾄﾞ



OPEN I-O 相対ﾌｧｲﾙ
DELETE　ﾌｧｲﾙ名



OPEN I-O 相対ﾌｧｲﾙ
REWRITE　ﾚｺｰﾄﾞ



７．２．２　相対ファイルの定義

ＣＯＢＯＬプログラムにおける相対ファイル定義の全体像を解説する。

相対ファイル定義の概要

```
IDENTIFICATION  DIVISION.
PROGRAM-ID.     RELATIVE-F.

ENVIRONMENT  DIVISION.

INPUT-OUTPUT  SECTION.
FILE-CONTROL.

    SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語1                          ①
            ORGANIZATION  IS  RELATIVE                            ②
            ACCESS  MODE  IS  呼出し法                            ③
            RELATIVE  KEY  IS  相対ﾚｺｰﾄﾞ番号                      ④
            FILE  STATUS  IS  ﾌｧｲﾙｽﾃｰﾀｽ.                          ⑤

DATA  DIVISION.
FILE  SECTION.

FD    ﾌｧｲﾙ名
      BLOCK  CONTAINS  整数2  RECORDS                             ⑥
      RECORD  CONTAINS  整数3  CHARACTERS.
01    ﾚｺｰﾄﾞ記述項.                                                ⑦
      03  ﾃﾞｰﾀ名
          :
          :

WORKING-STORAGE  SECTION.
01    ﾌｧｲﾙｽﾃｰﾀｽ          PICTURE  X(2).

01    相対ﾚｺｰﾄﾞ番号      PICTURE  9(桁数).                       ⑧

PROCEDURE  DIVISION.

      OPEN  入出力ﾓｰﾄﾞ  ﾌｧｲﾙ名                                     ⑨

      WRITE  ﾚｺｰﾄﾞ名  INVALID  KEY  ～
                                                                  ⑩
      READ  ﾌｧｲﾙ名  AT  END／INVALID  KEY  ～

      REWRITE  ﾚｺｰﾄﾞ名  INVALID  KEY  ～

      START  ﾌｧｲﾙ名  KEY  IS  比較演算子  ﾃﾞｰﾀ名  INVALID  KEY  ～

      DELETE  ﾌｧｲﾙ名  INVALID  KEY  ～

      CLOSE  ﾌｧｲﾙ名.
```

相対ファイル定義の説明

①ファイル名
ＣＯＢＯＬプログラム上の論理ファイル名を定義する。
SELECT ﾌｧｲﾙ名 ASSIGN TO 作成者語

②ファイル編成
ファイル編成として相対ファイルを定義する。
SELECT ﾌｧｲﾙ名 ASSIGN TO 作成者語
ORGANIZATION IS RELATIVE

③呼出し方式
相対ファイルの呼出し方式を定義する。相対ファイルでは全ての方式をサポートしている。
ACCESS MODE IS SEQUENTIAL － 順呼出し法
ACCESS MODE IS RANDOM － 乱呼出し法
ACCESS MODE IS DYNAMIC － 動的呼出し法

④相対レコード番号
相対ファイルは相対レコード番号の値をキーとしてレコードを取り出す。その相対レコード番号の項目名を定義する。
相対レコード番号の値は、１から始まり、２、３、・・と続く。

⑤ファイルステータス
ＣＯＢＯＬでの入出力の実行結果（入出力状態）を返す領域名（2ﾊﾞｲﾄ）を定義する。

⑥ブロック、レコード
レコードとブロックの構成を定義する。

⑦レコード記述
入出力命令で外部ファイルとのレコードのやりとりを行う領域と、そのデータ構造を定義する。

⑧相対レコード番号
相対ファイルのキー項目として相対レコード番号を「数字項目」で定義する。
数字項目の桁数は、１～相対レコード番号の最大値の範囲を表せる桁数を指定する。

⑨入出力モード
相対ファイルの処理モードを指定する。
INPUT － 入力ファイル
OUTPUT － 出力ファイル
I-O － 入出力両用ファイル
EXTEND － 拡張ファイル

⑩入出力命令
入出力の動作を指定する。
OPEN文を最初に実行して、その後、入出力モードに対応した文を実行する。
最後はCLOSE文を実行する。

７．２．３　ＳＥＬＥＣＴ

相対ファイルに関する属性を定義する。

```
SELECT  [OPTIONAL] ﾌｧｲﾙ名1  ASSIGN  TO  作成者語1／定数1
        [RESERVE  整数1  AREA／AREAS]
        [ORGNIZATION  IS]  RELATIVE
        [ACCESS  MODE  IS
           ┌SEQUENTIAL   [RELATIVE  KEY  IS  ﾃﾞｰﾀ名1] ┐
           ┤RANDOM        RELATIVE  KEY  IS  ﾃﾞｰﾀ名1  ├]
           └DYNAMIC       RELATIVE  KEY  IS  ﾃﾞｰﾀ名1  ┘
        [FILE  STATUS  IS  ﾌｧｲﾙｽﾃｰﾀｽ] .
```

ＳＥＬＥＣＴ句は相対ファイルの呼出し法と、相対レコード番号のデータ項目名を定義する。相対ファイルをＳＴＡＲＴ文で使用する場合には、RELATIVE　KEY指定を書かなければならない。

呼出し法（ACCESS　MODE）には３種類あり、索引ファイルでは全ての呼出し法が使える。

①順呼出し（SEQUENTIAL）
　相対レコード番号の値が昇順になるように呼び出される。

②乱呼出し（RANDOM）
　相対レコード番号の値で呼び出すレコードが決まる。

③動的呼出し（DYNAMIC）
　レコードは乱呼出しと順呼出しを組み合わせてアクセスできる。

相対レコード番号は、符号なし整数項目として、そのファイルのレコード領域外に定義してデータ管理システム（ファイルシステム）との間での連絡に使用する。
最初のレコードの相対レコード番号は１であり、続くレコードは２、３、・・となる。

相対レコード番号は、「Ｐ」を含まない符号なし整数項目で、そのファイルのレコード記述項以外に定義する。

```
WORKING-STORAGE  SECTION.

01  相対ﾚｺｰﾄﾞ番号  PICTURE  9(桁数)  USAGE  DISPLAY/BINARY/PACKED-DECIMAL
                            ↑         ↑
                            │         └── BINARYが望ましい
                            └── S、V、Pを含まない（符号なし整数）
```

相対ファイルのファイルステータス

上位桁	下位桁	意味	
0	0	成功	情報なし
0	4		ﾚｺｰﾄﾞ長がﾌｧｲﾙ属性と合わない
1	0	ﾌｧｲﾙの終了	順呼出しのREADでﾌｧｲﾙの終了になった
1	4		順呼出しのREADで相対ﾚｺｰﾄﾞ番号がﾌｧｲﾙの範囲外
2	2	無効ｷｰ	重複ｷｰ（相対ﾚｺｰﾄﾞ番号と同じ値が既に存在）
2	3		ﾌｧｲﾙに存在しないﾚｺｰﾄﾞを呼び出そうとした
2	4		ﾌｧｲﾙの区域外に書き出そうとした
3	0	永続ｴﾗｰ	情報なし
3	5		OPEN　I-O/INPUTでﾌｧｲﾙが存在しない
3	7		OPENﾓｰﾄﾞと物理ﾌｧｲﾙの属性が一致しない
3	8		CLOSE　WITH　LOCKのﾌｧｲﾙを再OPENした
3	9		物理ﾌｧｲﾙ属性とﾌｧｲﾙ定義との間で矛盾を検出
4	1	論理ｴﾗｰ	OPEN済みﾌｧｲﾙに再OPENを実行した
4	2		OPENされていないﾌｧｲﾙにCLOSEを実行した
4	3		順呼出しでDELETE/REWRITEの前がREADでない
4	4		区域外書き出し
4	6		終了したﾌｧｲﾙにREADを実行した
4	7		OUTPUTﾓｰﾄﾞのﾌｧｲﾙにREADを実行した
4	8		INPUTﾓｰﾄﾞのﾌｧｲﾙにWRITEを実行した
4	9		I-Oﾓｰﾄﾞ以外のﾌｧｲﾙにDELETE/REWRITEを実行した
9	x	その他ｴﾗｰ	作成者が定義する

７．２．４　ＯＰＥＮ

ファイルの処理を行うための準備をする。

```
OPEN { INPUT    {ファイル名1} ･･･ }
     { OUTPUT   {ファイル名2} ･･･ }  ･･･
     { I-O      {ファイル名3} ･･･ }
     { EXTEND   {ファイル名4} ･･･ }
```

相対ファイルのＯＰＥＮモードと入出力文の組合せは、次のとおり。

呼出し法	入出力文	ＯＰＥＮモード			
		INPUT	OUTPUT	I-O	EXTEND
順呼出し	READ	○	－	○	－
	WRITE	－	○	－	○
	REWRITE	－	－	○	－
	START	○	－	○	－
	DELETE	－	－	○	－
乱呼出し	READ	○	－	○	－
	WRITE	－	○	○	－
	REWRITE	－	－	○	－
	START	－	－	－	－
	DELETE	－	－	○	－
動的呼出し	READ	○	－	○	－
	WRITE	－	○	○	－
	REWRITE	－	－	○	－
	START	○	－	○	－
	DELETE	－	－	○	－

一つのファイルを同じ実行単位で、ＩＮＰＵＴ指定、ＯＵＴＰＵＴ指定、ＥＸＴＥＮＤ指定、Ｉ－Ｏ指定を書いて何回も開くことができる。ただし、２回目以降のＯＰＥＮを実行するためにはＬＯＣＫ指定のないＣＬＯＳＥを実行しておくこと。

```
PROCEDURE  DIVISION.

    OPEN  OUTPUT  相対ﾌｧｲﾙ
    WRITE 相対ﾚｺｰﾄﾞ①
    WRITE 相対ﾚｺｰﾄﾞ②
    CLOSE  相対ﾌｧｲﾙ

    OPEN OUTPUT  相対ﾌｧｲﾙ          再度OUTPUTでｵｰﾌﾟﾝしたので相対ﾚｺｰﾄﾞ①と②は消えてしまう。I-Oでｵｰﾌﾟﾝすれば残る。
    WRITE 相対ﾚｺｰﾄﾞ③
    WRITE 相対ﾚｺｰﾄﾞ④
    CLOSE  相対ﾌｧｲﾙ

    OPEN  I-O  相対ﾌｧｲﾙ            ﾌｧｲﾙ中には相対ﾚｺｰﾄﾞの③と④が存在する

    CLOSE  相対ﾌｧｲﾙ  WITH  LOCK

    OPEN  INPUT  相対ﾌｧｲﾙ          ←ｵｰﾌﾟﾝできない
```

7．2．5　ＷＲＩＴＥ

出力ファイル（OUTPUT）、入出力両用ファイル（I-O）にレコードを書き出す。

```
WRITE  ﾚｺｰﾄﾞ名1  [FROM  一意名1]
       [INVALID  KEY  無条件文1]
       [NOT  INVALID  KEY  無条件文2]
 [END-WRITE]
```

順呼出し法（ACCESS　MODE　IS　SEQUENTIAL）のＷＲＩＴＥでは、最初のレコードの相対レコード番号が１になり、次に続くレコードの相対レコード番号が２、３、・・になる。
相対レコード番号項目を定義するとＷＲＩＴＥで書き出された相対レコード番号の値が返される。

```
       SELECT  相対ﾌｧｲﾙ  ASSIGN  TO  作成者語1
               ORGANIZATION  IS  RELATIVE
               ACCESS  MODE  IS  SEQUENTIAL  RELATIVE  KEY  相対ｷｰ.

DATA  DIVISION.
FILE  SECTION.
FD  相対ﾌｧｲﾙ  ～.
01  相対ﾚｺｰﾄﾞ.
    03  ﾃﾞｰﾀ    ～

WORKING-STORAGE  SECTION.
01  相対ｷｰ    PICTURE  9(5).  <----------+
PROCEDURE  DIVISION.                      |
                                          |
    OPEN  OUTPUT  相対ﾌｧｲﾙ                |
                                          |
  WRITE  相対ﾚｺｰﾄﾞ  INVALID  KEY  ～  ;    | 相対ﾚｺｰﾄﾞ番号が１から書き出され
  WRITE  相対ﾚｺｰﾄﾞ  INVALID  KEY  ～      | その番号がｼｽﾃﾑにより設定される。
```

乱呼出し法（RANDOM）や動的呼出し法(DYNAMIC）では、相対レコード番号に任意の値を指定して書き出せる。
また、相対レコード番号の値と等しいレコードが既にファイル中に書かれていると無効キー条件になる。

```
       SELECT  相対ﾌｧｲﾙ  ASSIGN  TO  作成者語1
               ORGANIZATION  IS  RELATIVE
               ACCESS  MODE  IS  RANDOM      RELATIVE  KEY  相対ｷｰ.

DATA  DIVISION.
FILE  SECTION.
FD  相対ﾌｧｲﾙ  ～.
01  相対ﾚｺｰﾄﾞ.
    03  ﾃﾞｰﾀ    ～

WORKING-STORAGE  SECTION.
01  相対ｷｰ    PICTURE  9(5).  ----------+
PROCEDURE  DIVISION.                     |
                                         v
    OPEN  OUTPUT  相対ﾌｧｲﾙ
  MOVE  相対ﾚｺｰﾄﾞ番号  TO  相対ｷｰ        | ﾕｰｻﾞが指定した相対ﾚｺｰﾄﾞ番号の
  WRITE  相対ﾚｺｰﾄﾞ  INVALID  KEY  ～    | 位置に書き出される。
```

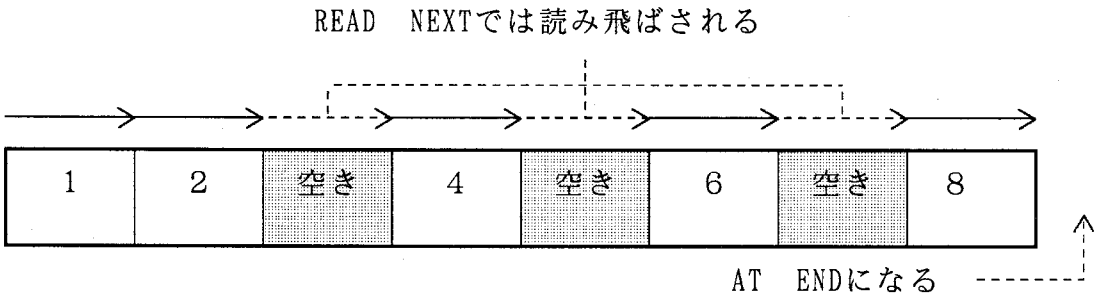
７．２．６　ＲＥＡＤ

順呼出し法では、次のレコードを取り出す。乱呼出し法では、相対レコード番号の値で指定されたレコードを取り出す。

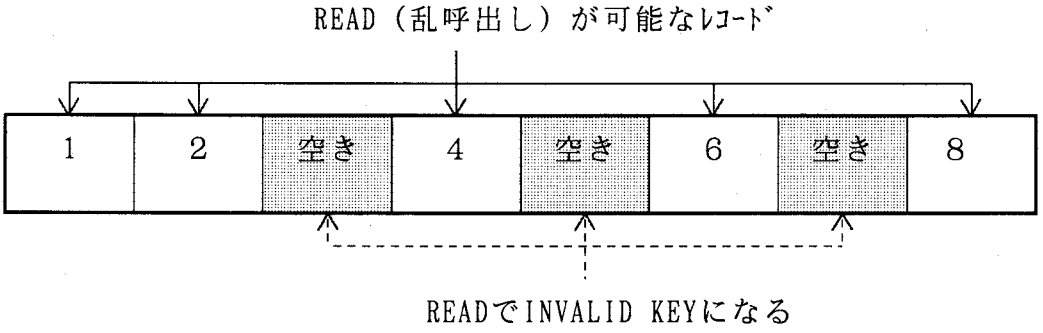
```
READ  ﾌｧｲﾙ名1  [NEXT]  RECORD  [INTO  一意名1]
      [AT  END  無条件文1]
      [NOT  AT  END  無条件文2]
 [END-READ]

READ  ﾌｧｲﾙ名1  RECORD  [INTO  一意名1]
      [INVALID  KEY  無条件文1]
      [NOT  INVALID  KEY  無条件文2]
 [END-READ]
```

順呼出し法（ACCESS　MODE　IS　SEQUENTIAL）では、「READ　ﾌｧｲﾙ名　AT　END　～」を使用する。RELATIVE　KEY句を指定するとREADの実行が成功すると読み出されたレコードの相対レコード番号が返される。

動的呼出し法（DYNAMIC）で順呼出しの場合は、「READ　ﾌｧｲﾙ名 NEXT　AT　END　～」を使用する。空きレコード領域は読み飛ばされる。



乱呼出し法（RANDOM）、動的呼出し法（DYNAMIC）で乱呼出しの場合は、「READ　ﾌｧｲﾙ名 INVALID　KEY　～」を使用する。空きレコード領域の相対レコード番号を指定するとINVALID　KEYになる。

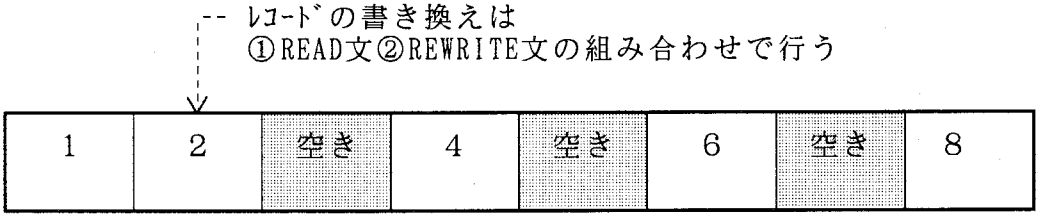


７．２．７　ＲＥＷＲＩＴＥ

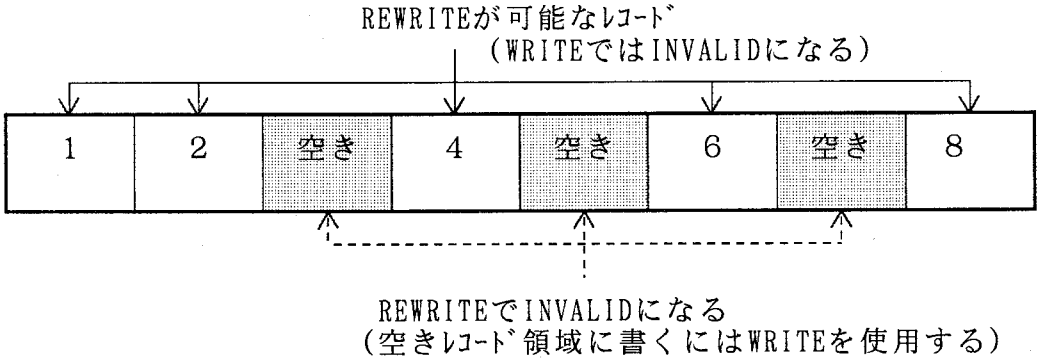
入出力両用モード（I-O）でオープンされたファイルのレコードを書き換える。

```
REWRITE  ﾚｺｰﾄﾞ名1   [FROM  一意名1]
         [INVALID  KEY  無条件文1]
         [NOT  INVALID  KEY  無条件文2]
 [END-REWRITE]
```

順呼出し法（ACCESS　MODE　IS　SEQUENTIAL）では、最後にＲＥＡＤで読み込んだレコードが書き換えられる。



乱呼出し法（RANDOM）、動的呼出し法（DYNAMIC）では、相対レコード番号を指定してファイル中のレコードを直接書き換えられる。空きレコード領域にはＷＲＩＴＥを使用してレコードを書く。



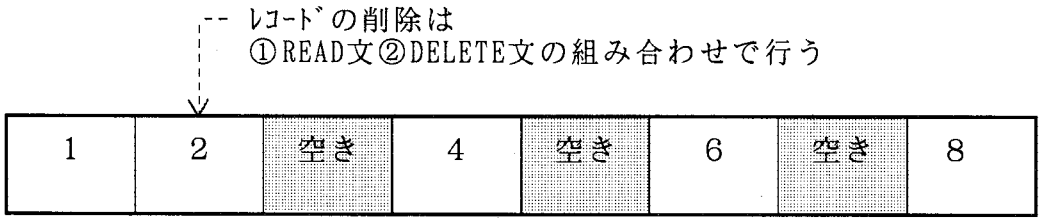
７．２．８　ＤＥＬＥＴＥ

入出力両用モード（I-O）でオープンされたファイルのレコードを論理的に取り除く。

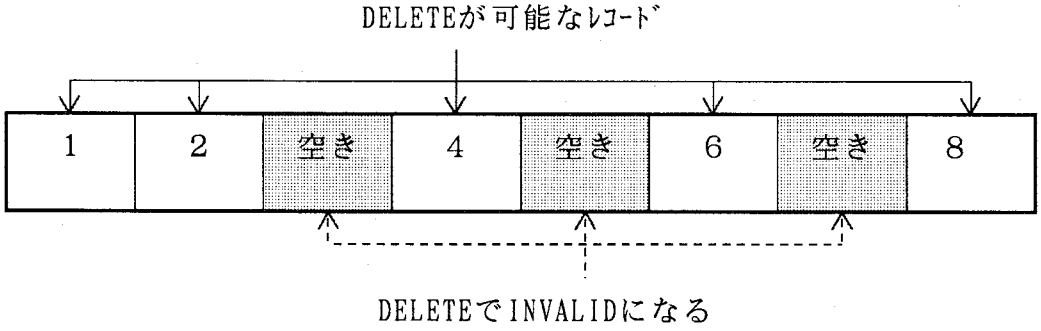
```
DELETE  ﾚｺｰﾄﾞ名1  RECORD
        [INVALID  KEY  無条件文1]
        [NOT  INVALID  KEY  無条件文2]
 [END-DELETE]
```

絨完穐鰺彊

順呼出し法（ACCESS　MODE　IS　SEQUENTIAL）では、直前のＲＥＡＤで読み込んだレコードが取り除かれる。



乱呼出し法（RANDOM）、動的呼出し法（DYNAMIC）では、相対レコード番号を指定してファイル中のレコードを直接取り除くことができる。



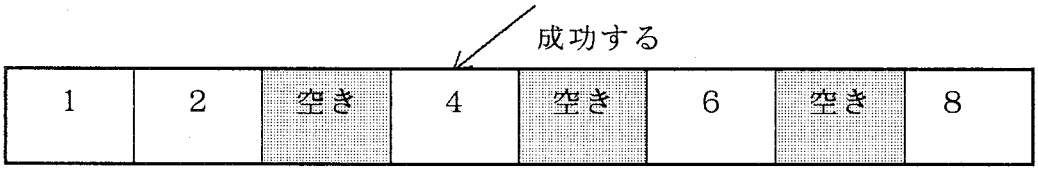
７．２．９　ＳＴＡＲＴ

順呼出し法、動的呼出し法で相対レコード番号の値とその比較条件をＫＥＹ句で指定して、ファイル中のレコードに位置づける。
レコードの取り出しには、ＲＥＡＤ　ＮＥＸＴを使用する。

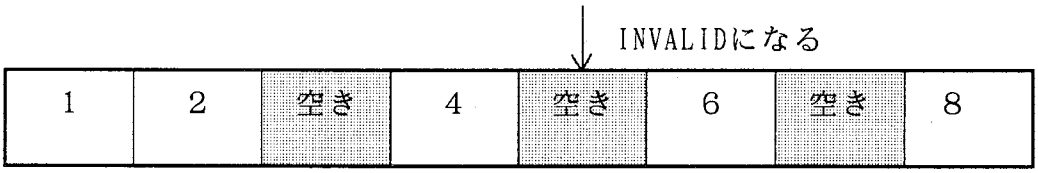
```
START  ﾌｧｲﾙ名1  [KEY { IS  EQUAL  TO                          }
                   { IS  =                                    }
                   { IS  GREATER  THAN                        }
                   { IS  >                                    }
                   {                                          } ﾃﾞｰﾀ名1]
                   { IS  NOT  LESS  THAN                      }
                   { IS  NOT  <                               }
                   { IS  GREATER  THAN  OR  EQUAL  TO         }
                   { IS  >=                                   }
       [INVALID  KEY  無条件文1]
       [NOT  INVALID  KEY  無条件文2]
 [END-START]
```

ＫＥＹ句のデータ名１は相対レコード番号として定義したデータ項目を指定する。

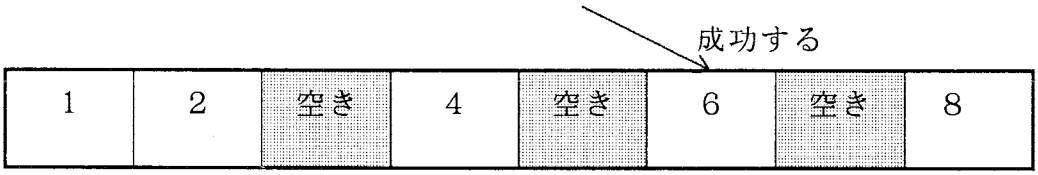
```
MOVE  4  TO  相対ﾚｺｰﾄﾞ番号
START  相対ﾌｧｲﾙ  KEY  IS  =  相対ﾚｺｰﾄﾞ番号  INVALID  ～
```




```
MOVE  5  TO  相対ﾚｺｰﾄﾞ番号
START  相対ﾌｧｲﾙ  KEY  IS  =  相対ﾚｺｰﾄﾞ番号  INVALID  ～
```




```
MOVE  5  TO  相対ﾚｺｰﾄﾞ番号
START  相対ﾌｧｲﾙ  KEY  IS  >=  相対ﾚｺｰﾄﾞ番号  INVALID  ～
```

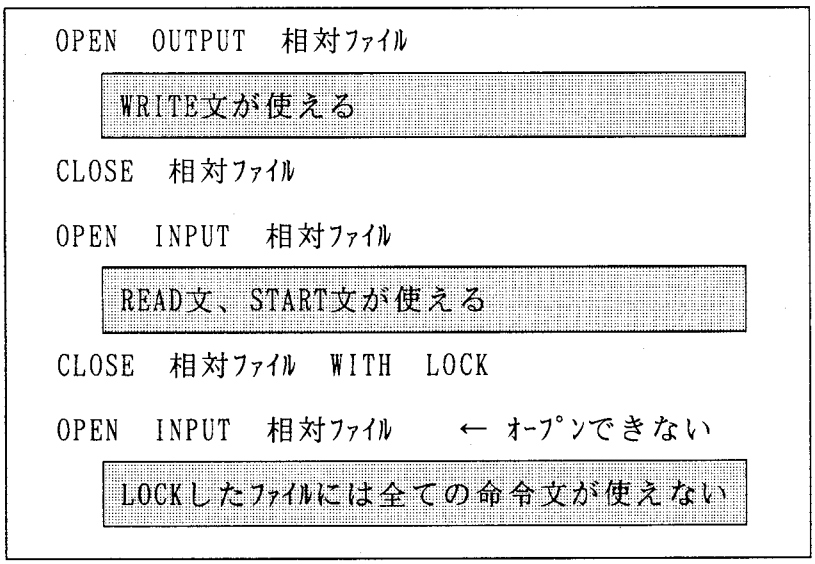


７．２．１０　ＣＬＯＳＥ

ファイルの処理を終了させる。

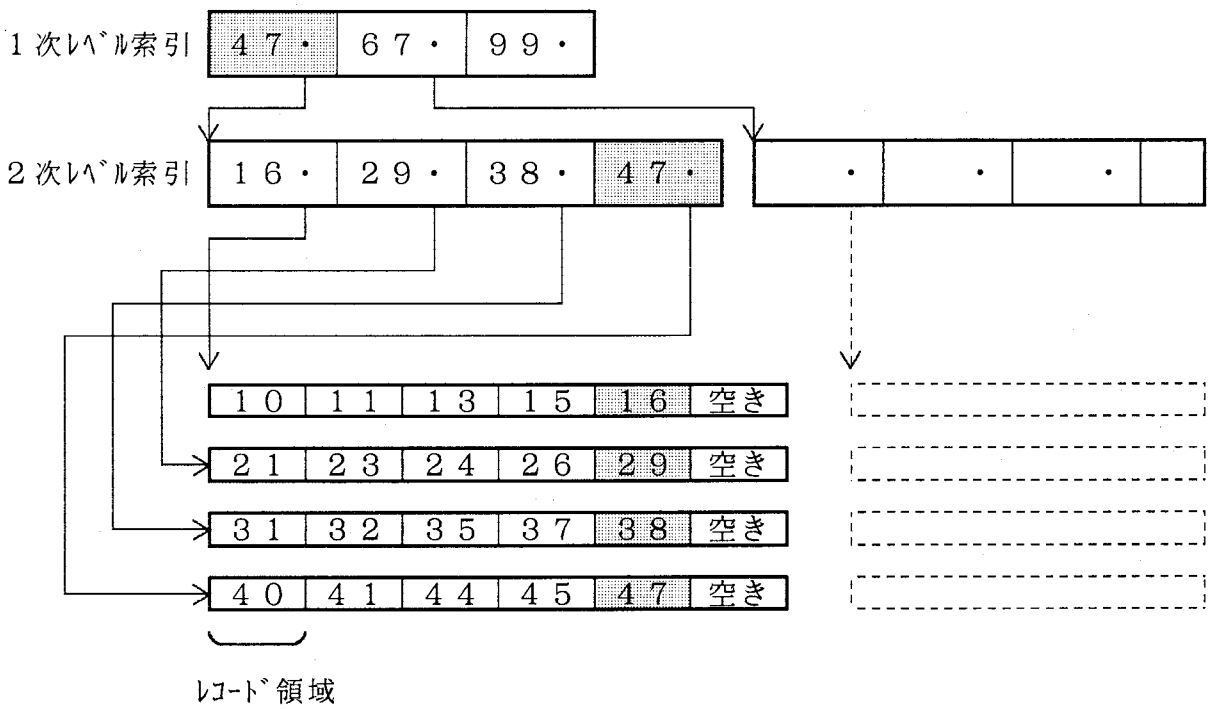
```
CLOSE   {ﾌｧｲﾙ名1   [WITH  LOCK] } ･･･
```

必要ならば施錠（LOCK）して、同一プログラム（実行単位）内で再びＯＰＥＮすることを不可能にすることもできる。



指導上の留意点

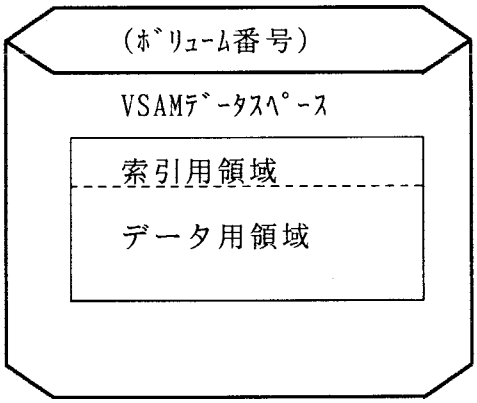
◎索引ファイルの構造
汎用コンピュータでは、ＶＳＡＭファイルのＫＳＤＳ（Key Sequenced Data Set）で構成され、高速でアクセスするために次のような構造になっている。



◎索引ファイルの定義
ＶＳＡＭを使用して索引ファイルを生成する場合は、前もってＶＳＡＭユーティリティ（ＡＭＳなど）を使用して索引ファイルの領域確保とキーの位置等の情報を定義する。
定義の例は、次のとおり。

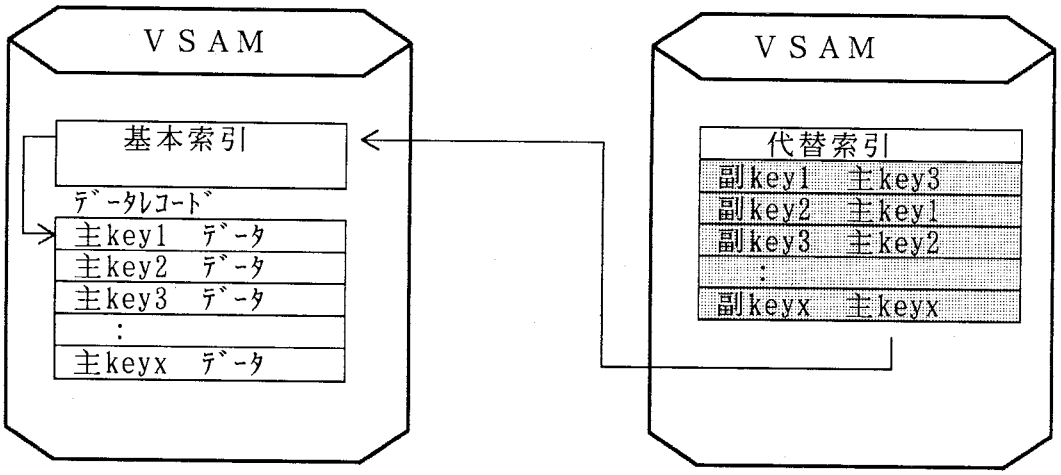
```
DEFINE  CLUSTER  (NAME(名前)  -
                  VOLUMES(ﾎﾞﾘｭｰﾑ番号)  -
                  INDEXED  -
                  KEYS(長さ,位置)  -
                  RECORDS(ﾚｺｰﾄﾞ件数)  -
                  RECORDSIZE(平均ｻｲｽﾞ,最大ｻｲｽﾞ))
```

↓



索引ファイル毎に「DEFINE　CLUSTER」が必要であるが、前もってＶＳＡＭカタログやＶＳＡＭデータ・スペースを定義する必要がある。これらはシステム管理者により特別に定義、管理されることが多い。

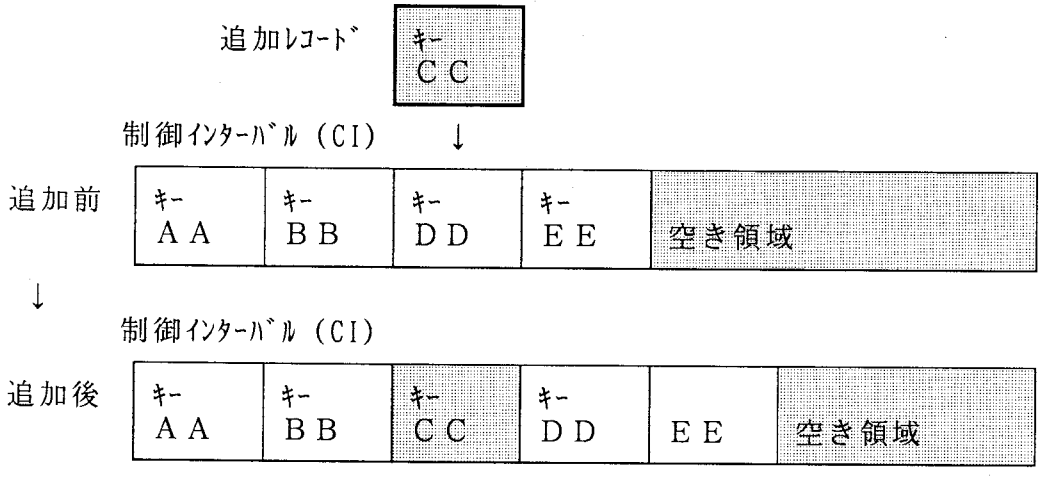
◎副レコードキーの作成
ＣＯＢＯＬの副レコードキーは、ＶＳＡＭでは代替索引と呼ばれる。また、主レコードキーは基本索引と呼ばれる。



代替索引は、索引ファイルの生成後にＶＳＡＭユーティリティで次のように作成する。
(1)「DEFINE　ALTERNATEINDEX」で代替索引のキーになるレコード上の副レコードキーの位置等を定義して代替索引領域を確保する。
(2)「BUILDINDEX」で基本となる索引ファイルをもとに代替索引を作成する。
(3)「DEFINE　PATH」で基本索引と代替索引を結び付ける。

◎索引ファイルのレコードの挿入
索引ファイルの更新処理で、途中にレコードを挿入することが多い索引ファイルの処理では索引ファイルの定義時に、空き領域（ﾌﾘｰｽﾍﾟｰｽ）を効果的に確保するとよい。
空き領域はレコードの追加が頻繁に発生する処理に有効であるが、検索のみに使用する処理では空き領域のディスクスペースが無駄になる。
空き領域の定義の例は、次のとおり。

```
DEFINE  CLUSTER  (NAME(名前)  -
                  VOLUMES(ﾎﾞﾘｭｰﾑ番号)  -
                  INDEXED  -
                  KEYS(長さ,位置)  -
                  RECORDS(ﾚｺｰﾄﾞ件数)  -
                  RECORDSIZE(平均ｻｲｽﾞ,最大ｻｲｽﾞ) -
                  FREESPACE(n)  )←制御ｲﾝﾀｰﾊﾞﾙ中の空き領域のﾊﾟｰｾﾝﾄ
```



◎索引ファイルの再使用
索引ファイルは、「DEFINE　CLUSTER」後に、「OPEN　OUTPUT　索引ﾌｧｲﾙ名」と「WRITE」でレコードを生成して「CLOSE　索引ﾌｧｲﾙ名」で終わる。
その後、再度「OPEN　OUTPUT　索引ﾌｧｲﾙ」を実行するとエラーになることがある。

```
DEFINE CLUSTERの定義
        ↓①
OPEN  OUTPUT  索引ﾌｧｲﾙ名
WRITE  ﾚｺｰﾄﾞ
  :
CLOSE  索引ﾌｧｲﾙ
        ↓②
OPEN  OUTPUT  索引ﾌｧｲﾙ名      ←再作成は失敗する。
WRITE  ﾚｺｰﾄﾞ                   INPUT、I-Oのｵｰﾌﾟﾝならば成功する
  :
CLOSE  索引ﾌｧｲﾙ
```

再作成（OUTPUT）を許可するには、次のように指定する。

```
DEFINE  CLUSTER  (NAME(名前)  -
                  VOLUMES(ﾎﾞﾘｭｰﾑ番号)  -
                  INDEXED  -
                  KEYS(長さ,位置)  -
                  RECORDS(ﾚｺｰﾄﾞ件数)  -
                  RECORDSIZE(平均ｻｲｽﾞ,最大ｻｲｽﾞ) -
                  FREESPACE(n)  -
                  REUSE | NOREUSE  )  ←REUSE指定は再作成が可能になる
```

NOREUSEで再作成するには、索引ファイルを削除（DELETE）してから、再度、「DEFINE CLUSTER」を行ってから作成プログラムを実行する。
REUSEの場合は、作成プログラムの実行だけで再作成できる。

◎相対ファイルの定義

ＶＳＡＭを使用して相対ファイルを生成する場合は、前もってＶＳＡＭユーティリティ（ＡＭＳなど）を使用して相対ファイルのレコードサイズと領域確保（レコード件数）等の情報を定義する。

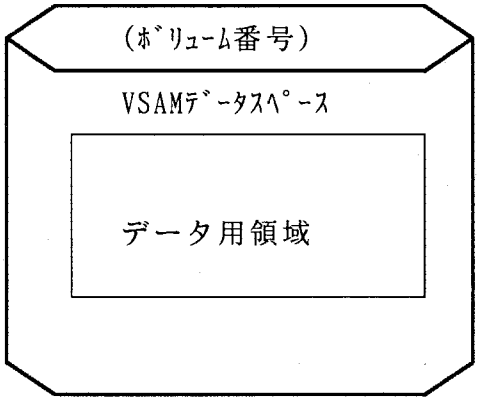
ＶＳＡＭデータ・セット編成はＲＲＤＳ（Relative Record Data Set）を使用する。

RECORDSIZEﾊﾟﾗﾒｰﾀの平均と最大サイズには同じ値を指定して固定長レコードにする。

定義の例は、次のとおり。

```
DEFINE  CLUSTER  (NAME(名前)  -
                  VOLUMES(ﾎﾞﾘｭｰﾑ番号)  -
                  NUMBERED  -
                  RECORDS(ﾚｺｰﾄﾞ件数)  -
                  RECORDSIZE(平均ｻｲｽﾞ,最大ｻｲｽﾞ))
```

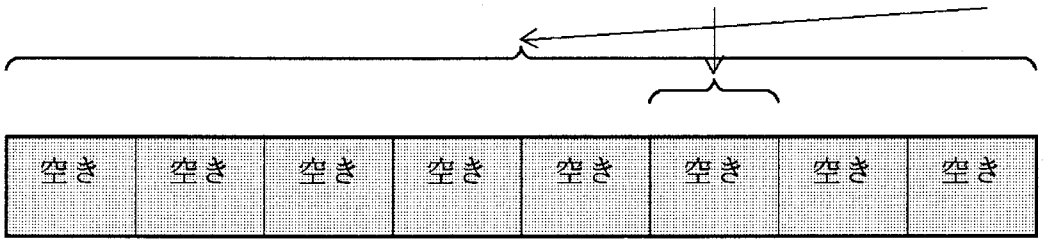
↓



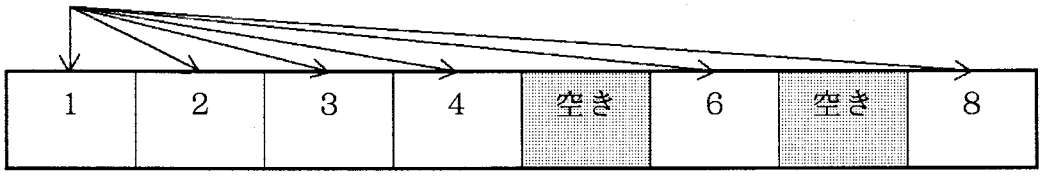
◎相対ファイルの構造

ＶＳＡＭでの相対ファイルの構造は、次のとおり。

DEFINE CLUSTER(NAME(名前) NUMBERED RECORDSIZE(ｻｲｽﾞ) RECORDS(件数) ・・)



WRITE ﾚｺｰﾄﾞ



DELETE ﾌｧｲﾙ名