LA DOCUMENTATION AUTOMATIQUE EN TEMPS PARTAGE

par Jacques F. VALLÉE (1)

Sommaire. — *Bien que les méthodes de gestion des fichiers se soient perfectionnées sans cesse au fur et à mesure que les ordinateurs évoluaient, ce n'est qu'après 1965 que sont apparus des langages dits non-procéduraux, adaptés au traitement généralisé des informations numériques et non-numériques. L'objet de ces langages est de réduire par un ordre de magnitude le coût unitaire des interrogations, ' à la demande ' et de faciliter la centralisation des fichiers en vue d'une gestion intégrée. Ces systèmes ont maintenant dépassé le stade du prototype. Le présent article discute leurs performances et présente deux exemples d'applications scientifiques du langage DIRAC fonctionnant en temps partagé sur l'ordinateur de l'Université Stanford.*

Mots-clés : Langages non-procéduraux, temps partagé, documentation automatique, DIRAC, INFOL, Astronomie, Médecine, gestion intégrée, GDML, Université de Stanford.

Dans un article précédent (1) les principes élémentaires des langages de documentation ont été décrits, et des applications de Gestion ont été examinées à titre d'exemple. Il est intéressant de reprendre la question alors que l'utilisation des ordinateurs en temps partagé, et le télétraitement sont devenus choses courantes. Le présent article vise donc à montrer comment les langages de documentation ont quitté le stade du prototype à l'occasion de cette évolution de l'équipement. Les exemples que nous prendrons seront tirés cette fois du domaine scientifique : en particulier nous décrirons les applications en Astronomie et en recherche médicale qui sont actuellement exploitées à l'Université Stanford. Auparavant, il n'est pas inutile de faire le point des progrès réalisés récemment dans le domaine des applications de Gestion.

1. La Gestion des Fichiers

gestion s'est restreinte au codage de ces manipulations. L'industrie a reconnu et formalisé ce problème en créant COBOL pour diminuer le délai entre la formulation et l'exécution des programmes, mais on a payé un double tribut : en temps d'exécution d'une part, en lourdeur de formulation d'autre part. Il est devenu aisé de gérer des enregistrements, mais la notion d'élément de fichier, ou d'entité de base, s'est différenciée. Pour de multiples raisons, parmi lesquelles on trouve la nécessité de prévoir la longueur des zones, leur blocage, etc., les programmeurs ont souvent dispersé les paramètres des entités fondamentales du fichier en plusieurs types d'enregistrements, que l'on regarde comme formant des « fichiers » artificiels. Le problème n'est pas négligeable dans la mesure où l'utilisateur lui-même est responsable de la maintenance de ces fichiers secondaires qui alourdissent considérablement sa tâche.

On n'a fait que transposer encore ces techniques quand la troisième génération est apparue, avec des améliorations mineures : par exemple, on met souvent en librairie les Divisions-Données de programmes fréquemment utilisés. Mais l'objet essentiel de la majorité des programmes écrits en COBOL reste bien la tabulation pure et simple d'éléments de fichiers. Les tables de décision et les systèmes spécialisés à cartes-paramètres n'ont fait que diversifier le problème sans attaquer la racine.

Le temps qui s'écoule entre la formulation d'une question et l'exécution du programme est fréquemment de plusieurs heures, sinon de plusieurs jours. Enfin, on se rend compte que la juxtaposition de ces systèmes, même assortie d'une centralisation plus ou moins artificielle des descriptions de fichiers baptisée « base de données » ne représente pas l'infrastructure que la gestion intégrée exige dans une entreprise moderne. Et l'on retrouve le même problème dans les grandes applications scientifiques.

2. Langages non-procéduraux

La communication avec les machines — et la manipulation des ensembles de données accessibles par leur intermédiaire — s'est faite jusqu'ici grâce à des langages symboliques ou algorithmiques qui se ressemblent sur un point

gestion s'est restreinte au codage de ces manipulations. L'industrie a reconnu et formalisé ce problème en créant COBOL pour diminuer le délai entre la formulation et l'exécution des programmes, mais on a payé un double tribut : en temps d'exécution d'une part, en lourdeur de formulation d'autre part. Il est devenu aisé de gérer des enregistrements, mais la notion d'élément de fichier, ou d'entité de base, s'est différenciée. Pour de multiples raisons, parmi lesquelles on trouve la nécessité de prévoir la longueur des zones, leur blocage, etc., les programmeurs ont souvent dispersé les paramètres des entités fondamentales du fichier en plusieurs types d'enregistrements, que l'on regarde comme formant des « fichiers » artificiels. Le problème n'est pas négligeable dans la mesure où l'utilisateur lui-même est responsable de la maintenance de ces fichiers secondaires qui alourdissent considérablement sa tâche.

On n'a fait que transposer encore ces techniques quand la troisième génération est apparue, avec des améliorations mineures : par exemple, on met souvent en librairie les Divisions-Données de programmes fréquemment utilisés. Mais l'objet essentiel de la majorité des programmes écrits en COBOL reste bien la tabulation pure et simple d'éléments de fichiers. Les tables de décision et les systèmes spécialisés à cartes-paramètres n'ont fait que diversifier le problème sans attaquer la racine.

Le temps qui s'écoule entre la formulation d'une question et l'exécution du programme est fréquemment de plusieurs heures, sinon de plusieurs jours. Enfin, on se rend compte que la juxtaposition de ces systèmes, même assortie d'une centralisation plus ou moins artificielle des descriptions de fichiers baptisée « base de données » ne représente pas l'infrastructure que la gestion intégrée exige dans une entreprise moderne. Et l'on retrouve le même problème dans les grandes applications scientifiques.

2. Langages non-procéduraux

La communication avec les machines — et la manipulation des ensembles de données accessibles par leur intermédiaire — s'est faite jusqu'ici grâce à

LA DOCUMENTATION AUTOMATIQUE EN TEMPS PARTAGE

par Jacques F. VALLÉE (1)

Sommaire. — *Bien que les méthodes de gestion des fichiers se soient perfectionnées sans cesse au fur et à mesure que les ordinateurs évoluaient, ce n'est qu'après 1965 que sont apparus des langages dits non-procéduraux, adaptés au traitement généralisé des informations numériques et non-numériques. L'objet de ces langages est de réduire par un ordre de magnitude le coût unitaire des interrogations, ' à la demande ' et de faciliter la centralisation des fichiers en vue d'une gestion intégrée. Ces systèmes ont maintenant dépassé le stade du prototype. Le présent article discute leurs performances et présente deux exemples d'applications scientifiques du langage DIRAC fonctionnant en temps partagé sur l'ordinateur de l'Université Stanford.*

Mots-clés : Langages non-procéduraux, temps partagé, documentation automatique, DIRAC, INFOL, Astronomie, Médecine, gestion intégrée, GDML, Université de Stanford.

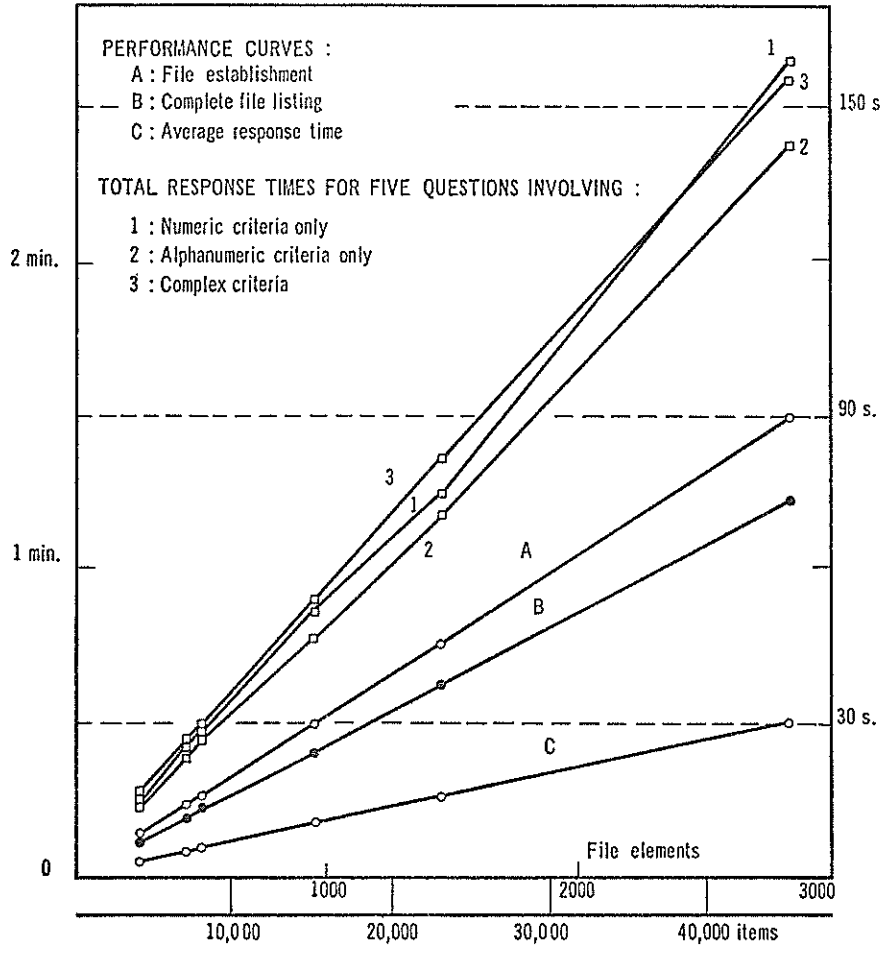
Dans un article précédent (1) les principes élémentaires des langages de documentation ont été décrits, et des applications de Gestion ont été examinées à titre d'exemple. Il est intéressant de reprendre la question alors que l'utilisation des ordinateurs en temps partagé, et le télétraitement sont devenus choses courantes. Le présent article vise donc à montrer comment les langages de documentation ont quitté le stade du prototype à l'occasion de cette évolution de l'équipement. Les exemples que nous prendrons seront tirés cette fois du domaine scientifique : en particulier nous décrirons les applications en Astronomie et en recherche médicale qui sont actuellement exploitées à l'Université Stanford. Auparavant, il n'est pas inutile de faire le point des progrès réalisés récemment dans le domaine des applications de Gestion.

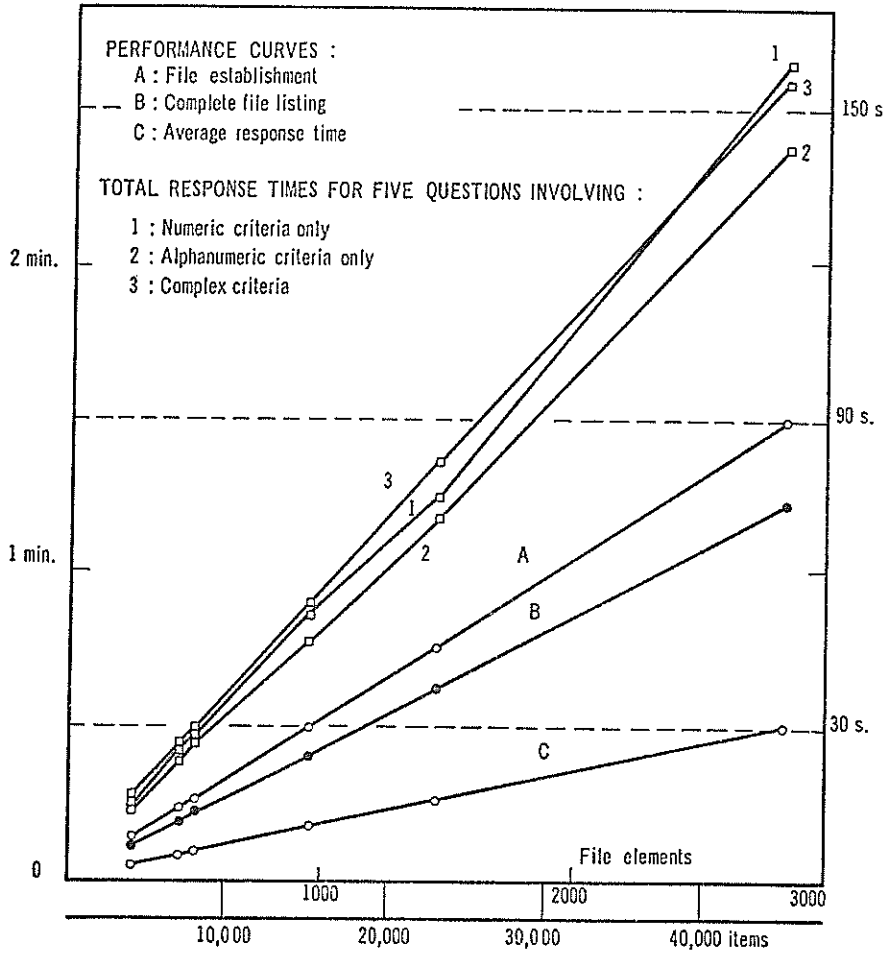
1. La Gestion des Fichiers

Et il est possible de créer une famille de langages dans lesquels un utilisateur non spécialisé (et non plus un programmeur professionnel) peut formuler directement le problème que l'ordinateur doit résoudre, sans plus spécifier *comment* il doit être résolu. Il doit pouvoir demander, par exemple, la liste des employés nés en septembre triée par salaire, et l'obtenir directement sans avoir à écrire un programme spécial, même si cette requête n'a pas été anticipée au moment de la création du fichier du personnel.

Il existe dans la littérature plusieurs descriptions de langages de type non-procédural, et plusieurs analyses comparatives [6,7]. Toutes ces études sont globales et n'abordent ni les questions de performance, ni les détails d'implantation. Les premières sont en général très mal connues et les seconds restent protégés par le secret commercial d'un domaine restreint de l'informatique actuelle, où les équipes de chercheurs sont rares et bien spécialisées. Sans refaire les comparaisons que l'on trouvera dans la littérature, il est utile de citer brièvement les travaux de quelques équipes.

On peut retracer l'origine des langages non-procéduraux à plusieurs développements industriels, universitaires et surtout militaires de la période 1955-1960. Les systèmes généralisés d'entrée-sortie et de tri ont représenté les premiers jalons dans cette direction, mais les constructeurs n'ont pas cherché à pousser des recherches actives dans cette voie : jusqu'en 1965, l'essentiel des travaux théoriques est resté centré sur les compilateurs classiques et les systèmes de base. En 1966, *Control Data* introduisit le langage *Infol*, qui était entièrement non-procédural. Il autorisait des structures d'information complexes, en particulier dans sa version *Infol2* développée à l'Université Northwestern, où des « branches » de l'arbre d'information constituant un élément de fichier pouvaient être associées à des niveaux divers. Dès 1965, IBM avait annoncé un langage semblable, appelé GIS, qui est apparu sur le marché à la fin de 1969. General Electric avait aussi introduit IDS (Integrated Data Store), une extension de COBOL d'une puissance théorique très intéressante, au prix d'une certaine lourdeur, semble-t-il, dans un environnement de gestion. Tous ces prototypes ont été révisés et adaptés et leurs versions actuelles semblent satisfaire les utilisateurs. De même, on cite souvent MARK IV (Informatics),


PERFORMANCE CURVES :
A : File establishment
B : Complete file listing
C : Average response time
TOTAL RESPONSE TIMES FOR FIVE QUESTIONS INVOLVING :
1 : Numeric criteria only
2 : Alphanumeric criteria only
3 : Complex criteria
2 min.
1 min.
0
150 s
90 s.
30 s.
1
2
3
A
B
C
File elements
1000
2000
3000
10,000
20,000
30,000
40,000 items


PERFORMANCE CURVES :
A : File establishment
B : Complete file listing
C : Average response time
TOTAL RESPONSE TIMES FOR FIVE QUESTIONS INVOLVING :
1 : Numeric criteria only
2 : Alphanumeric criteria only
3 : Complex criteria
150 s
90 s.
30 s.
2 min.
1 min.
0
1
2
3
A
B
C
File elements
1000
2000
3000
10,000
20,000
30,000
40,000 items

non spécialisé (et non plus un programmeur professionnel) peut formuler directement le problème que l'ordinateur doit résoudre, sans plus spécifier *comment* il doit être résolu. Il doit pouvoir demander, par exemple, la liste des employés nés en septembre triée par salaire, et l'obtenir directement sans avoir à écrire un programme spécial, même si cette requête n'a pas été anticipée au moment de la création du fichier du personnel.

Il existe dans la littérature plusieurs descriptions de langages de type non-procédural, et plusieurs analyses comparatives [6,7]. Toutes ces études sont globales et n'abordent ni les questions de performance, ni les détails d'implantation. Les premières sont en général très mal connues et les seconds restent protégés par le secret commercial d'un domaine restreint de l'informatique actuelle, où les équipes de chercheurs sont rares et bien spécialisées. Sans refaire les comparaisons que l'on trouvera dans la littérature, il est utile de citer brièvement les travaux de quelques équipes.

On peut retracer l'origine des langages non-procéduraux à plusieurs développements industriels, universitaires et surtout militaires de la période 1955-1960. Les systèmes généralisés d'entrée-sortie et de tri ont représenté les premiers jalons dans cette direction, mais les constructeurs n'ont pas cherché à pousser des recherches actives dans cette voie : jusqu'en 1965, l'essentiel des travaux théoriques est resté centré sur les compilateurs classiques et les systèmes de base. En 1966, *Control Data* introduisit le langage *Infol*, qui était entièrement non-procédural. Il autorisait des structures d'information complexes, en particulier dans sa version *Infol2* développée à l'Université Northwestern, où des « branches » de l'arbre d'information constituant un élément de fichier pouvaient être associées à des niveaux divers. Dès 1965, IBM avait annoncé un langage semblable, appelé GIS, qui est apparu sur le marché à la fin de 1969. General Electric avait aussi introduit IDS (Integrated Data Store), une extension de COBOL d'une puissance théorique très intéressante, au prix d'une certaine lourdeur, semble-t-il, dans un environnement de gestion. Tous ces prototypes ont été révisés et adaptés et leurs versions actuelles semblent satisfaire les utilisateurs. De même, on cite souvent MARK IV (Informatics), TDMS (Systems Development Corporation) ainsi que SCORE, ML/3,

avaient été employés.

3. Traitement non-procédural en temps partagé

En dépit des avantages que nous avons cités, les langages non-procéduraux actuellement offerts ne représentent qu'un perfectionnement des méthodes de gestion des fichiers que nous avons rappelées au début de cet article, et qui caractérisent la seconde génération. On a certes tiré profit des méthodes d'accès aléatoire, mais on n'échappe pas aux contraintes classiques, les deux plus graves étant : 1) la difficulté de définition d'interfaces avec d'autres systèmes et 2) le caractère local, centralisé du traitement des données. Ces obstacles sont bien connus des utilisateurs et freinent leur enthousiasme pour les nouveaux langages.

Pour s'affranchir de ces contraintes, il est nécessaire de disposer d'un ordinateur fonctionnant en temps partagé, et d'un système permettant le télétraitement et l'accès direct, d'une part; d'un traducteur construit de telle façon que ses échanges avec son environnement ne causent pas de difficulté de conversion, d'autre part. C'est dans cette direction que les recherches du groupe de Stanford se sont orientées.

Le prototype de langage que nous avons utilisé sur une 360/67 s'appelle DIRAC, le nom étant composé des initiales des cinq types d'information reconnus par le système (Date, Integer, Real, Alphanumeric, Code). Il est axé exclusivement sur l'exploitation en temps partagé à partir de terminaux et nous n'avons pas l'intention d'en offrir une version batch. La base de données est gérée par ORVYL, le superviseur écrit par Stanford, c'est-à-dire que toutes les transactions avec le système des fichiers se font par l'intermédiaire de blocs standards dont la gestion détaillée revient ensuite à DIRAC.

Les enregistrements ont une structure à trois niveaux semblable à celle d'INFOL [1].

Nous recherchons ici une simplification structurale : pour OS/360, DIRAC est un utilisateur comme un autre. Pour DIRAC, le système tout entier (y compris la gestion physique des fichiers) se comporte comme un *operating*

téristique originale dans un système de ce genre est de permettre le traitement de textes en vocabulaire naturel et la recherche d'expressions alphanumériques contenant des caractères *majuscules ou minuscules.* Cette caractéristique permet d'aborder des applications fermées aux langages précédents, en particulier dans le traitement des catalogues scientifiques.

Deux exemples de traitement de fichiers réels illustreront cette discussion :

Dans un premier exemple, on donne un fichier hospitalier de structure simple, contenant treize caractéristiques d'un rapport hématologique pour chaque malade. Ces paramètres sont de types différents : la date du rapport et le numéro de la fiche sont, respectivement, une date et un nombre entier.

```
CATION

                CUMUL.TERMINAL TIME :   5.85 MIN
                    CUMUL.CPU TIME :   0.25 MIN

_____
:T
.ES
19691126 AND @7 >= 19691115

_____
: SELECTED

_____
:T
.ES
01126 AND @7 >= 19691115
 @8 CONTAINS "Hodgkin" OR @9 CONTAINS "red cell" )

_____
: SELECTED

_____
:T
.ES
<691126 AND >=691115) AND (@8 CONTAINS "Hodgkin"
) CONTAINS "red cell")
110 EXISTS
112 CONTAINS tumor

_____
: SELECTED
```

Figure 2-1

téristique originale dans un système de ce genre est de permettre le traitement de textes en vocabulaire naturel et la recherche d'expressions alphanumériques contenant des caractères *majuscules ou minuscules*. Cette caractéristique permet d'aborder des applications fermées aux langages précédents, en particulier dans le traitement des catalogues scientifiques.

Deux exemples de traitement de fichiers réels illustreront cette discussion :

Dans un premier exemple, on donne un fichier hospitalier de structure simple, contenant treize caractéristiques d'un rapport hématologique pour chaque malade. Ces paramètres sont de types différents : la date du rapport et le numéro de la fiche sont, respectivement, une date et un nombre entier.

```
CATION

                CUMUL.TERMINAL TIME :   5.85 MIN
                   CUMUL.CPU TIME :   0.25 MIN

_____

.
ES
.9691126 AND @7 >= 19691115

SELECTED
_____

.
ES
.126 AND @7 >= 19691115
@8 CONTAINS "Hodgkin" OR @9 CONTAINS "red cell" )

SELECTED
_____

.
S
691126 AND >=691115) AND (@8 CONTAINS "Hodgkin"
CONTAINS "red cell")
0 EXISTS
2 CONTAINS tumor

SELECTED
```

Figure 2-1

3. Traitement non-procédural en temps partagé

En dépit des avantages que nous avons cités, les langages non-procéduraux actuellement offerts ne représentent qu'un perfectionnement des méthodes de gestion des fichiers que nous avons rappelées au début de cet article, et qui caractérisent la seconde génération. On a certes tiré profit des méthodes d'accès aléatoire, mais on n'échappe pas aux contraintes classiques, les deux plus graves étant : 1) la difficulté de définition d'interfaces avec d'autres systèmes et 2) le caractère local, centralisé du traitement des données. Ces obstacles sont bien connus des utilisateurs et freinent leur enthousiasme pour les nouveaux langages.

Pour s'affranchir de ces contraintes, il est nécessaire de disposer d'un ordinateur fonctionnant en temps partagé, et d'un système permettant le télétraitement et l'accès direct, d'une part; d'un traducteur construit de telle façon que ses échanges avec son environnement ne causent pas de difficulté de conversion, d'autre part. C'est dans cette direction que les recherches du groupe de Stanford se sont orientées.

Le prototype de langage que nous avons utilisé sur une 360/67 s'appelle DIRAC, le nom étant composé des initiales des cinq types d'information reconnus par le système (Date, Integer, Real, Alphanumeric, Code). Il est axé exclusivement sur l'exploitation en temps partagé à partir de terminaux et nous n'avons pas l'intention d'en offrir une version batch. La base de données est gérée par ORVYL, le superviseur écrit par Stanford, c'est-à-dire que toutes les transactions avec le système des fichiers se font par l'intermédiaire de blocs standards dont la gestion détaillée revient ensuite à DIRAC.

Les enregistrements ont une structure à trois niveaux semblable à celle d'INFOL [1].

Nous recherchons ici une simplification structurale : pour OS/360, DIRAC est un utilisateur comme un autre. Pour DIRAC, le système tout entier (y compris la gestion physique des fichiers) se comporte comme un *operating*

```
ALL

  311587
  XXXXXXXX
  24 yr
  cbc
  b69691
  Dr.R.Fu
  25/NOV/1969
  24 yr old Negro female with stage IIb Hodgkin's disease.
  The red cells are pale. Platelets are adequate. Atypical lymphs are noted,
  eosinophilia.
  Megakaryocytes are adequate. Myelogenesis is very active. Erythrogenesis i
  normoblastic. Eosinophilia is prominent. Iron stores are absent
  Blood loss suggested, but increased eosinophilia along with increased myel
  erythrogenesis suggest inflammation and /or tumor.

ELECTED


)U CAN EXIT (BY TYPING AN EXCLAMATION MARK)
V EXECUTION MODE

TION

     CUMUL.TERMINAL TIME :  12.30 MIN
          CUMUL.CPU TIME :   0.30 MIN

INT" AND GO TO A NEW PAGE
```

Figure 2-2

le rapport sur l'échantillon analyse doit contenir le texte « the red cells », ou l'histoire du malade contient le mot « Hodgkin ». Il reste six enregistrements. Le médecin doit donc restreindre à nouveau sa recherche en spécifiant qu'un autre commentaire doit être présent et que l'impression (qui est également un texte libre de longueur imprévisible) contiendra le mot « tumor ». DIRAC localise alors un enregistrement unique, dont le médecin demande l'impression immédiate sur le terminal.

On voit que la tâche n'a nécessité ni l'intervention d'un programmeur, ni un entraînement spécial de l'utilisateur. Celui-ci travaille entièrement dans sa propre terminologie. (La notation @n dénote ici le n-ième paramètre de l'entité considérée).

Dans notre second exemple, un fichier astronomique contient une liste, régulièrement mise à jour, de toutes les supernovae connues avec leurs para-

FLD NAME		DESCRIPTION	1
1	SN	Supernova Number	A
2	zI	Zwicky I System	A
3	zII	Zwicky II System	A
4	Design	Other designation	A
5	Galaxy	Parent Galaxy	A
6	Alpha	Right Ascension 1950.0	A
7	Delta	Declination 1950.0	A
8	Morphology	Morphology of parent	A
9	Mp	Photographic magnitude of galaxy	R
10	Vs	Recession Velocity (km/s)	I
11	Cluster	Cluster Membership	A
12	12	Galactic Longitude	R
13	b2	Galactic Latitude	R
14	Maxdate	Date of maximum	D
15	Discovery	Date of Discovery	D
16	Magnitude	Maximum photog. magnitude	R
17	Position	Position within parent	A

le rapport sur l'échantillon analysé doit contenir le texte « the red cells », ou l'histoire du malade contient le mot « Hodgkin ». Il reste six enregistrements. Le médecin doit donc restreindre à nouveau sa recherche en spécifiant qu'un autre commentaire doit être présent et que l'impression (qui est également un texte libre de longueur imprévisible) contiendra le mot « tumor ». DIRAC localise alors un enregistrement unique, dont le médecin demande l'impression immédiate sur le terminal.

On voit que la tâche n'a nécessité ni l'intervention d'un programmeur, ni un entraînement spécial de l'utilisateur. Celui-ci travaille entièrement dans sa propre terminologie. (La notation @n dénote ici le n-ième paramètre de l'entité considérée).

Dans notre second exemple, un fichier astronomique contient une liste, régulièrement mise à jour, de toutes les supernovae connues avec leurs para-

	FLD NAME	DESCRIPTION	1
1	SN	Supernova Number	A
2	zI	Zwicky I System	A
3	zII	Zwicky II System	A
4	Design	Other designation	A
5	Galaxy	Parent Galaxy	A
6	Alpha	Right Ascension 1950.0	A
7	Delta	Declination 1950.0	A
8	Morphology	Morphology of parent	A
9	Mp	Photographic magnitude of galaxy	R
10	Vs	Recession Velocity (km/s)	I
11	Cluster	Cluster Membership	A
12	12	Galactic Longitude	R
13	b2	Galactic Latitude	R
14	Maxdate	Date of maximum	D
15	Discovery	Date of Discovery	D
16	Magnitude	Maximum photog. magnitude	R
17	Position	Position within parent	A

```
' ALL

   311587
   XXXXXXXX
   24 yr
   cbc
   b69691
   Dr.R.Fu
   25/NOV/1969
   24 yr old Negro female with stage IIb Hodgkin's disease.
   The red cells are pale. Platelets are adequate. Atypical lymphs are noted,
   eosinophilia.
   Megakaryocytes are adequate. Myelogenesis is very active. Erythrogenesis i
   normoblastic. Eosinophilia is prominent. Iron stores are absent
   Blood loss suggested, but increased eosinophilia along with increased myel
   erythrogenesis suggest inflammation and /or tumor.

;ELECTED


'OU CAN EXIT (BY TYPING AN EXCLAMATION MARK)
:W EXECUTION MODE

.TION

      CUMUL.TERMINAL TIME :  12.30 MIN
           CUMUL.CPU TIME :   0.30 MIN

:INT" AND GO TO A NEW PAGE
```

Figure 2-2

traitement en temps réel dans un langage de haut niveau.

Pour chaque supernova on a donné les 23 paramètres indiqués dans la table ci-dessus, la colonne (1) indiquant le type de l'information.

L'astronome étudie les supernovae observées dans Virgo (le système en trouve 24) et demande combien sont fausses ou suspectes. DIRAC localise l'objet s1922α dont l'utilisateur fait imprimer trois paramètres (fig. 3).

```
ACTION
:       SELECT
SELECTION RULES
:       @11 CONTAINS Virgo END

   24 RECORDS SELECTED

ACTION
:       RETAIN

ACTION
:       @1 CONTAINS s END

    1 RECORDS SELECTED

ACTION
:       DISPLAY @1 @10 @11 END

   24
SN                  s1922alpha
Vs                  1243
Cluster             Virgo

    1 RECORDS SELECTED

ACTION
:       RELEASE
```

Figure 3

Revenant alors aux 23 supernovae confirmées dans Virgo (leur numéro ne commence pas par la lettre *s*) on restreint la recherche à celles dont la vitesse de récession *Vs* est connue. On en trouve 19. Parmi celles-ci, on impose main-

```
ACTION
:       @11 CONTAINS Virgo AND @1 DOES NOT CONTAIN s
:       END

   23 RECORDS SELECTED

ACTION
:       RETAIN

ACTION
:       @10 EXISTS END

   19 RECORDS SELECTED

ACTION
:       @10 (<=2000 AND >=1000) END

   11 RECORDS SELECTED

ACTION
:       @22(1) CONTAINS "Mt.Wilson" END

    1 RECORDS SELECTED

ACTION
:       DISPLAY @1 @10 @12 @13 @22 END

    2
SN                1901b
Vs                1617
12                271.15
b2                76.90
Sources         1 Ap.J., 88 (1938 ), 285 -304 -Contr.Mt.Wilson, 25 (1938 )No.600.
                2 XIV Colloque int.Astrophys., Paris (1941 ), 186,188
                3 Ann.Observ.Paris, 9 (1945 )fasc.1, 165 -179.
                4 Astronomie, 55 (1941 ), 78, 106.
                5 Astronomie, 63 (1949 ), 68.
                6 Astronomie, 74 (1965 ), 135.
                7 Lick Obs.Bull., 9 (1917 ), 108 -110
                8 P.A.S.P., 29 (1917 ), 180 -182
                9 Observatory, 41 (1918 ), 103
               10 Proc.Am.Phil.Soc., 81 (1939 ), 272, 275; Harvard reprint (1939 ), no.170.
               11 Obscij katalog peremennyh zvzd (1958 ), 612
               12 Kungl.Sv.Vet.Akad.Handl., 60 (1920 ), 53 -60.
               13 V.J.S.,74 (1939 ), 245 ; Medd.Lunds. astr.Obs.Ser.I 8 (1939 )No.155
```

Figure 4

mate avec toutes les partitions en fonctionnement.

```
ACTION
:       @11 CONTAINS Virgo AND @1 DOES NOT CONTAIN s
:       END
   23 RECORDS SELECTED
ACTION
:       RETAIN
ACTION
:       @10 EXISTS END
   19 RECORDS SELECTED
ACTION
:       @10 (<=2000 AND >=1000) END
   11 RECORDS SELECTED
ACTION
:       @22(1) CONTAINS "Mt.Wilson" END
    1 RECORDS SELECTED
ACTION
:       DISPLAY @1 @10 @12 @13 @22 END
    2
SN                1901b
Vs                1617
12                271.15
b2                76.90
Sources          1 Ap.J., 88 (1938 ), 285 -304 -Contr.Mt.Wilson, 25 (1938 )No.600.
                 2 XIV Colloque int.Astrophys., Paris (1941 ), 186,188
                 3 Ann.Observ.Paris, 9 (1945 )fasc.1, 165 -179.
                 4 Astronomie, 55 (1941 ), 78, 106.
                 5 Astronomie, 63 (1949 ), 68.
                 6 Astronomie, 74 (1965 ), 135.
                 7 Lick Obs.Bull., 9 (1917 ), 108 -110
                 8 P.A.S.P., 29 (1917 ), 180 -182
                 9 Observatory, 41 (1918 ), 103
                10 Proc.Am.Phil.Soc., 81 (1939 ), 272, 275; Harvard reprint (1939 ), no.170
                11 Obscij katalog peremennyh zvzd (1958 ), 612
                12 Kungl.Sv.Vet.Akad.Handl., 60 (1920 ), 53 -60.
                13 V.J.S.,74 (1939 ), 245 ; Medd.Lunds. astr.Obs.Ser.I 8 (1939 )No.155
```

Figure 4

traitement en temps réel dans un langage de haut niveau.

Pour chaque supernova on a donné les 23 paramètres indiqués dans la table ci-dessus, la colonne (1) indiquant le type de l'information.

L'astronome étudie les supernovae observées dans Virgo (le système en trouve 24) et demande combien sont fausses ou suspectes. DIRAC localise l'objet s1922α dont l'utilisateur fait imprimer trois paramètres (fig. 3).

```
ACTION
:       SELECT
SELECTION RULES
:       @11 CONTAINS Virgo END
   24 RECORDS SELECTED
ACTION
:       RETAIN
ACTION
:       @1 CONTAINS s END
    1 RECORDS SELECTED
ACTION
:       DISPLAY @1 @10 @11 END
   24
SN                  s1922alpha
Vs                  1243
Cluster             Virgo
    1 RECORDS SELECTED
ACTION
:       RELEASE
```

Figure 3

Revenant alors aux 23 supernovae confirmées dans Virgo (leur numéro ne commence pas par la lettre *s*) on restreint la recherche à celles dont la vitesse de récession *Vs* est connue. On en trouve 19. Parmi celles-ci, on impose maintenant la condition supplémentaire

[2] W. C. MCGEE, « Generalized File Processing », *Annual Review in Automatic Programming*, vol. 8, Pergamon Press, 1968.

[3] T. W. OLLE, « Non-Procedural languages- the next step in automatic programming », *Annual Review in Automatic Programming*, vol. 6, 1969.

[4] J. D. ARON, « Real-time systems in perspective », *IBM Systems Journal*, vol. 6, nº 1, 1967.

[5] B. G. LAMSON and B. DIMSDALE, « A Natural-language Information Retrieval System », *Proceedings of the IEEE*, vol. 54, nº 12, déc. 1966.

[6] « The Large Data-Base : its organization and user interface », *Data-Base*, nº 1, May 1969.

[7] « Generalized Data-Base Management Systems », *CODASYL Systems Committee Report*, 1969.

[8] J. F. VALLEE and J. A. HYNEK, « DIRAC and astronomical data retrieval », Proceedings of the 1970 Convention of the Association for Computing Machinery, New York, 2 sept. 1970.

[9] J. F. VALLEE, « DIRAC : An interactive retrieval language with computational interface », *Information storage and retrieval journal*, vol. 6, nº 8, déc. 1970.

[10] Trois rapports décrivant le langage DIRAC et ses applications ont été publiés par le centre de calculs de Stanford. Ils sont intitulés :

— The DIRAC Language : Concepts and Facilities.
— Scientific Information Networks : A Case Study.
— Medical Data-Management in Time-sharing.

Ces rapports peuvent être obtenus sur demande adressée à l'auteur.