# Gestion des fichiers

## 1. Organisation et accès supportés

Le langage COBOL permet la manipulation de fichiers d'**organisation**

- s**équentielle** (`SEQUENTIAL`)
- **séquentielle indexée** (`INDEXED`)
- **relative** (`RELATIVE`)

L'accès à un fichier organisé séquentiellement est nécessairement séquentiel. Les enregistrements sont atteints dans l'ordre physique où ils sont rencontrés.

Pour les autres organisations de fichiers, l'**accès** pourra être :

- **séquentiel**, soit dans l'ordre logique des enregistrements (`SEQUENTIAL`)
- **sélectif** (ou aléatoire) selon une clé (`RANDOM`)
- **mixte**, c'est à dire séquentiel et aléatoire, à l'intérieur d'un même programme (`DYNAMIC`), sans avoir à ré-ouvrir le fichier.

L'organisation d'un fichier est fixée lors de sa **création** ; le mode d'accès est précisé dans chaque programme manipulant le fichier en question.

## 2. Input-output section

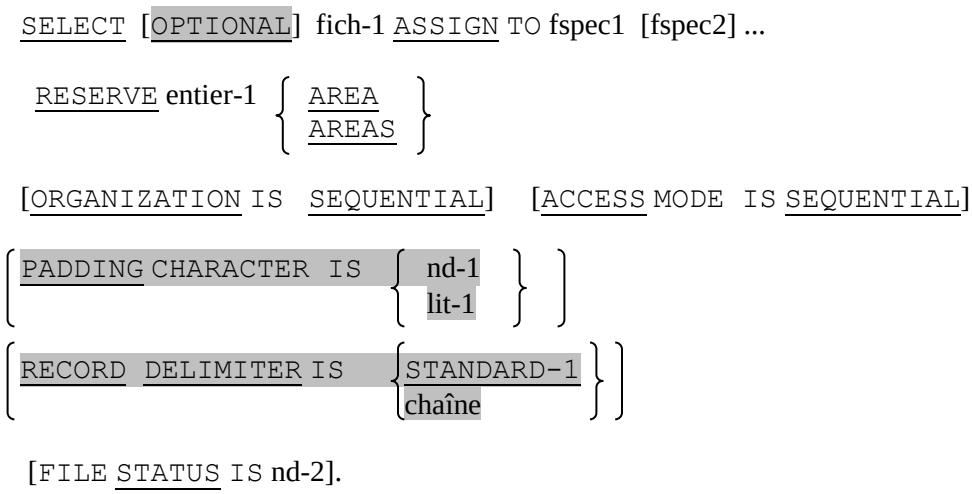
Cette section fait partie de l' `ENVIRONMENT DIVISION` et se compose :

- du paragraphe `FILE-CONTROL` comprenant une phrase `SELECT` par fichier utilisé ; cette phrase permet de préciser en particulier les paramètres organisation et accès pour ce fichier et de faire le lien avec le SGF par l'intermédiaire d'un mnémonique.
- du paragraphe `I-O-CONTROL` (facultatif) permettant de préciser les zones d'entrée-sortie (buffers) communes à plusieurs fichiers.

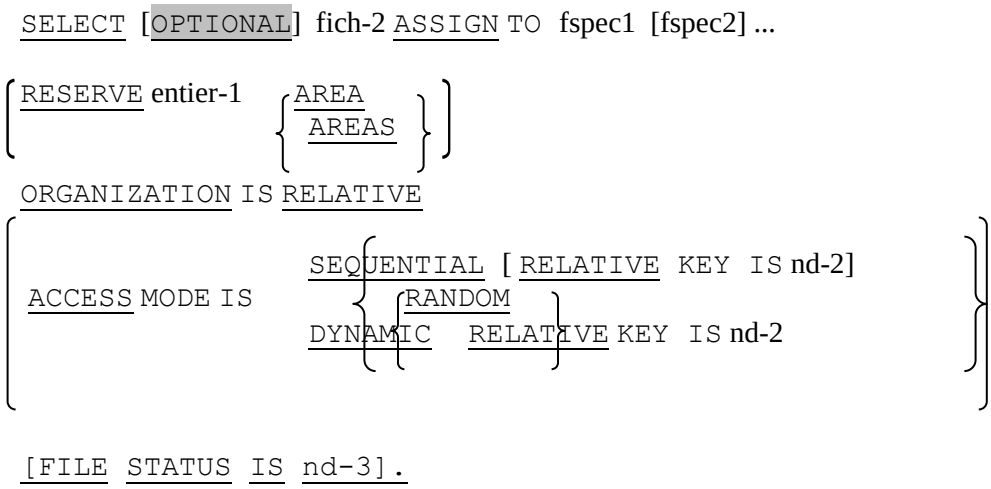
## 2.1. Le paragraphe FILE-CONTROL

### 2.1.1. Syntaxe générale

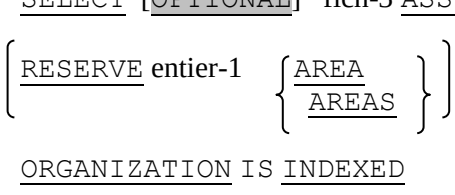
***Format 1 (pour fichiers d'organisation séquentielle)***




***Format 2 (pour fichiers d'organisation relative)***




***Format 3 (pour fichiers d'organisation séquentielle indexée)***



```
[ ACCESS MODE IS { SEQUENTIAL }                                    ]
                 { RANDOM     }
                 { DYNAMIC    }

RECORD KEY IS nd-4

[ALTERNATE RECORD KEY IS nd-5 [WITH DUPLICATES]]...

[FILE STATUS IS nd-3].
```

- Par **défaut**, organisation et accès sont considérés comme séquentiels.
- A chaque phrase SELECT doit correspondre un paragraphe FD.
- Les spécifications de fichiers permettent l'assignation d'un nom externe au nom de fichier interne, et sont spécifiques de l'implantation.
- OPTIONAL indique que le fichier peut être absent ; dans ce cas, la condition de fin-de-fichier est validée dès la première lecture.
- RESERVE spécifie, à titre documentaire, le nb de buffers d'E/S alloués au fichier.
- PADDING indique le caractère à utiliser pour compléter les blocs incomplets.
- RECORD DELIMITER traite des enregistrements de longueur variable.
- FILE STATUS définit la zone de WORKING-STORAGE SECTION accueillant le code de retour de l'opération d'E/S (cf ci-après).
- Les autres clauses sont définies par la suite.

### 2.1.3. Exemple de FILE-CONTROL

```
ENVIRONMENT DIVISION.
 *
 *
CONFIGURATION SECTION.
SOURCE-COMPUTER.xxx.
OBJECT-COMPUTER.xxx.
 *
 *
INPUT-OUTPUT SECTION.
FILE-CONTROL.
      SELECT ARTICLES-S ASSIGN TO ARTICLES-S-EX
      ORGANIZATION IS SEQUENTIAL ACCESS MODE IS SEQUENTIAL.
      SELECT ARTICLES-R ASSIGN TO ARTICLE-R-EX
      ORGANIZATION IS RELATIVE
      ACCESS MODE IS RANDOM RELATIVE KEY IS RK.
      SELECT MOUVEMENTS ASSIGN TO MVTS-EX
      ORGANIZATION IS SEQUENTIAL MODE IS SEQUENTIAL.
```

Ce programme manipule

- un fichier ARTICLES-S, d'organisation séquentielle, sur disque, de numéro logique 1
- un fichier ARTICLE-R, d'organisation relative, de numéro logique 2
- un fichier MOUVEMENTS, d'organisation séquentielle, de numéro logique 3

## 2.2. Le paragraphe facultatif I-O CONTROL

Il spécifie, le cas échéant, la zone mémoire d'E/S qui doit être partagée par différents fichiers :

[SAME [RECORD] AREA FOR fichier1 ,fichier 2 ...]...

Ceci équivaut à une redéfinition de zone.

# 3. Data Division

Il faudra y inclure une FILE SECTION permettant de définir la structure de chaque fichier et celle des enregistrements correspondants.

## 3.1. Syntaxe générale

Pour toutes les organisations

```
DATA DIVISION.
FILE SECTION.

[ FD fich BLOCK CONTAINS [ent-1 TO] ent- 2   { RECORDS }  ]
                                             { CHARACTERS }

[ DATA      { RECORD  } { IS  }  nd-1, [nd-2]... ]
            { RECORDS } { ARE }

[ LABEL  { RECORD IS  }  { STANDARD }  ]
         { RECORDS ARE }  { OMITTED  }

[RECORD CONTAINS [ent-2 TO ] ent-3 CHARACTERS]

description d'article .....
```

**Toutes les clause sont facultatives en ANS85**

- La clause BLOCK n'est utile que pour les fichiers sur bande et spécifie la taille d'un bloc.
- Les noms de données nd-1, nd-2, ... correspondent aux noms attribués à chaque description d'article. Ils correspondent à des redéfinitions de la zone d'entrée-sortie du fichier.
- Seule la clause LABEL peut être obligatoire et précise la présence (STANDARD) ou l'absence (OMITTED) d'étiquettes pour le fichier, selon l'organisation et le support retenus. Elle n'est significative que pour les fichiers sur bande.

# 4. Procedure Division

## 4.1. Ouverture et fermeture d'un fichier

Tout fichier manipulé dans un programme doit être ouvert auparavant et fermé en fin de traitement.

```
OPEN  { INPUT  fichier ...
      { OUTPUT fichier ...
      { I-O    fichier ...
      { EXTEND   fichier ...

CLOSE { fichier1 WITH { NO REWIND } }...
                      { LOCK      }
```

- L'ouverture rend disponible le fichier, prépare en mémoire centrale une zone enregistrement associée au programme, vérifie les étiquettes en `INPUT` ou les crée en `OUTPUT`.

- Un pointeur d'enregistrement courant est positionné pour désigner le premier enregistrement à traiter pour les fichiers en lecture (`INPUT`) ou en mise à jour (`I-O`). Ce premier enregistrement est déterminé en fonction de l'organisation du fichier (premier enregistrement physique pour les fichiers séquentiels, enregistrement de clé donnée, valide, pour les autres organisations).

- L'ouverture en mode `OUTPUT` positionne le pointeur d'enregistrement courant sur l'espace correspondant au premier enregistrement du fichier.

- L'ouverture en mode `EXTEND` positionne le pointeur d'enregistrement courant sur l'espace disponible se trouvant après le dernier enregistrement du fichier et permet de ce fait l'adjonction de nouveaux enregistrements en fin de fichier (uniquement pour fichiers d'organisation séquentielle).

- Pour les rajouts d'enregistrements il faudra ouvrir le fichier :

  - soit en mode `I-O` ou `OUTPUT` pour les fichiers non séquentiels, et en accès non séquentiel,

  - soit en mode `EXTEND`, pour les fichiers séquentiels.

- L'instruction `CLOSE` est indispensable pour :

  - éviter que le fichier ne reste à l'issue du programme dans un état indéfini qui pourrait empêcher de la rouvrir (fichier perdu)

  - éviter la perte des derniers enregistrements écrits qui se trouveraient encore dans le buffer d'entrée-sortie (buffer incomplet).

4.2. Opérations possibles sur les fichiers

| Mode d'Accès | Instruction | INPUT | OUTPUT | I-O | EXTEND |
|---|---|---|---|---|---|
| Séquentiel | READ | X | | X | |
| | WRITE | | X | | X |
| | REWRITE | | | X | |
| Random | READ | X | | X | |
| | WRITE | | X | X | |
| | REWRITE | | | X | |
| | START | | | | |
| | DELETE | | | X | |
| Dynamic | READ | X | | X | |
| | WRITE | | X | X | |
| | REWRITE | | | X | |
| | START | X | | X | |
| | DELETE | | | X | |

**4.2.1. READ**

L'instruction provoque la lecture d'un enregistrement ; il sera utilisé sous la forme

Format 1(accès séquentiel)

```
READ  fich [NEXT] RECORD  [INTO nd-1]
[AT END phrase impérative1]

[NOT AT END phrase impérative2]
[END-READ]
```

Format 2 (accès sélectif)

```
      READ  fich RECORD  [INTO nd-1]  [KEY IS nd]
[INVALID KEY phrase impérative-1]

[INVALID KEY phrase impérative-2]
[END-READ]
```

L'option `KEY` permet d'indiquer la clé à utiliser pour la recherche, dans le cas d'emploi de clès alternées ; elle sera aussi valable, en accès dynamique pour les lectures séquentielles consécutives, jusqu'à établissement d'une nouvelle référence. (fichiers indexés seulement).

L'option `NEXT` est utilisée pour une lecture séquentielle en accès dynamique.

L'instruction `READ` rend disponible l'enregistrement du fichier désigné par le pointeur. il est accessible dans la zone d'entrée décrite en `DATA DIVISION` sous la rubrique `FD`.

Le fichier doit être ouvert en mode `INPUT` ou `I-O`. En cas de fin de fichier ou de clé invalide, la phrase introduite par les clauses `AT END` ou `INVALID KEY` est exécutée. Dans le cas contraire, (lecture réussie) le programme se poursuit normalement en séquence.

Après une lecture réussie, le pointeur d'enregistrement désigne le prochain enregistrement valide pour les fichiers en accès séquentiel. Dans les autres cas, le pointeur est positionné sur l'enregistrement identifié par la clé. La longueur des enregistrements du fichier réel doit être identique, à celle de la zone d'entrée associée au fichier (cas d'un fichier avec des enregistrements de longueur fixe) ; elle doit être dans tous les cas inférieure ou égale à celle de la zone d'entrée (cas d'un fichier avec des enregistrements de longueur variable).

### 4.2.2. WRITE

L'instruction provoque l'écriture d'un enregistrement dans un fichier ouvert en OUTPUT, I-O ou EXTEND. Elle sera utilisée sous la forme

Format 1(en accès séquentiel)
```
WRITE enreg [FROM nd-1]
[END-WRITE]
```

Format 2 (en accès sélectif)
```
WRITE enreg [FROM nd-1]
[INVALID KEY phrase impérative-1]

[NOT INVALID KEY phrase impérative-2]
[END-WRITE]
```

L'instruction `WRITE` délivre un enregistrement au système de gestion de fichier. Cet enregistrement n'est alors **plus** disponible dans la zone de sortie associée au fichier (paragraphe `FD` de la `DATA DIVISION`), à moins que l'exécution de l'instruction `WRITE` n'échoue (violation de limite de fichier ou clé invalide par exemple).

Même remarque que pour la lecture : la longueur des enregistrements du fichier réel doit être compatible avec la description du fichier dans programme COBOL.
Pour les fichiers d'organisation indexée, une écriture en accès séquentiel suppose que les enregistrements sont triés dans l'ordre croissant des clés.

### 4.2.3. Options READ... INTO... et WRITE ... FROM ...

Elles permettent de combiner en une seule instruction les instructions de lecture ou d'écriture classqiue et les transferts entre le buffer d'entrée-sortie et une zone de travail (définie par exemple en `WORKING-STORAGE`) et inversement.

Ainsi :

```
                READ fichier INTO donnée ...
équivaut à      READ fichier ...
                MOVE enreg. TO donnée
```

- enreg. étant une description d'article au niveau `FD` du fichier correspondant
- donnée est une zone quelconque de la `DATA DIVISION`.

De même :

```
                WRITE enreg. FROM donnée ...
équivaut à      MOVE donnée TO enreg.
```

WRITE enreg ....

### 4.2.4. REWRITE

L'instruction permet la réécriture d'un enregistrement modifié dans un fichier ouvert en I-O.

Format1 (accès séquentiel) REWRITE enreg [FROM nd-1]

Format 2 (accès sélectif) REWRITE enreg [FROM nd-1] [INVALID KEY ph-imp1]

[NOT INVALID KEY ph-imp2]
[END-REWRITE]

L'enregistrement modifié est le dernier lu en accès séquentiel, celui pointé par la clé pour les autres cas. Cet ordre n'est pas utilisable pour les fichiers sur bande.

### 4.2.5. DELETE

L'instruction permet la suppression d'un enregistrement dans un fichier d'organisation relative ou séquentielle indexée. Elle s'écrit :

DELETE fichier RECORD [INVALID KEY ph-imp1]

[NOT INVALID KEY ph-imp2]
[END-DELETE]

L'enregistrement supprimé est le dernier lu en accès séquentiel, celui pointé par la clé pour les autres cas.

### 4.2.6. START

L'instruction permet le positionnement dans un fichier d'organisation relative ou séquentielle indexée, en vue d'une lecture séquentielle à partir de l'enregistrement pointé.Elle s'écrit :

START fichier ( KEY ( IS EQUAL TO | IS = | IS GREATER THAN | IS > | IS NOT LESS THAN | IS NOT < ) nd-1 )

[INVALID KEY ph-imp1]

[NOT INVALID KEY ph-imp2]
[END-START]

Cette instruction ne concerne que les fichiers relatifs ou indexés, en accès séquentiel ou dynamique, ouverts en mode INPUT ou I-O. Si KEY IS n'est pas spécifié, l'option par défaut est KEY IS EQUAL TO nd-1.

Le système de gestion de fichiers compare les clés des enregistrements avec la valeur fournie par nd1 et positionne le pointeur d'enregistrement courant sur le premier enregistrement valide dont la clé satisfait la comparaison. En cas d'échec, la condition INVALID KEY est validée et la phrase impérative

correspondante est exécutée (le pointeur est alors dans une position indéfinie). nd-1 est le nom de clé donné en `INPUT-OUTPUT SECTION` (clause `RELATIVE KEY` ou `RECORD KEY`).

Exemple :

```
START FICH KEY IS > RKEY INVALID KEY.
```

ou encore

```
START FICH2 INVALID KEY.
```

# 5. Erreurs d'entrée-sortie et code d'état du fichier

Toute opération sur un fichier (ouverture, lecture, écriture, suppression, etc...) entraîne la mise à jour par le SGF, d'un code d'état du fichier indiquant la réussite ou la nature de l'échec de l'opération en question. Ce code est accessible au programmeur et lui permet de gérer lui-m^me les erreurs d'Entreés/sorties, d'interrompre ou de poursuivre d'exécution du programme, d'afficher des messages, etc... La valeur de ce code est rangée par le SGF dans la rubrique désignée par la clause `FILE STATUS` de la phrase `SELECT`.

## 5.1. FILE STATUS

S'il est utilisé, le `FILE STATUS` doit être

- **défini** dans `l'INPUT-OUTPUT SECTION` (phrase `SELECT` correspondant au fichier ; clause `FILE STATUS`)
- **décrit** dans la `DATA DIVISION` (`PICTURE XX`)
- **analysé** selon la grille jointe

## 5.2. Gestion des erreurs d'entrées / sortie

Le traitement des erreurs d'E/S s'effectuera à l'aide d'un ou plusieurs outils suivants :

- analyse du code-état associé au fichier
- utilisation des locutions standards `AT END` et `INVALID KEY`
- utilisation de sections de déclaratives

Les **déclaratives** sont constituées d'un ensemble de **sections**, formant un bloc autonome.
Elles sont regroupées en début de `PROCEDURE DIVISION.` A chaque section est associée une phrase `USE`, spécifiant le rôle (cas d'emploi) de la déclarative. L'ensemble des déclaratives est délimité par un en-tête et un pied:

```
PROCEDURE DIVISION.
DECLARATIVES.
.....
.....
END DECLARATIVES.
```

Constituant une entité séparée de la `PROCEDURE DIVISION`, il ne peut y avoir (sauf cas particulier) de référence à des blocs standards de celle-ci depuis une section de `DECLARATIVES` (et réciproquement).

Une section de déclaratives traitant les erreurs d'E/S sera identifiée par une phrase `USE` au format suivant :

```
USE AFTER STANDARD { ERROR     } PROCEDURE ON { I-O    }
                   { EXCEPTION }              { INPUT  }
                                              { OUTPUT }
                                              { EXTEND }
                                              fich-1 [fich-2]...
```

- Une phrase USE par section, après l'en-tête.
- L'organisation et l'accès des fichiers concernés (implicitement ou explicitement) peuvent être différents.
- Une référence explicite à un fichier prévaut sur la référence par le mode d'ouverture.
- `EXCEPTION` et `ERROR` sont équivalents.

A la suite d'une erreur d'E/S/, le système de gestion des fichiers provoquera éventuellement l'exécution dela section déclarative associée au fichier concerné ; après exécution de cette section, le contrôle reviendra à l'instruction suivant celle qui a déclenché l'appel.

Le traitement général d'une erreur d'E/S est donc le suivant :

1. Le SGF renseigne le code-état associé au fichier (s'il a été défini dans le programme)

2. Si l'instruction d'E/S a provoqué une **erreur standard** (i.e. validant une locution `AT END` ou `INVALID KEY`) :

   2.a) Si la locution est fournie, la phrase impérative correspondant est exécutée ; puis le programme se poursuit en séquence .

   2.b) sinon, si une section déclarative existe pour ce fichier, elle est exécutée, puis le programme se poursuit en séquence.

   2.c) Si aucun des traitements en a) ou en b) n'est prévu, le programme sera soit **interrompu**, (soit poursuivi en séquence, suivant le système utilisé).

3. Si l'erreur d'E/S est NON-STANDARD :
   le traitement est le même qu'en 2, à partir de 2.b)

# 6. Codes d'Etat des Opérations d'E/S

| Type | S1 | S2 | Seq | Ind | Rel | Signification |
|---|---|---|---|---|---|---|
| OK | 0 | 0 | | | | Entree-Sortie réussie. |
| | 0 | 2 | | | | Entrée-Sortie réussie, mais le système a détecté une clé dupliquée. |
| | 0 | 4 | | | | Lecture faite, mais l'enregistrement n'a pas la longueur déclarée dans le programme |
| | 0 | 5 | | | | Tentative d'OPEN d'un fichier OPTIONAL qui n'est pas présent. En mode I-O ou EXTEND, le fichier est créé. |
| | 0 | 7 | | | | A l'occasion d'une instruction CLOSE ou OPEN avec une des clauses NO REWIND, REEL/UNIT ou REMOVAL, on constate que le fichier n'est pas sur bande magnétique. |
| AT END | 1 | 0 | | | | Fin de fichier en lecture. |
| | 1 | 4 | | | | Pour les fichiers relatifs, le nombre d'enregistrements du fichier dépasse la capacité de la clé relative. |
| INVALID KEY | 2 | 1 | | | | Problème de séquence en traitement séquentiel. Par exemple, on a lu un enregistrement et on veut le réécrire avec une clé différente. |
| | 2 | 2 | | | | On veut écrire un enregistrement alors qu'il en existe déjà un avec une clé identique. |
| | 2 | 3 | | | | Tentative d'accès à un enregistrement qui n'existe pas. |
| | 2 | 4 | | | | Tentative de WRITE hors des limites d'un fichier relatif ou indexé. |
| Erreur Permanente | 3 | 0 | | | | Erreur permanente. |
| | 3 | 5 | | | | OPEN sur un fichier, non OPTIONAL, qui n'est pas présent. |
| | 3 | 7 | | | | OPEN sur un fichier supposé en accès direct qui, en fait, ne l'est pas. |
| | 3 | 8 | | | | OPEN sur un fichier au préalable fermé par CLOSE WITH LOCK . |
| | 3 | 9 | | | | OPEN impossible suite à un conflit entre la description Cobol et la réalité. |
| Erreur Logique | 4 | 1 | | | | Tenœtive d'OPEN sur un fichier déjà ouvert. |
| | 4 | 2 | | | | Tenmtive de CLOSE sur un fichier non ouvert. |
| | 4 | 3 | | | | Tentative de DELETE ou REWRITE sur un enregistrement qui n'a pu être lu par un READ précédent. |
| | 4 | 4 | | | | Tentative de WRITE ou de REWRITE d'un enregistrement dont la dimension ne correspond pas à la description de fichier |
| | 4 | 6 | | | | Tentative de lecture séquentielle d'un enregistrement alors que la lecture précédente n'a as réussi. |
| | 4 | 7 | | | | Tentative de READ ou START (fichiers relatifs ou indexés) sur un fichier non ouvert en INPUT ou en I-O. |
| | 4 | 8 | | | | WRITE sur un fichier non ouvert en OUTPUT, en I-O ou en EXTEND. |
| | 4 | 9 | | | | Tentative de DELETE ou REWRITE sur un fichier non ouvert en I-0. |
| Erreur Run-time | 9 | 1 | | | | Fichier endommagé. |
| | 9 | X | | | | Erreur irréparable - fichier endommagé. |

# 7. Le Module Séquentiel

## 7.1 Les opérations d'entrées / sorties sur un fichier d'organisation séquentielle

L'**accès** à un fichier d'organisation séquentielle ne peut être que **séquentiel**.

A l'ouverture, le pointeur d'enregistrement se positionne sur le premier enregistrement disponible du fichier. Les opérations possibles sont :

- la lecture pour un fichier ouvert en mode `INPUT` ou `I-O`
- l'écriture pour un fichier ouvert en mode `OUTPUT` ou `EXTEND`
- la réécriture pour un fichier ouvert en mode `I-O`

### 7.1.1. LECTURE

- Elle concerne l'enregistrement indiqué par le pointeur d'enregistrement courant. L'ordre de lecture s'écrit suivant le Format 1.
- Le nom de fichier est défini dans le paragraphe `FD` de la `DATA DIVISION`.
- La locution facultative `AT END` permet l'exécution d'instructions (calculs, branchements, etc...) au cas où le pointeur d'enregistrement pointe sur la fin de fichier. Dans ce cas le code d'état du fichier prend la valeur "10" et la lecture échoue. Le repositionnement en début de fichier s'obtient en fermant puis rouvrant ce dernier. (`CLOSE`... suivi de `OPEN`...).
- A l'issue d'une lecture réussie, le pointeur d'enregistrement pointe sur l'enregistrement suivant.

### 7.1.2. ECRITURE

- Elle ne peut se faire que dans un fichier ouvert en mode `OUTPUT` ou `EXTEND`, au Format 1.
- enreg désigne une description d'article de ce fichier (voir paragraphe FD).
- Un nouvel enregistrement, dont la valeur est celle de la zone de sortie associée au fichier, est créé à la suite des enregistrements existants.

### 7.1.3. RE-ECRITURE

Elle est utilisée pour modifier un enregistrement existant. Plus précisément, elle permet de réécrire sur le fichier, après modification, le dernier enregistrement obtenu par une lecture réussie.

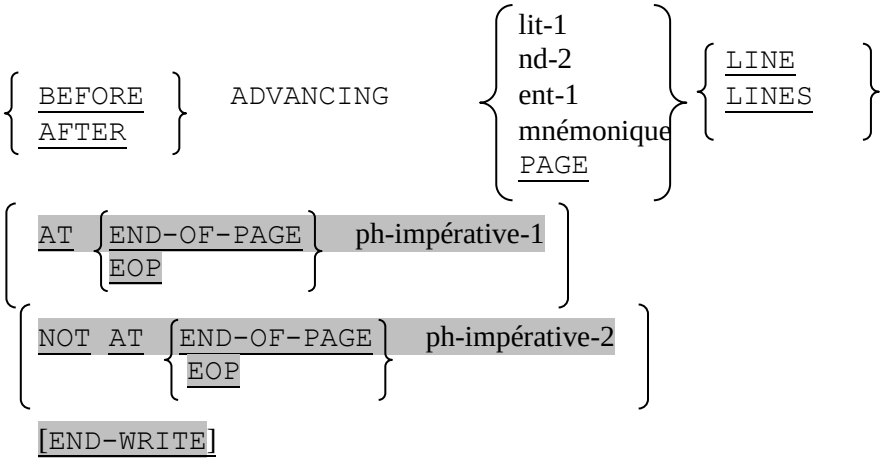
- Format 1
- Le fichier doit être ouvert en mode `I-O`.
- La longueur de l'enregistrement réécrit doit être égale à celle de l'enregistrement modifié (donc lu).

## 7.2. CREATION DE FICHIERS D'EDITION

On ne modifie jamais un fichier d'édition qui sera de ce fait toujours un fichier d'organisation séquentielle ouvert en mode OUTPUT. Afin de répondre aux impératifs de la mise en page, le programmeur dispose au niveau de l'ordre WRITE d'une option complémentaire lui permettant de gérer l'avance du papier (sauts de lignes et de page).
Par ailleurs, il est nécessaire de préciser pour un fichier d'édition un mode particulier au niveau de la phrase SELECT, clause ORGANIZATION du paragraphe FILE-CONTROL.

### 7.2.1. Syntaxe générale de l'ordre

WRITE enreg [ FROM nd-1]




### 7.2.2. Exemples

```
WRITE ART-STCK AFTER ADVANCING PAGE;
```
écriture en début de page suivante
```
WRITE ART-STCK AFTER ADVANCING 2 LINES.
WRITE ART-STCK AFTER ADVANCING N LINES.
```
(la valeur de N ayant été précisée précédemment).
écriture après saut de 2 (ou N) lignes

```
WRITE ART-STCK BEFORE 3 LINES.
```
saut de 3 lignes après l'écriture.

**Remarque importante concernant les fichiers d'organisation séquentielle**
**On ne peut pas supprimer un enregistrement dans un fichier d'organisation séquentielle.**

# 8. Le Module Indexé

## 8.1. Description d'un fichier d'organisation séquentielle indexée

- Chaque enregistrement est repéré par une clé d'enregistrement faisant partie intégrante de cet enregistrement. Cette clé précisée par la clause RECORD KEY est une rubrique alphanumérique qui occupe une position fixe dans l'enregistrement.

La déclaration d'un tel fichier en COBOL se fera de la façon suivante, par exemple :

```
SELECT HISTO     ASSIGN TO FLCH
                 ORGANIZATION IS INDEXED
                 ACCESS IS DYNAMIC
                 RECORD KEY IS CLE-HISTO
                 STATUS IS ST-HISTO.
```

La description de la clé `CLE-HISTO` est faite dans la `DATA DIVISION` dans la description du fichier `HISTO`, par exemple

```
FD HISTO RECORD STANDARD
     DATA RECORD ENRE-HISTO.
01 ENR-HISTO.
     02 CLE-HISTO.
          03 DATE-MOUV-HISTO    PIC X(6).
          03 NOCPTE-HISTO       PIC X(6).
          03 NO-ORDRE-HISTO     PIC X(3).
     02 ...
        .....
```

## 8.2. L'accès séquentiel

Les enregistrements sont atteints dans l'ordre croissant des clés d'enregistrements.

### 8.2.1. ECRITURE

- `WRITE` Format 2

- Elle permet d'écrire un nouvel enregistrement à la suite des enregistrements déjà créés, le fichier étant ouvert en mode `OUTPUT`. La condition `INVALID KEY` est validée si la clé d'enregistrement est inférieure ou égale à la dernière valeur de clé écrite ou si l'espace alloué au fichier est rempli.

### 8.2.2. LECTURE

- `READ` Format 1

- Elle délivre l'enregistrement de clé immédiatement supérieure à celui qui vient d'être traité, le fichier étant ouvert en mode `INPUT` ou `I-O`.

### 8.2.3. SUPPRESSION

- DELETE fichier RECORD

- Elle permet la suppression du **dernier enregistrement lu**, le fichier étant ouvert en mode I-O.

Exemple :

```
READ HISTO
     .
     .
DELETE HISTO.
```

#### 8.2.4. REECRITURE

- Format 2

- Même procédure que pour les fichiers séquentiels ; la clause INVALID KEY est validée si la clé d'enregistrement ne correspond pas à celle du dernier article lu.

## 8.3. L'accès sélectif (RANDOM)

Les enregistrements sont recherchés par comparaison de la clé d'enregistrement avec les valeurs de clés des enregistrements existant dans le fichier.

### 8.3.1. ECRITURE

- Format 2

- Le fichier étant ouvert en mode I-O ou OUTPUT, l'enregistrement est inséré logiquement dans le fichier en tenant compte de l'ordre croissant des clés. Si un enregistrement valide de même clé existe déjà, la clause INVALID KEY est validée.

#### 8.3.2. LECTURE

- Format 2

- L'enregistrement dont la clé correspond à la valeur courante de la rubrique clé d'enregistrement est délivré dans la zone d'entrée associée au fichier. Si cet enregistrement n'existe pas, la clause INVALID KEY est valide. Le fichier est ouvert en mode I-O ou INPUT.

#### 8.3.3. SUPPRESSION

Elle provoque la suppression logique dans le fichier de l'enregistrement identifié par la clé. La clause INVALID KEY est validée si l'enregistrement n'existe pas. Le fichier est ouvert en mode I-O.

#### 8.3.4. REECRITURE

- Format 2

- Elle concerne un fichier ouvert en mode `I-O` provoque le remplacement de l'enregistrement identifié par la clé par les nouvelles données. La clause `INVALID KEY` est validée si un tel enregistrement n'existe pas.

## 8.4. L'accès dynamique (DYNAMIC)

### 8.4.1. Instructions classiques

- L'utilisation d'un fichier en accès dynamique permet "indifféremment" l'accès sélectif et l'accès séquentiel au cours du même programme.
- Les instructions WRITE, REWRITE, DELETE correspondent à l'utilisation **en accès sélectif** (affectation ou suppression selon la clé).
- La **lecture sélective** se fait par l'instruction au Format 2 après avoir mis à jour la clé.
- La **lecture séquentielle** se traduit par l'instruction

  `READ` fichier `NEXT RECORD` [`INTO` nd-2] [`AT END` phrase]

Dans ce cas le système recherche l'enregistrement **suivant l'ordre des clés**.

- L'accès dynamique permettra ainsi, par exemple, de lire séquentiellement un fichier à partir d'un enregistrement correspondant à une certaine clé. Pour réaliser ce positionnement on utilise l'instruction `START` décrite au paragraphe 4.2.6.

### 8.4.2. POSITIONNEMENT (START)

- Le système compare la valeur figurant dans nd-1 avec les valeurs correspondantes dans les enregistrements du fichier et sélectionne le premier enregistrement satisfaisant à cette comparaison. Pour être accessible celui-ci devra faire l'objet d'une lecture.
- nd-1 peut être la rubrique introduite par la clause `RECORD KEY` ou toute rubrique incluse dans cette dernière, partant de sa 1ère position gauche.
- Si la clause `KEY` est omise, la comparaison porte sur la **clé d'enregistrement complète et l'égalité** des valeurs est recherchée.
- Le fichier est ouvert en `INPUT` ou `I-O` et l'accès est séquentiel ou dynamique.

Exemple :
```
MOVE CLE-LUE  TO CLE-HISTO.
START HISTO KEY NOT < CLE-HISTO INVALID KEY.
….
READ HISTO NEXT.
```

## 8.5. Clés secondaires

Une clé secondaire (ou **alternée**) permet d'accéder aux articles d'un ficher indexé suivant un second critère; ce sera donc une rubrique alphanumérique présente dans chaque article. Un index secondaire

spécifiant cette clé aura été préalablement créé et chargé.

### 8.5.1. Déclaration d'une clé secondaire

- Elle se fait par la clause optionnelle

{ ALTERNATE RECORD KEY IS nd-5 [WITH DUPLICATES] } …

de la phrase SELECT, où nd-5 désigne la clé secondaire.

- La locution DUPLICATES sera spécifiée si la valeur de l'indicatif supplémentaire peut ne pas être unique dans le fichier (synonymes).

- On peut spécifier plusieurs clauses ALTERNATES (i.e. plusieurs clés secondaires), chaque clé étant décrite dans l'article du fichier.

### 8.5.2. Utilisation d'une clé secondaire

Une clé alternée ne peut être spécifiée que dans les instructions de lecture sélective (READ) et de positionnement (START).

#### 8.5.2.1. Lecture sélective

READ fichier RECORD [INTO nd-1] [KEY IS clé] [INVALID KEY ph.imp]

Clé désignant la clé de référence pour la lecture (par défaut, c'est la clé primaire). Si le mode d'accès est dynamique, la clé spécifiée servira aussi de référence pour les lectures séquentielles ultérieures jusqu'à établissement d'une nouvelle clé de référence.

#### 8.5.2.2. Positionnement

La rubrique nd-1 dans la locution KEY de l'instruction START peut désigner une clé secondaire (ou une partie gauche de celle-ci). Elle devient alors clé de référence pour toutes les lectures séquentielles ultérieures.

# 9. Le Module Relatif

## 9.1. Description d'un fichier d'organisation relative

Chaque enregistrement est repéré par sa position relative par rapport au début du fichier. Cette position est fournie par un **entier positif** défini par la clause `RELATIVE KEY IS` nd au niveau de la phrase `SELECT` de ce fichier.

```
SELECT FIART ASSIGN TO FLCH
ORGANIZATION IS RELATIVE
ACCESS MODE IS RANDOM RELATIVE KEY IS CLE1.
```

La description de la donnée `CLE1` se fait dans la `DATA DIVISION`, il s'agit d'un **entier non signé**. Par exemple :

```
* RELATIVE KEY DE FIART :
 77 CLE1 PIC 99 value 1.
```

Remarque :

**Cette rubrique n'a pas le droit d'appartenir à une description d'article du fichier concerné**. Toutefois, rien ne s'oppose à ce que la donnée correspondante à la valeur de la clé soit incorporée dans l'enregistrement par copie dans une rubrique portant un nom différent. On prendra garde cependant à conserver la concordance des valeurs lors des mises à jour d'enregistrements. La clé relative d'un enregistrement ne peut bien sur être modifiée lors d'une mise à jour.

## 9.2. Les différents accès à un fichier relatif

Les fichiers d'organisation relative autorisent comme les fichiers indexés les trois accès :

- séquentiel (SEQUENTIAL)
- sélectif (RANDOM)
- dynamique (DYNAMIC)

L'utilisation des différentes instructions d'entrée et de sortie est rigoureusement semblable à celle décrite dans le module indexé, auquel le lecteur se référera. Il suffit d'y remplacer la notion de clé d'enregistrement par celle de clé relative. Il en est de même pour l'instruction `START` qui portera sur la clé relative.

La gestion de cette clé pour les lectures et les écritures non séquentielles relève du programmeur qui choisira une méthode de calcul de la clé adaptée au contexte de travail (voir les modes d'adressage relatif des fichiers).

On notera qu'après une lecture séquentielle réussie, le système de gestion des fichiers met à jour la rubrique, si la locution (facultative en séquentiel) `RELATIVE KEY` est spécifiée, en y chargeant la valeur correspondant à l'enregistrement qui vient d'être lu. Il en est de même pour l'écriture d'enregistrements en accès séquentiel. On peut ainsi constituer une table des clés relatives des différents articles écrits au fur et à mesure de la création du fichier.