Bases de données

Bull DPS 7000

IQS-V4-Manuel de référence

Volume 2

Sujet : Documentation de référence concernant les aspects procéduraux du système d'information relationnel IQS : langage de requêtes, processeur de macros, générateur d'états, éditeur de texte IQS et langage de requêtes.

Observations : La révision 04 remplace la révision 03 pour tous les utlisateurs de GCOS 7 IQS-V4.

Version du logiciel : GCOS 7 V8 TS8560.

Logiciel/Matériel requis :

Date : Février 2000

Bull S.A.
CEDOC
Atelier de reprographie
34, rue du Nid de Pie BP 428
49004 ANGERS Cedex 01
FRANCE

Bull HN Information Systems Inc.
Publication Order Entry
FAX: (978) 294-7411
MA030/415
300, Concord Rd.
Billerica, MA 01821
U.S.A.

47 F2 78UR Rev04

Vos suggestions sur la forme et le fond de ce manuel seront les bienvenues. Une feuille destinée à recevoir vos remarques se trouve à la fin du présent manuel.

Ce document est fourni à titre d'information seulement. Il n'engage pas la responsabilité de Bull S.A. en cas de dommages résultant de son application. Des corrections ou modifications au contenu de ce document peuvent intervenir sans préavis ; des mises à jour ultérieures les signaleront éventuellement aux destinataires.

47 F2 78UR Rev04

Préface

Objet du manuel

Le présent manuel constitue la deuxième partie du manuel de référence IQS. Il traite des aspects procéduraux du système d'information relationnel IQS, c'est-à-dire du langage de requêtes, du processeur de macros, du générateur d'états IQS et de l'éditeur de texte IQS.

Utilisateurs concernés

Ce manuel est principalement destiné aux développeurs d'applications utilisant le langage de requêtes.

Pour vérifier un point de syntaxe, il est préférable de se reporter au guide pratique IQS-V4 (75UR).

Pour une description plus détaillée de la programmation en langage de requêtes, se reporter au guide du programmeur IQS-V4 (79UR).

En ce qui concerne les langages GCL et DDL, la gestion générale des accès GAC, les droits d'accès, etc. se référer au guide de l'administrateur IQS-V4 (80UR). Pour une description plus globale de ces sujets, se référer aux manuels GCOS 7 correspondants.

Les utilisateurs non informaticiens exécutant généralement des applications prédéfinies, ce manuel ne leur est d'aucune utilité, sauf indication contraire de l'administrateur IQS. S'ils souhaitent apprendre à écrire des requêtes simples, ils devront plutôt consulter l'initiation au langage de requêtes (74UR) et, éventuellement, la première partie du guide du programmeur IQS-V4 (79UR).

Structure du manuel

Le manuel de référence comprend deux volumes : le premier est consacré au langage "non procédural" (commandes IQS) et le second au langage de requêtes et aux fonctions procédurales d'IQS.

VOLUME 2

Chapitre 1	Présentation d'IQS, concepts de base et caractéristiques.
Chapitre 2	Description des éléments du langage de requêtes IQS.
Chapitre 3	Description des variables système IQS.
Chapitre 4	Description des commandes de l'éditeur de texte IQS.
Chapitres 5 à 7	Description détaillée des instructions du langage de requêtes.
Chapitre 8	Description des clauses du générateur d'états IQS.
Chapitre 9	Création et utilisation de macros IQS.
Chapitre 10	Description de la métabase IQS.
Annexe A	Schémas correspondant aux exemples fournis dans le corps du manuel.
Annexe B	Mots réservés du langage de requêtes.
Annexe C	Cas particuliers de @MESSAGE

Bibliographie

Dans ce manuel, les références sont citées sous forme abrégée (4 derniers caractères alphanumériques).

IQS-V4 Bulletin de logiciel *47 F2 70UR*
IQS-V4 Software Release Bulletin *47 A2 70UR*

IQS-V4 - Présentation générale du système *47 F2 71UR*
IQS-V4 Quick Tour *47 A2 71UR*

Initiation à IQS-V4 ..... *47 F2 72UR*
Getting Started with IQS-V4 ..... *47 A2 72UR*

IQS-V4 - Guide de l'utilisateur novice *47 F2 73UR*
IQS-V4 End User's Guide *47 A2 73UR*

Initiation au langage de requêtes (IQS-V4) *47 F2 74UR*
Getting Started with Query in IQS-V4 ..... *47 A2 74UR*

IQS-V4 - Guide pratique *47 F2 75UR*
IQS-V4 Quick Reference Handbook *47 A2 75UR*

IQS-V4 - Guide de l'utilisateur expert *47 F2 76UR*
IQS-V4 Advanced User's Guide *47 A2 76UR*

IQS-V4 - Manuel de référence - Volume 1 ..... *47 F2 77UR*
IQS-V4 Reference Manual Volume 1 ..... *47 A2 77UR*

IQS-V4 - Manuel de référence - Volume 2 ..... *47 F2 78UR*
IQS-V4 Reference Manual Volume 2 ..... *47 A2 78UR*

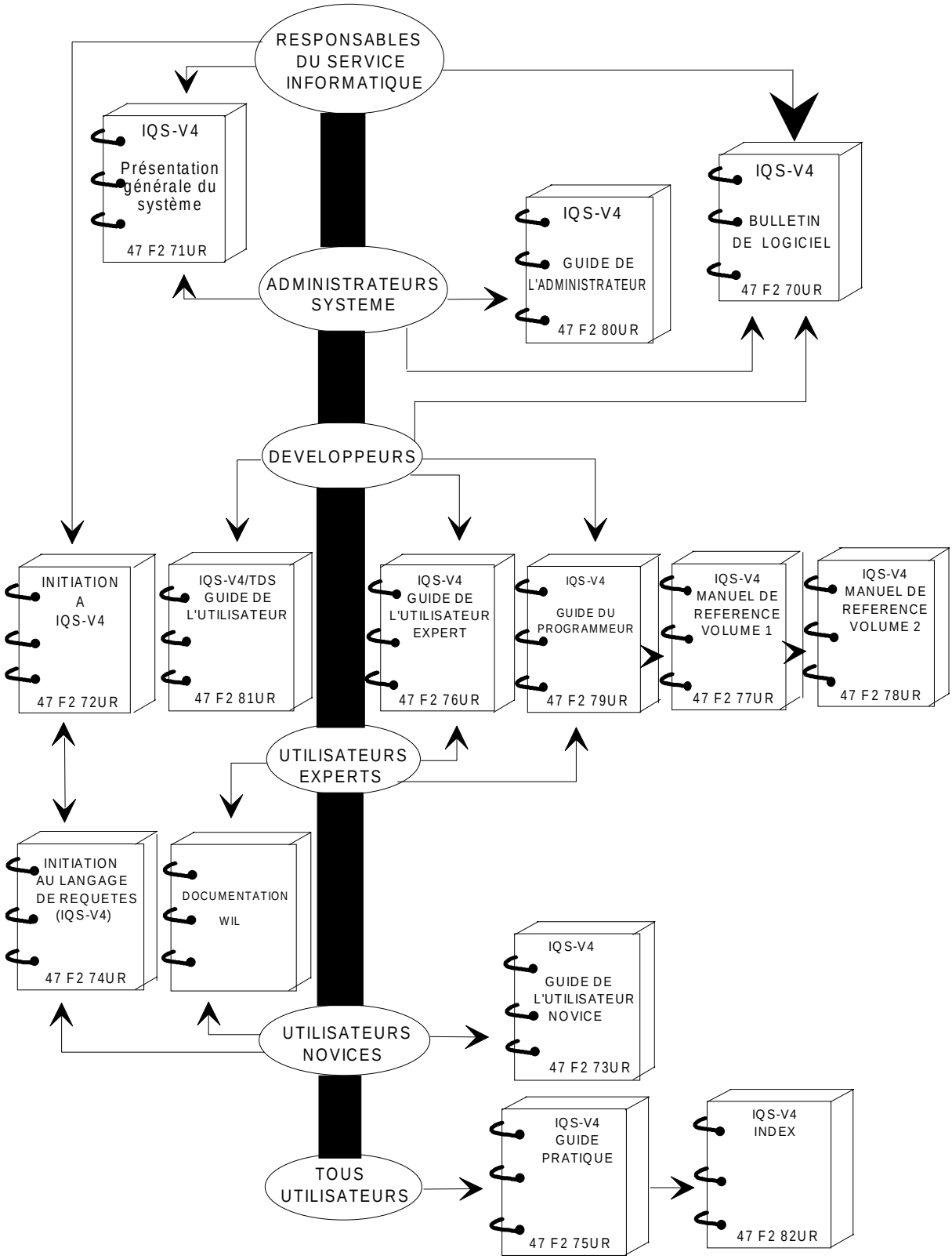
IQS-V4 - Guide du programmeur *47 F2 79UR*
IQS-V4 Programmer's Guide *47 A2 79UR*

IQS-V4 - Guide de l'administrateur *47 F2 80UR*
IQS-V4 Administrator's Guide *47 A2 80UR*

IQS-V4/TDS - Guide de l'utilisateur *47 F2 81UR*
IQS-V4/TDS User's Guide *47 A2 81UR*

IQS-V4 - Index *47 F2 82UR*
IQS-V4 Index *47 A2 82UR*

Pour plus de précisions (disponibilité du manuel, numéro de révision, indice de mise à jour), veuillez consulter le Catalogue de la documentation" et le fascicule "Documents Nouveaux" qui portent tous deux la référence 00 F4 7210.


RESPONSABLES DU SERVICE INFORMATIQUE
IQS-V4 Présentation générale du système 47 F2 71UR
IQS-V4 BULLETIN DE LOGICIEL 47 F2 70UR
ADMINISTRATEURS SYSTEME
IQS-V4 GUIDE DE L'ADMINISTRATEUR 47 F2 80UR
DEVELOPPEURS
INITIATION A IQS-V4 47 F2 72UR
IQS-V4/TDS GUIDE DE L'UTILISATEUR 47 F2 81UR
IQS-V4 GUIDE DE L'UTILISATEUR EXPERT 47 F2 76UR
IQS-V4 GUIDE DU PROGRAMMEUR 47 F2 79UR
IQS-V4 MANUEL DE REFERENCE VOLUME 1 47 F2 77UR
IQS-V4 MANUEL DE REFERENCE VOLUME 2 47 F2 78UR
UTILISATEURS EXPERTS
INITIATION AU LANGAGE DE REQUETES (IQS-V4) 47 F2 74UR
DOCUMENTATION WIL
IQS-V4 GUIDE DE L'UTILISATEUR NOVICE 47 F2 73UR
UTILISATEURS NOVICES
IQS-V4 GUIDE PRATIQUE 47 F2 75UR
IQS-V4 INDEX 47 F2 82UR
TOUS UTILISATEURS

Conventions d'écriture

Les conventions d'écriture utilisées dans ce manuel pour la syntaxe des commandes et instructions sont les suivantes :

ELEMENT	Sous IQS, les majuscules indiquent un mot-clé à introduire tel quel.
ELEMENT	Les majuscules soulignées indiquent une valeur implicite dans un format. Quand l'utilisateur a le choix entre plusieurs valeurs pour un paramètre donné, la valeur implicite (éventuelle) est soulignée. A ne pas confondre avec les informations à fournir par l'utilisateur qui sont également soulignées (voir "Exemples" ci-dessous).
<élément> élément	Un élément en minuscules indique une valeur (ou chaîne) à fournir par l'utilisateur. A noter que dans les exemples, les informations fournies par l'utilisateur peuvent apparaître en majuscules (voir "Exemples" ci-dessous).
[élément]	Les crochets indiquent un élément facultatif pouvant être omis.
{élément1} {élément2} {élément3} ou {élément1|élément2| élément3}	Une colonne d'éléments, ou une succession d'éléments séparés par une barre verticale, entre accolades indique que l'un deux doit être sélectionné, si le paramètre correspondant est retenu.
||élément1|| ||élément2|| ||élément3||	Une colonne d'éléments entre barres verticales indique qu'au moins l'un d'eux doit être sélectionné, chaque élément ne devant figurer qu'une seule fois.
élément...	Les points de suspension indiquent que l'élément qui précède peut être répété une ou plusieurs fois. Si l'élément est entre crochets, cette règle s'applique à tout le contenu des crochets.
+ = , () ' / . $ * -	Caractères spéciaux à introduire tels quels. Ils sont obligatoires si la portion du format dans laquelle ils figurent est utilisée.
Exemples	Dans les exemples, les informations à fournir par l'utilisateur sont toujours soulignées. Elles sont également indiquées en MAJUSCULES sauf dans le cas d'une chaîne contenant des minuscules.
::=	Dans un format développé, signifie "l'élément qui précède peut être explicité comme suit".

Table des matières

1. Présentation du langage procédural IQS

2. Eléments du langage de requêtes

3. Variables système IQS

4. Commandes de l'éditeur de texte IQS

5. Instructions du langage de requêtes IQS (A-D)

6. Instructions du langage de requêtes IQS (E-P)

7. Instructions du langage de requêtes IQS (Q-Z)

8. Générateur d'états IQS

9. Macros IQS

B. Mots réservés du langage de requêtes

C. Cas particuliers de @MESSAGE

Index

Table des Illustrations

Figures

Tableaux

1. Présentation du langage procédural IQS

1.1 Le langage de requêtes

Le processeur IQS permet à l'utilisateur d'écrire des programmes (ou procédures) en langage de requêtes. Ces "requêtes" peuvent être utilisées pour rechercher des informations, les trier et les visualiser sous une certaine forme. Elles permettent également de créer, de mettre à jour et de supprimer des données dans les fichiers de production ou encore, d'enregistrer des échantillons de données dans des fichiers de travail pour utilisation ultérieure.

IQS fournit d'autre part un jeu spécial de commandes de préparation de programmes permettant de manipuler et traiter les requêtes, ainsi que les macros et les descriptions d'état. Ces commandes sont décrites dans le volume 1 du présent manuel de référence.

Le présent manuel contient toutes les informations de référence concernant les éléments et la syntaxe du langage de requêtes.

1.1.1 Requêtes enregistrées

Une manière commode d'utiliser IQS consiste à appeler des requêtes enregistrées. Cette méthode est particulièrement intéressante pour les tâches répétitives demandant des informations légèrement différentes à chaque fois.

Les requêtes peuvent en effet contenir des paramètres dont la valeur est demandée à l'utilisateur par le système au moment de l'exécution. Ces valeurs fournies de manière interactive déterminent le résultat de la requête.

```
 ------------------------------------------------------------
|   V: EXEC R-FACTURE                                        |
|                                                            |
|      CODE DU CLIENT : 12345                                |
|                                                            |
|   CODE-CLIENT  NOM-CLIENT    QTE  MONTANT  DESIGNATION     |
|                                                            |
|      12345     DUPONT J-C     25    325    CRAYONS        |
|                                1     30    TAILLE-CRAYONS |
|                                2     12    GOMMES         |
|                                     _____                  |
|                             TOTAL    367                   |
|                                                            |
|   V:                                                       |
 ------------------------------------------------------------
```

A l'exécution de la requête R-FACTURE, le système demande la valeur requise pour le paramètre CODE-CLIENT. L'utilisateur fournit la valeur 12345. Le résultat de la requête est ensuite visualisé.

1.1.2 Liste des instructions du langage de requêtes

ACCEPT	Demande d'introduction de données à partir du terminal utilisateur ou d'un fichier traitement par lots.
AFTER...CHANGE	Instruction de contrôle. Les valeurs prises en compte sont celles de l'article fourni par la boucle courante de READ ou RETRIEVE.
ALTER	Visualisation des anciennes valeurs et demande d'introduction de nouvelles valeurs au terminal utilisateur ou à partir d'un fichier traitement par lots.

ASSIGN	Association d'un fichier de travail à un fichier séquentiel ou d'un fichier d'impression à SYSOUT (pour sortie sur une imprimante ligne) ou à un fichier permanent (pour impression ultérieure).
BEFORE...CHANGE	Instruction de contrôle. Les valeurs prises en compte sont celles de l'article fourni par la boucle précédente de READ ou RETRIEVE.
CANCEL FORMAT	Suppression du lien existant entre un fichier de travail et un format de présentation.
CANCEL REPORT	Suppression du lien existant entre un fichier d'impression et une description d'état.
CANCEL TITLE	Annulation de l'effet d'une instruction HEADING et/ou FOOTING précédente pour un fichier d'impression.
CHECKPOINT	Exécution d'une consolidation (angl. commitment) de la base de données et constitution d'un point de reprise. Utilisable sous TDS et en traitement par lots uniquement.
COMMIT	Enregistrement des mises à jour effectuées depuis la dernière consolidation (angl.commitment). Utilisable sous IOF et en traitement par lots uniquement.
CONNECT	Rattachement d'une occurrence d'article IDS/II à une occurrence de son article maître.
CREATE	Initialisation du processus de création d'une occurrence d'article pour une base de données.
DEFINE	Définition de variables permanentes ou temporaires ou redéfinition de variables temporaires.
DELETE	Suppression d'une ou plusieurs occurrences d'article dans une base de données.
DISCONNECT	Suppression de la liaison existant entre une occurrence d'article IDS/II et une occurrence de son article maître.
DISPLAY	Visualisation de données au terminal (ou enregistrement des données dans un fichier d'impression traitement par lots).

DO	Définition d'une boucle d'itération.
EJECT	Effacement de l'écran avec positionnement du curseur en début d'écran.
END	Fin de bloc AFTER, BEFORE, CREATE, DO, IF, ON ABSENT, READ ou RETRIEVE.
EXECUTE	Branchement sur une sous-requête.
EXIT	Sortie d'une boucle d'itération introduite par CREATE, DO, READ ou RETRIEVE.
FOOTING	Demande d'impression des titres d'un état en bas de chaque page.
HEADING	Demande d'impression des titres d'un état en haut de chaque page.
IF	Définition d'une condition déterminant l'exécution d'une séquence d'instructions.
INSERT	Insertion d'une nouvelle occurrence d'article dans une base de données.
LET	Affectation d'une valeur à une zone (en mémoire).
MODIFY	Modification du contenu d'une base de données à partir de données en mémoire.
ON ABSENT	Instruction de contrôle déterminant l'exécution d'une séquence d'instructions lorsqu'aucun article n'est délivré par READ ou RETRIEVE.
PARAMETER	Définition des paramètres à fournir à l'exécution d'une requête.
PRINT	Impression d'une ligne d'état conformément à un format de présentation défini par l'utilisateur.
READ	Lecture d'un fichier de travail ou d'une base de données.
RECONNECT	Changement d'occurrence d'article maître pour une occurrence d'article détail IDS/II.

REPEAT	Branchement déterminant un retour à la première instruction d'un bloc CREATE, DO, READ ou RETRIEVE.
REPORT	Edition d'un fichier de travail monoarticle au moyen du générateur d'états.
RESTART	Annulation de toutes les modifications effectuées dans la base de données et relance de la requête au dernier point de reprise. Utilisable sous TDS et en traitement par lots uniquement.
RETRIEVE	Recherche d'articles dans des bases de données définies par un schéma, une vue ou une structure.
RETURN	Sortie d'une requête IQS.
REWIND	Repositionnement en début de fichier ou d'aire pour une opération de lecture.
ROLLBACK	Annulation de toutes les mises à jour effectuées depuis la dernière consolidation. Utilisable sous IOF uniquement.
SORT	Tri d'un fichier de travail ou d'une base de données et rangement des articles triés dans un nouveau fichier.
SPACE	Saut de ligne(s) ou saut à une nouvelle page dans un état.
USE FORMAT	Association d'un format de présentation à un fichier de travail.
USE REPORT	Association d'une description d'état à un fichier d'impression.
WRITE	Ecriture d'un ou plusieurs articles dans un fichier de travail.

1.2 Macros IQS

Les textes d'usage fréquent peuvent être enregistrés sous forme de macros IQS qu'il est ensuite possible d'appeler depuis les requêtes. Ces textes sont modifiables à chaque appel en fournissant des valeurs spécifiques aux paramètres qu'ils contiennent. Les macros sont décrites en détail chapitre 9.

1.3 Générateur d'états IQS

Pour normaliser différents états ayant une présentation similaire, l'utilisateur peut écrire une description d'état, la compiler, puis la sauvegarder, en utilisant le générateur d'états (ou le processeur d'états interactif décrit dans le volume 1). Une fois sauvegardée, elle peut être associée à n'importe quelle requête. Cette description définit les blocs en-tête et fin d'état, en-tête et fin de page, ainsi que l'emplacement des lignes détail dans la page.

Le générateur d'états est décrit en détail au chapitre 8.

1.4 Espaces de travail et bibliothèques

IQS utilise des espaces de travail et des bibliothèques.

1.4.1 Espaces de travail

Au cours d'une session IQS, l'utilisateur dispose automatiquement des deux espaces de travail décrits ci-dessous. Il existe également un espace de travail auquel l'utilisateur n'a pas accès et qui est réservé à l'exécution des scénarios.

1.4.1.1 Espace de travail origine

C'est l'espace de travail dans lequel l'utilisateur crée et modifie le code origine (c'est-à-dire ses requêtes, macros, scénarios et descriptions d'état). Cet espace comporte des lignes identifiées par un numéro. IQS effectue automatiquement la numérotation des lignes et l'affectation de place requise.

L'utilisateur peut introduire des informations (par exemple des instructions du langage de requêtes avec leurs paramètres) sur une ligne, modifier son contenu, insérer une nouvelle ligne ou supprimer une ligne existante. Ces opérations s'effectuent au moyen des commandes de l'éditeur de texte IQS, décrites au chapitre 4.

Un requête peut être chargée dans l'espace de travail origine à partir de la bibliothèque origine (SLLIB), et sauvegardée dans une bibliothèque à partir d'un espace de travail origine. Les requêtes peuvent être compilées à partir de l'espace de travail origine, puis exécutées à partir de l'espace de travail résultant. Ces opérations s'effectuent au moyen des commandes de préparation de programmes IQS et des différentes commandes utilitaires, qui font partie du langage "non procédural" et sont donc décrites dans le volume 1 de ce manuel.

Le contenu de tous les espaces de travail est perdu à la fin de chaque session IQS, sauf si une sauvegarde a été effectuée avant d'arrêter la session.

1.4.1.2 Espace de travail résultant

Pour être exécutable, une requête doit d'abord être compilée. La compilation fournit une requête résultante. Celle-ci peut être enregistrée de manière temporaire dans un espace de travail résultant, ou de manière permanente (sauvegarde) dans une bibliothèque (binaire) résultante. La compilation d'une requête origine ou le chargement d'une requête résultante à partir d'une bibliothèque résultante sont les seules manières d'affecter le contenu de l'espace de travail résultant.

Contrairement à celui de l'espace de travail origine, le contenu de l'espace de travail résultant ne peut pas être manipulé ligne par ligne au moyen des commandes de l'éditeur de texte. Le contenu de cet espace de travail est perdu lorsque (i) le contenu de l'espace de travail origine est modifié (ii) l'utilisation d'une vue est terminée (iii) une session IQS est terminée. Le contenu de l'espace de travail résultant doit donc être sauvegardé s'il est à réutiliser ultérieurement.

1.4.1.3 Espace de travail des scénarios

L'utilisateur IQS n'a pas accès à cet espace de travail qui est réservé à l'exécution des scénarios. Le processeur de scénarios est décrit dans le volume 1 (77UR).

1.4.2 Bibliothèques

Plusieurs bibliothèques sont nécessaires pour contenir les différents éléments (requêtes, schémas, procédures GCL, grilles, etc.) utilisés au cours d'une session IQS. Celles-ci sont de deux types :

- bibliothèques origine (type SL)
- bibliothèques binaires (type BIN).

Le nombre requis de bibliothèques de chaque type dépend de l'utilisateur qui pourra, par exemple, souhaiter séparer ses requêtes résultantes de son GCL.

Une seule bibliothèque de chaque type peut suffire, mais il est généralement recommandé d'utiliser au moins trois bibliothèques binaires (voir plus loin).

1.4.2.1 Bibliothèque origine

La bibliothèque origine (SLLIB) contient les éléments suivants :

- Schémas origine (DDL)
- Requêtes, macros, scénarios et descriptions d'états origine.

Le contenu de l'espace de travail origine étant perdu à la fin de chaque session, la bibliothèque origine (SLLIB) peut servir à le sauvegarder pour utilisation ultérieure. Un nom est attribué au contenu de l'espace de travail et celui-ci est sauvegardé sous la forme d'une unité de bibliothèque portant son nom.

Lors d'une session IQS suivante, cette unité de bibliothèque peut être rechargée dans l'espace de travail origine au moyen d'une commande LOAD. Le contenu de l'espace de travail origine peut alors être modifié en utilisant les commandes de l'éditeur de texte IQS.

Si la modification est temporaire, il n'y a pas lieu d'effectuer une sauvegarde. Si la modification doit être permanente, le nouveau contenu devra être sauvegardé dans la bibliothèque origine sous l'ancien nom (valeur implicite) si l'ancien contenu n'est plus nécessaire (au moyen d'une commande REPLACE) ou bien sous un nouveau nom (au moyen d'une commande SAVE).

1.4.2.2 Bibliothèques binaires

Les bibliothèques binaires sont les suivantes :

- BINLIB, qui contient les objets compilés suivants : requêtes, macros, descriptions d'état et formats de présentation.
- DDLIBn (n = 1 à 3), qui contient les schémas résultants de type DD et les schémas résultants IQS de type SDD produits par la commande COMPILE SCHEMA. DDLIB1 contient également les structures de type FDD produites par les commandes DEFINE STRUCTURE et UPDATE STRUCTURE ainsi que les vues de type VDD produites par les commandes DEFINE VIEW et UPDATE VIEW.
- CULIB, qui contient les requêtes résultantes compilées utilisables sous TDS.
- #BLIB et #BINLIBn (n = 1 à 3), qui contiennent les grilles utilisateur ainsi que les procédures GCL servant à lancer les sessions IQS.

BINLIB est la bibliothèque de sauvegarde de l'espace de travail résultant de la même manière que SLLIB pour l'espace de travail origine. Si le contenu de l'espace de travail résultant est à réutiliser ultérieurement, il doit être sauvegardé dans BINLIB.

En fait, les différentes commandes SAVE disponibles pour la préparation de programmes permettent une sauvegarde simultanée des espaces de travail origine et résultant.

Pour éviter tout risque de décalage entre version résultante et version origine des requêtes à l'exécution (version origine modifiée après compilation de la version résultante), le système invalide automatiquement les unités de bibliothèque contenant des versions résultantes périmées.

Un jeu spécial de commandes IQS permet de sauvegarder, supprimer, lister ou remplacer les unités des bibliothèques résultantes. Pour plus de détails, se reporter au volume 1 (77UR).

Des informations sur les bibliothèques et leur affectation pour une session IQS sont fournies dans le guide de l'administrateur IQS-V4 (80UR).

1.4.2.3 Types d'objets des bibliothèques IQS

Tableau 1-1. Types d'objets des bibliothèques IQS

BIBLIOTHEQUES	UNITES DE BIBLIOTHEQUES	TYPE D'OBJETS
DDLIB DDLIB1	schéma (DDL) schéma_D (IQS) vue_D (IQS) structure_D (IQS)	DD SDD VDD FDD
BINLIB	requête résultante macro résultante descr.-état-résultante format	QRY QRM QRP FMT
BLIB	grille_D grille_OF_terminal grille_SF	RDD FRM FRM
SLLIB	requête origine macro origine description-état-origine scénario	QRY QRM QRP SCR
CULIB	requête compilée IQS/TDS	QRY

REMARQUES :

1. Le type de langage DD s'obtient en compilant un schéma DDL avec DDLPROC.
2. En compilant un schéma DDL avec un processeur MNDD, les différents types de langage obtenus sont les suivants :
   - DD2 pour un schéma IDS II,
   - DD4 pour un sous-schéma IDS II,
   - DD3 pour un schéma de fichier UFAS.

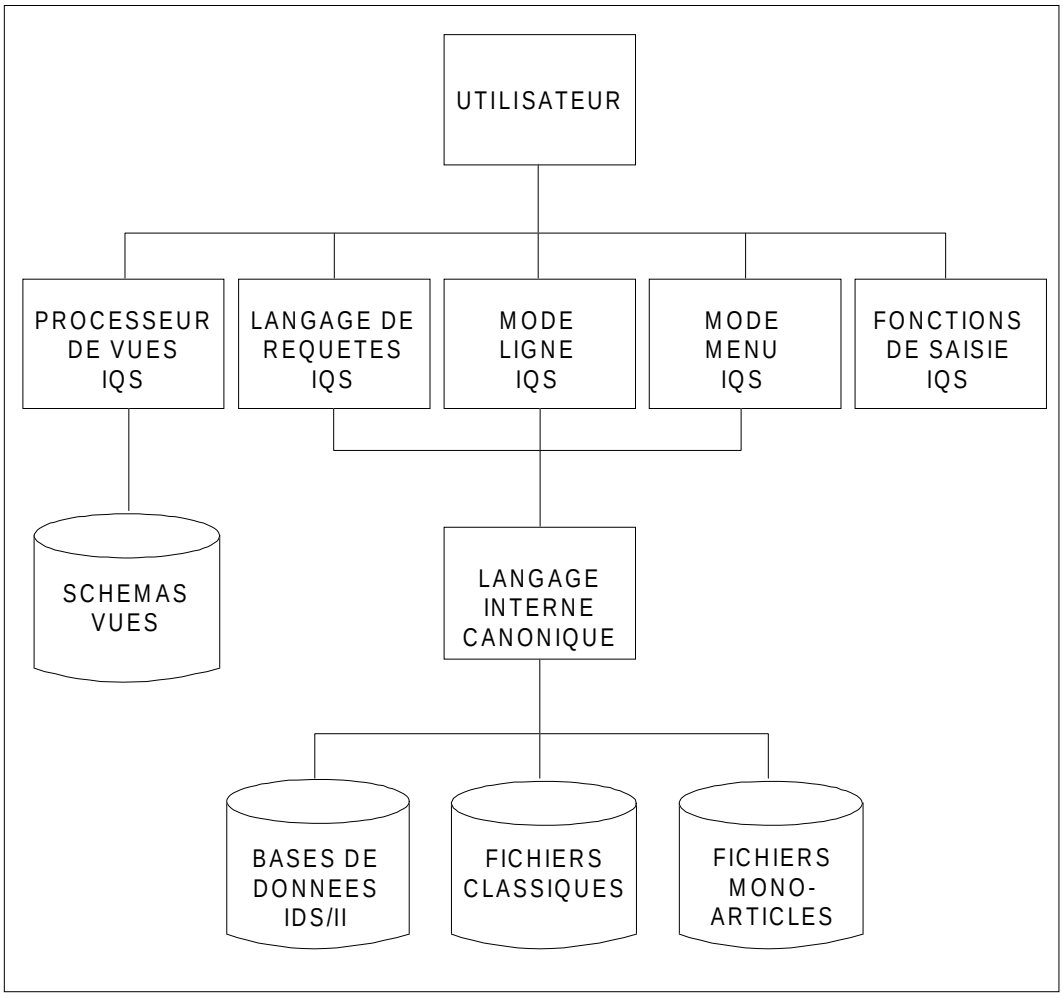


Figure 1-1. Architecture IQS

2. Eléments du langage de requêtes

Ce chapitre décrit les éléments composant les instructions du langage de requêtes.

Bien que la plupart des informations données ici soient également valables pour les commandes IQS, il existe de légères différences de syntaxe entre langage procédural et non procédural. Il est donc recommandé de se reporter au volume 1 du manuel de référence IQS-V4 pour la syntaxe des commandes.

2.1 Présentation du langage

Une instruction se compose d'un verbe suivi de mots-clés et d'éléments combinés selon les règles exposées au début du chapitre 5, les instructions proprement dites étant décrites dans les chapitres 5 à 7.

2.1.1 Mot IQS

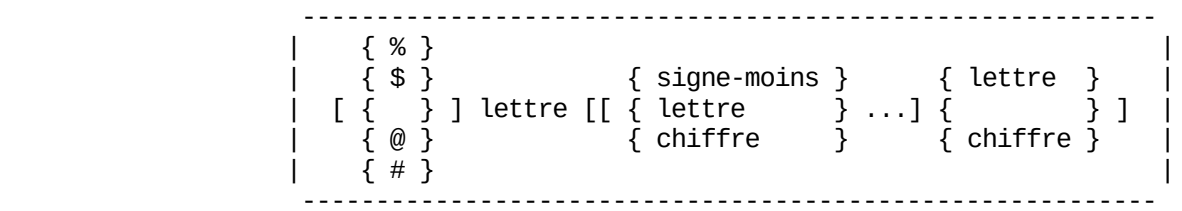
Tous les éléments ci-dessous sont des mots IQS.

```
 ----------------------------
|  { nom                   } |
|  { littéral              } |
|  { constante-hexadécimale } |
|  { constante-numérique   } |
|  { séparateur-1          } |
|  { séparateur-2          } |
|  { séparateur-3          } |
|  { appel-de-macro-IQS    } |
|  { clause-PICTURE        } |
|  { élément-non-standard  } |
 ----------------------------
```

Les mots IQS sont décrits dans les paragraphes suivants.

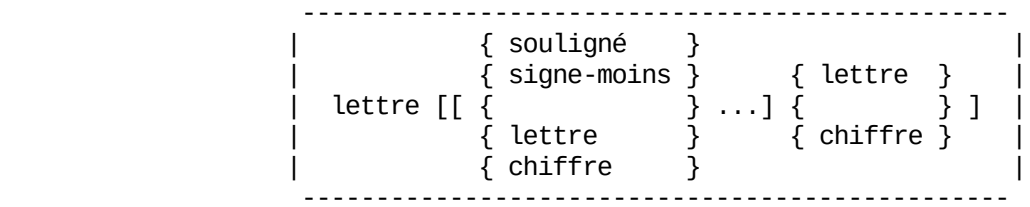
- Le mot IQS est la plus petite unité syntaxique du langage de requêtes.
- Chaque instruction est constituée de mots IQS.
- La syntaxe des appels de macros IQS est décrite au chapitre 9.

2.1.2 Nom



- Tout caractère qui est un séparateur termine un nom.
- Le signe moins (-) appartient au nom lorsqu'il est suivi d'une lettre ou d'un chiffre.
- Un nom ne doit pas excéder 30 caractères.
- Les noms des différents types d'éléments sont décrits plus loin dans ce chapitre. Ce sont :
- les noms base de données
- les noms de zones de données
- les noms de zones variables temporaires
- les noms de zones variables système
- les noms de paramètres.

2.1.3 Nom de fichier logique



Un nom de fichier logique ne doit pas excéder 30 caractères.

2.1.4 Lettre

Une lettre est l'un des caractères A à Z et a à z.

IQS traite les minuscules dans les noms comme s'il s'agissait de majuscules. Elles ne sont interprétées comme minuscules que lorsqu'elles apparaissent entre guillemets, c'est-à-dire dans un littéral.

2.1.5 Souligné

Le caractère souligné _ est utilisé dans les noms de fichiers logiques.

2.1.6 Signe moins

Il s'agit du caractère -. Il est utilisé comme tiret (dans les identificateurs) et comme signe moins (dans les expressions). Il est également employé comme caractère suite en fin de ligne lors de l'introduction ou de la modification du contenu de l'espace de travail origine et lors de l'introduction de commandes en mode ligne. A noter que lorsque ce caractère est utilisé comme opérateur numérique (signe moins), il doit être précédé et suivi d'un espace.

2.1.7 Chiffre

Un chiffre est l'un des caractères 0 à 9.

2.1.8 Littéral

```
|--------------------------------------|
|  [(entier)] "chaîne de caractères"   |
|--------------------------------------|
```

- Littéral est synonyme de chaîne protégée.
- L'entier, facultatif, indique combien de fois doit se répéter la chaîne de caractères. La valeur 1 n'a donc pas d'effet.
- La longueur d'un littéral ne doit pas excéder 255 caractères.

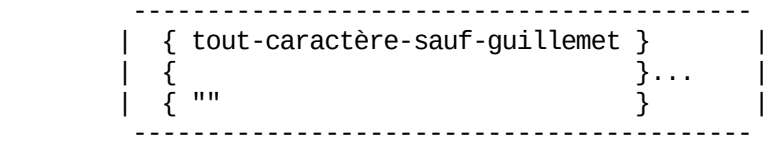
2.1.9 Constante hexadécimale

```
|-------------------------------------------------|
|"{caractère-hexadécimal caractère-hexadécimal} ..."X |
|-------------------------------------------------|
```

La longueur maximum d'une constante hexadécimale est de 254 caractères hexadécimaux.

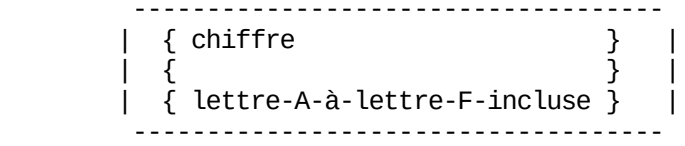
2.1.10 Entier

2.1.11 Chaîne de caractères

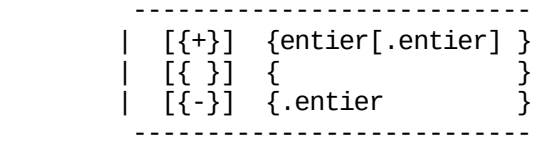


- Lorsqu'un guillemet (") fait partie intégrante d'une chaîne de caractères, il doit être doublé. Le guillemet doublé ("") ne compte que pour un caractère dans la longueur de la chaîne.
- La longueur maximum d'une chaîne est de 255 caractères.

2.1.12 Caractère hexadécimal



2.1.13 Constante numérique



- Une constante numérique ne doit pas contenir plus de 30 chiffres.
- Lorsque la valeur spécifiée pour la constante ne comporte pas de point décimal, le système l'enregistre automatiquement sous forme binaire.
- Le point (.) n'est pas un séparateur-1 lorsqu'il est utilisé dans une constante numérique.

2.1.14 Séparateur-1

```
 --------------------------------
|  { signe-plus             + }  |
|  { astérisque             * }  |
|  { barre-oblique          / }  |
|  { virgule                , }  |
|  { inférieur-à            < }  |
|  { supérieur-à            > }  |
|  { égal                   = }  |
|  { accent-circonflexe     ^ }  |
|  { parenthèse-gauche      ( }  |
|  { parenthèse-droite      ) }  |
|  { point                  . }  |
 --------------------------------
```

- Un séparateur-1 n'a pas à être obligatoirement précédé ou suivi d'un espace.
- Le point (.) n'est pas un séparateur-1 lorsqu'il est utilisé dans une constante numérique.

2.1.15 Séparateur-2

```
 ------------------------------------
|  { crochet-gauche            [ }  |
|  { crochet-droit             ] }  |
|  { point-d'exclamation       ! }  |
|  { point-virgule             ; }  |
|  { point-d'interrogation     ? }  |
|  { deux-points               : }  |
|  { apostrophe                ' }  |
 ------------------------------------
```

- Un séparateur-2 n'a pas à être obligatoirement précédé ou suivi d'un espace.
- Un séparateur-2 ne peut être utilisé que dans un appel de macro IQS ou dans la spécification d'appel de macro d'une définition de macro IQS.

2.1.16 Séparateur-3

```
|------------------|
|   signe-moins    |
|------------------|
```

Un signe moins terminant un identificateur ne doit pas être suivi d'une lettre ou d'un chiffre.

2.1.17 Elément non standard

Un élément non standard est une suite de caractères ne répondant à aucune des définitions ci-dessus (par exemple A$1 ou $$X_e).

Un tel élément n'est valide que lorsqu'il est utilisé dans la spécification d'appel de macro d'une définition de macro IQS et dans les appels de macro IQS correspondants (voir chapitre 9).

2.1.18 Séparateur-4

```
 ----------------------------
|  { et commercial      & }  |
|  { espace               }  |
|  { fin-de-ligne         }  |
|  { commentaire          }  |
 ----------------------------
```

2.1.19 Commentaire

```
|-----------------------------------------------------|
|  /* tout-caractère-sauf-astérisque-suivi-de-barre   |
|  -oblique-ou-l'inverse */                           |
|-----------------------------------------------------|
```

- Ce format permet d'insérer des commentaires dans une requête. Ceux-ci n'affectent en rien son exécution. Ils apparaissent chaque fois que la requête origine est imprimée et permettent, par exemple, d'expliquer ce que fait la requête.
- Un commentaire est délimité par /* (au début) et */ (à la fin). Il peut être inséré partout où un espace peut l'être. Il est syntaxiquement équivalent à un espace et peut donc être placé entre des paramètres et entre des mots-clé et des paramètres.
- Un commentaire ne doit pas être placé à l'intérieur d'un nom d'instruction ou de commande, d'un mot-clé, d'une constante numérique ou d'un nom de paramètre. Lorsqu'ils se trouvent à l'intérieur d'un littéral (entre guillemets), les délimiteurs de commentaire ne sont pas reconnus en tant que tels et sont traités comme partie intégrante du littéral.

EXEMPLE 1 :

```
RETRIEVE AB/* COMMENTAIRE VALIDE */ WHERE ...

/* DEBUT BOUCLE DE RECHERCHE */
..
..
..
END
```

❑

EXEMPLE 2 :

```
/* LISTE DU PERSONNEL AYANT PLUS DE 5 ANS */
/* D'EXPERIENCE MAIS FAISANT PARTIE DE LA */
/* COMPAGNIE DEPUIS MOINS DE 2 ANS */

RETRIEVE AB WHERE ...
..
..
END
```

❑

2.2 Noms

2.2.1 Noms base de données

Ce sont des noms qui peuvent être utilisés dans une requête mais qui ne désignent pas des zones d'article, à savoir :

```
nom-d'aire
nom-d'ensemble
```

Les noms base de données ne peuvent pas contenir de caractères souligné (_).

2.2.2 Autres noms

Ils peuvent se composer des caractères suivants : lettres, chiffres, tiret et souligné. Ces noms sont les suivants :

```
nom-de-description-d'état
nom-de-grille
nom-de-fichier-d'impression
nom-de-fichier-de-travail
nom-de-requête
nom-de-macro
nom-de-scénario
nom-de-format
```

2.3 Zones de données

```
|-----------------------------------------------------|
|  nom-de-zone-de-données [indice] [qualification]    |
|-----------------------------------------------------|
```

Les zones d'un article sont définies dans le schéma de la base de données.

Il est possible d'obtenir des informations sur ces zones soit en consultant la personne compétente, soit (si l'on dispose des droits d'accès nécessaires) en utilisant la commande IQS PRINT SCHEMA (en traitement par lots) ou DISPLAY SCHEMA (en interactif).
De même, les commandes PRINT VIEW/DISPLAY VIEW et PRINT STRUCTURE/DISPLAY STRUCTURE permettent d'obtenir des informations sur les zones contenues dans une vue ou dans une structure.

En général, les noms de zone de données doivent être uniques ; pour les exceptions, se reporter au paragraphe "Qualification".

2.4 Zones variables temporaires

```
|-------------------------------------------------|
| nom-de-variable-temporaire [indice] [qualification] |
|-------------------------------------------------|
```

Une variable temporaire est interne à la requête courante. Une fois celle-ci terminée, elle est perdue. A noter qu'une variable temporaire peut être une variable fichier ; dans ce cas, elle est transmissible à une commande IQS. Pour plus de détails, se reporter se reporter au volume 1 (77UR).

Une variable temporaire peut être définie implicitement en plaçant devant son nom le préfixe @, $ ou #. Ces préfixes déterminent les définitions implicites suivantes :

@ -->	CHARACTER 32 (caractère, longueur 32)
$ -->	BINARY 31 (binaire, longueur 31)
# -->	PACKED SIGNED (15,2) (condensé signé, longueur 15,2)

Elle peut également être définie explicitement au moyen de l'instruction DEFINE. Dans ce cas, le préfixe n'est pas nécessaire.

Les mots réservés ne peuvent pas être utilisés comme noms de variables temporaires.

La longueur totale de toutes les variables temporaires (définies implicitement et explicitement) ne doit pas excéder 32767. Leur nombre total ne doit pas excéder 1000.

EXEMPLE :

```
C:  ASSIGN WORK1
C:  AUTO
    .....
    ....
    WRITE #A TO WORK1
    .....
    .....
C:  GO
C:  FILE WORK1
F:  FORMAT
R:  ITEM #A JUST CENTER
❑
```

Ecriture d'une variable temporaire (#A) dans un fichier de travail par une requête. Cette variable est traitée comme une variable fichier par la commande IQS suivante et est mise en forme sous le processeur de formats standard.

2.5 Paramètres

```
 ------------------------------------------------------
|  { &n                                                |
|  {                                                   |
|  { nom-de-paramètre   [indice]   [qualification] }   |
 ------------------------------------------------------
```

A noter que n doit être compris entre 1 et 16.

Un paramètre permet de :

- conserver des valeurs d'une requête à l'autre au moyen d'une instruction EXECUTE,
- fournir des valeurs à une requête au moyen de la commande IQS EXEC.

Un paramètre peut être défini implicitement ou explicitement.

Il est défini implicitement au moyen du caractère & suivi d'un nombre entier compris entre 1 et 16. Les espaces sont autorisés entre le et commercial (&) et le nombre.

La définition implicite peut être utilisée pour le verbe RETRIEVE ou la commande RETAIN.

Dans ce cas, le paramètre doit constituer le deuxième terme d'une condition simple, le premier terme devant être un scalaire (c'est-à-dire une zone qui ne soit ni du type groupe, ni du type structure). Il adopte alors automatiquement la définition de cette zone. Lorsqu'un paramètre défini implicitement est utilisé plusieurs fois, il conserve la même définition.
Un paramètre est défini explicitement au moyen de l'instruction PARAMETER. Dans ce cas, il peut être préfixé de @, $ ou # (se reporter au paragraphe précédent).

Des paramètres implicites et explicites ne doivent pas être utilisés dans la même requête. Les mots réservés ne peuvent pas être utilisés comme noms de paramètres explicites.

Les paramètres sont également utilisés dans la commande IQS RETAIN. Ils permettent de définir de manière interactive les conditions à satisfaire dans des commandes CHANGE, EXTRACT, PRINT, REVIEW, STATISTICS et WRITE ultérieures.

EXEMPLE DE DÉFINITION IMPLICITE :

```
RETRIEVE CLIENTS
  WHERE CL-CODE = &1
  DISPLAY CLIENT
END
```

❑

Le paramètre &1 adopte la définition de la zone CL-CODE.

2.6 Qualification

```
 ---------------------------------------
|       { nom-de-fichier              }  |
|       { nom-d'aire                  }  |
|  OF   { nom-d'article               }  |
|       { nom-de-zone-structure       }  |
|       { nom-de-zone-groupe [indice] }  |
 ---------------------------------------
```

Un nom de zone de données spécifié explicitement doit être unique. Si plusieurs noms identiques figurent dans le contexte d'une même instruction, ils doivent être qualifiés au moyen de la clause OF.

EXEMPLES :

```
N-ANNEE OF DATE-NAISSANCE
N-JOUR OF DATE-NAISSANCE OF ETUDIANT
❑
```

Dans le deuxième exemple, N-JOUR fait partie de DATE-NAISSANCE qui est une zone de type structure appartenant elle-même à l'article ETUDIANT.

En ce qui concerne la visibilité des noms et donc l'utilisation de la qualification, deux cas sont à distinguer : celui des instructions CREATE, READ et RETRIEVE et celui des autres instructions.

1. Instructions autres que CREATE, READ et RETRIEVE

   Seuls les noms des variables système, des paramètres et des variables temporaires sont connus ; les noms de zones de données sont inconnus, sauf dans le cas de l'instruction DEFINE...IDEM où il est possible de spécifier le nom d'un article de la vue courante.

2. Instructions CREATE, READ et RETRIEVE

   Les noms des zones appartenant aux articles spécifiés dans les instructions CREATE, READ et RETRIEVE, qu'il s'agisse des instructions courantes ou d'instructions de niveau supérieur dans lesquelles sont emboîtées les instructions courantes, sont connus. Dans ce cas, si une variable temporaire porte le même nom qu'une zone de données, son nom est masqué par celui de la zone de données jusqu'à ce que soit détecté le mot-clé END de l'instruction considérée.

2.7 Indicage

```
|---------------------------|
|   (expression-numérique)  |
|---------------------------|
```

L'indiçage permet d'identifier un élément particulier dans une liste ou une table d'éléments semblables, par exemple une zone de données, une variable temporaire ou un paramètre dans un groupe ou vecteur, ou encore une structure dans un groupe.

Un indice est une expression numérique dont la valeur est entière (binaire). Consulter également le paragraphe "Expressions" plus loin dans ce chapitre.

La valeur la plus basse d'un indice est 1 ; cette valeur renvoie au premier élément du vecteur ou du groupe.

2.8 Fichiers base de données

2.8.1 Fichiers base de données sous IQS

Pour l'utilisateur IQS, un **fichier base de données** est une collection de données pour laquelle un schéma a été défini. Il peut s'agir d'un fichier UFAS ou d'un fichier IDS/II.

En principe, l'utilisateur n'a pas à se préoccuper des fichiers base de données ; il ne traite que des articles, des ensembles et des aires (voir ci-dessous). Il peut également utiliser des fichiers "classiques", c'est-à-dire des fichiers de type UFAS pour lesquels aucun schéma n'a été défini. Ceux-ci sont présentés plus loin.

L'utilisation de fichiers UFAS sous IQS entraîne certaines restrictions : un schéma IQS défini pour un fichier séquentiel ne peut avoir que deux types de structure : non hiérarchique (sans liaisons) ou hiérarchique (arborescente ou linéaire) ; de même un schéma défini pour un fichier séquentiel indexé ne peut avoir que ces deux types de structure, mais l'accès direct aux articles est possible. Un schéma défini pour un fichier relatif, par contre, ne peut pas avoir une structure hiérarchique. Des exemples de schémas sont donnés à l'annexe A et l'accès aux différents types de fichier est expliqué au chapitre 7 (instruction RETRIEVE).

2.8.2 Articles

Dans la terminologie IQS, le terme "article" a deux significations : il désigne un agrégat de données soit physique, soit logique. L'utilisateur IQS travaille à la fois sur des articles "réels" et sur des articles "logiques". Ces deux types d'article sont décrits ci-après.

2.8.3 Articles réels

Un **article réel** est un article au sens classique du terme, c'est-à-dire un agrégat de données se rapportant au même sujet et physiquement contiguës. Il se compose d'une ou de plusieurs zones. Dans la représentation des schémas, un article réel est symbolisé par un rectangle. Par exemple, dans la figure A-1 de l'annexe A, UNIVERSITE est un article réel, de même que COLLEGE et ETUDIANT.

2.8.4 Articles logiques

Un **article logique** est un agrégat virtuel de données constitué à partir d'articles réels. Il se compose d'une ou de plusieurs zones logiques. L'accès à un article logique, que ce soit par un utilisateur ou par une requête, se fait par l'intermédiaire d'une vue "relationnelle" précédemment définie. Les sous-commandes du processeur de vues sont fournies dans le volume 1 de ce manuel (77UR). La création des articles logiques est décrite dans le guide de l'administrateur IQS-V4 (80UR).

Par exemple, l'instruction RETRIEVE COLLEGE,ETUDIANT constitue un article logique comprenant les zones de l'article COLLEGE suivies des zones de l'article ETUDIANT. Le contenu de cet article logique peut être manipulé à l'intérieur du bloc RETRIEVE.

2.8.5 Ensembles et noms d'ensembles, articles maître et articles détail

Les articles réels sont reliés logiquement par des pointeurs internes symbolisés par des flèches dans les diagrammes. Ces liaisons déterminent des **ensembles** auxquels sont attribués des noms permettant d'y accéder. Un article peut être l'**article maître** d'autres articles (par exemple, UNIVERSITE est l'article maître de COLLEGE) qui sont alors ses **articles détail** (COLLEGE est donc l'article détail de UNIVERSITE). Se reporter à l'annexe A.

Un **nom d'ensemble** regroupe les désignations de l'article maître et de l'article détail de manière à mettre en évidence la hiérarchie ; dans la figure A-1, U-C est un nom d'ensemble de même que C-D et C-E. Les articles logiques peuvent également être reliés de la même manière pour constituer des ensembles.

2.8.6 Aires

Les schémas, les vues et les structures se composent d'**aires** qui peuvent être ouvertes (commande OPEN) ou fermées (commande CLOSE).
Les aires permettent de désigner les fichiers physiques dans une base de données. Chaque aire (ou fichier physique) peut contenir plusieurs types d'articles. Par exemple, dans la base de données SCH-CUSTOMERS souvent utilisée dans la documentation IQS, l'aire A-ORDERS contient les articles ORDERS et ORDER-LINES. Ces articles comportent des zones distinctes et une zone "type" qui permettent de les différencier, mais ils font partie du même fichier physique.

2.9 Fichiers classiques

Un fichier classique est un fichier pour lequel aucun schéma n'a été défini.

2.9.1 Fichiers de travail

Les fichiers de travail sont créés soit au moyen des instructions du langage de requêtes WRITE ou SORT, soit au moyen des commandes IQS EXTRACT, WRITE, COPY, MERGE ou SORT, soit au moyen des fonctions de saisie.

En général, ces fichiers sont temporaires (c'est-à-dire qu'ils sont perdus à la fin de la session IQS au cours de laquelle ils ont été créés). Ils peuvent cependant être rendus permanents au moyen d'une instruction ASSIGN ou d'une commande IQS ASSIGN. Un fichier de travail permanent peut être traité par un programme écrit dans un autre langage (en COBOL, par exemple). Si sa structure a été sauvegardée au moyen d'une commande IQS SAVE STRUCTURE, il peut servir de "modèle" pour des fichiers définis dans des sessions ultérieures (au moyen des commandes ASSIGN et USE STRUCTURE par exemple).

Les fichiers et les structures créés de manière procédurale au moyen du langage de requêtes peuvent ensuite être utilisés interactivement par des commandes IQS et vice versa.

2.9.2 Fichiers d'impression

Un fichier d'impression est un fichier contenant des informations destinées à être imprimées ou visualisées. L'instruction PRINT...TO... permet d'écrire des données dans un fichier d'impression ayant préalablement été affecté.

L'instruction REPORT du langage de requêtes fait appel au générateur d'états pour mettre en forme le contenu d'un fichier de travail et le placer, si nécessaire, dans un fichier d'impression.

A noter que l'ouverture d'un fichier d'impression s'effectue en mode adjonction (APPEND) et non en mode sortie (OUTPUT). Autrement dit, si le fichier n'est pas vide, les données sont ajoutées en fin de fichier, sans risque de recouvrement.

2.10 Etiquettes

Une étiquette est un nom fourni par l'utilisateur qui comporte au maximum 30 caractères (lettres, chiffres et tiret). Il doit contenir au moins un caractère alphabétique. Une étiquette peut être placée devant une instruction RETRIEVE, CREATE, READ, DO, PRINT ou IF, auquel cas elle doit être suivie d'un point, ou après une instruction REPEAT, EXIT, END, HEADING ou FOOTING sans point. Un mot réservé ne doit pas être utilisé comme étiquette.

Des exemples d'utilisation sont fournis dans le guide du programmeur IQS-V4 (79UR).

2.11 Expressions

2.11.1 Types d'expression

Le terme **expression** est utilisé tout au long de ce manuel. Il désigne une expression alphanumérique ou numérique.

Une expression alphanumérique peut être :
une variable alphanumérique
une zone de type groupe
une zone de type structure
une constante numérique
un littéral
une constante hexadécimale
une expression de type chaîne
une expression de conversion

Une expression numérique peut être :
une variable numérique
une constante numérique
une expression arithmétique

Selon le contexte, une **variable** peut être :
une zone de données
une variable temporaire
une variable système
un paramètre

Une expression numérique avec un entier pour résultat ne peut contenir que des variables binaires et/ou des entiers (qui servent à calculer les indices, les expressions SUBSTRING, READ ou RETRIEVE n TIMES, etc.)

2.11.2 Expressions de type chaîne

Il existe six **expressions de type chaîne** :

- CONCATENATE
- INDEX
- LENGTH
- LOWER
- SUBSTRING
- UPPER

Elles sont décrites ci-après.

2.11.2.1 CONCATENATE (CONC)

Fonction :

CONCATENATE (abréviation CONC) permet de concaténer des expressions alphanumériques.

Format :

```
 ____________________________________________________________
|                                                            |
| CONC[ATENATE]  (<expression-alphanumérique> [SHORT],       |
|                                                            |
|                 <expression-alphanumérique> [SHORT]        |
|                                                            |
|                 [,<expression-alphanumérique> [SHORT]] ...)|
|____________________________________________________________|
```

Description :

La valeur obtenue est une expression alphanumérique dont la longueur est égale à la somme des longueurs des expressions concaténées.

Paramètres :

expression-alphanumérique Expression alphanumérique à concaténer.

SHORT Option permettant de supprimer les espaces à droite.

EXEMPLE :

```
LET @B = "ABC"
LET @C = "DEF"
LET @A = CONCATENATE (@B SHORT,@C SHORT,"GHI")
```
❑

Donne :

```
A : ABCDEFGHI
```

2.11.2.2 INDEX

Fonction :

Cette fonction fournit la position relative d'une expression alphanumérique dans une autre.

Format :

```
|-----------------------------------------------------------|
| INDEX (<expression-alphanum-1>, <expression-alphanum-2>) |
|-----------------------------------------------------------|
```

Description :

La valeur obtenue est de type BINARY 31.
Elle indique la position début de expression-alphanumérique-2 dans expression-alphanumérique-1 ou est égale à zéro si la deuxième expression ne se trouve pas dans la première. Les espaces à droite de la deuxième expression sont ignorés ; lorsque expression-alphanumérique-2 ne contient que des espaces, c'est la position du premier espace (éventuel) trouvé dans expression-alphanumérique-1 qui est indiquée.

Paramètres :

expression-alphanum-1	Expression contenant expression-alphanumérique-2
expression-alphanum-2	Expression à trouver dans expression-alphanumérique-1

EXEMPLE :

```
RETRIEVE ...
  WHERE INDEX (CL-NOM, "G") EQ 4
  ...
END
```
❑

Recherche de toutes les occurrences de la zone CL-NOM où "G" se trouve en quatrième position.

2.11.2.3 LENGTH

Fonction :

Cette fonction fournit la longueur d'une expression alphanumérique sans les espaces à droite.

Format :

```
|----------------------------------------|
|   LENGTH (<expression-alphanumérique>)   |
|----------------------------------------|
```

Description :

La valeur obtenue est de type BINARY 31.

Paramètre :

expression-alphanumérique Expression dont la longueur est à fournir.

EXEMPLE :

```
DEFINE @A CHAR 20
LET @A = "ABCDEF"
DISPLAY LENGTH (@A)
```
❑

Donne : 6

Cette valeur représente la longueur de la variable temporaire @A.

2.11.2.4 LOWER

Fonction :

Cette fonction convertit les majuscules d'une expression alphanumérique en minuscules.

Format :

```
|---------------------------------------------|
|    LOWER (<expression-alphanumérique>)      |
|---------------------------------------------|
```

Description :

La valeur obtenue est une expression alphanumérique dont la longueur est égale à celle de l'expression à traiter.
Cette fonction ne convertit pas l'expression alphanumérique, mais renvoie une nouvelle expression utilisable dans une affectation, une comparaison ou autre fonction.

Paramètres :

expression-alphanumérique Expression à traiter.

EXEMPLE :

```
LET @A = "ABCD"
DISPLAY LOWER (@A)
❑
```

Donne :

```
A : abcd
```

mais la valeur reste en majuscules dans @A.

2.11.2.5 SUBSTRING (SUBSTR)

Fonction :

Cette fonction permet d'accéder à une partie (sous-chaîne) d'une expression alphanumérique.

Format :

```
|-------------------------------------------------------------|
|SUBSTR[ING] (<expression-alphanumérique>, <caractère-début>, |
|            <longueur>)                                      |
|-------------------------------------------------------------|
```

Description :

Lorsque SUBSTRING est utilisé comme terme de gauche dans une affectation, la sous-chaîne commençant à la position <caractère-début> est remplacée par l'expression alphanumérique spécifiée comme terme de droite, la longueur de la sous-chaîne étant conforme à celle spécifiée dans le format.

EXEMPLE :

```
LET @A = "ABCDEFGHI"
LET SUBSTR(@A,2,4) = "XXX"
DISPLAY @A
❑
```

Donne :

```
A : AXXX FGHI
```

Lorsque SUBSTRING est utilisé comme terme de droite dans une affectation, la valeur obtenue est de type CHARACTER. La sous-chaîne correspondante commence à la position <caractère-début> et sa longueur est :

- soit conforme à celle spécifiée dans le format si <longueur> est une constante numérique ;
- soit égale à la longueur de l'expression alphanumérique spécifiée si <longueur> n'est pas une constante numérique.

EXEMPLE :

```
DEFINE @A CHAR 9
DEFINE @B CHAR 100
LET @A = "ABCDEFGHI"
LET $L = 4
LET @B = CONC(SUBSTR(@A,2,4),"XXX")
DISPLAY @B
LET @B = CONC(SUBSTR(@A,2,$L),"XXX")
DISPLAY @B
```

❑

Donne :

```
B : BCDEXXX
B : BCDE     XXX
```

Paramètres :

expression-alphanumérique

Expression alphanumérique de laquelle est à extraire la sous-chaîne. A noter que lorsque SUBSTRING est utilisé comme terme de gauche dans LET ou MODIFIY, l'expression ne peut contenir qu'un nom de zone alphanumérique.

caractère-début

Il indique la position dans l'expression du premier caractère de la sous-chaîne, en partant de la gauche. Il s'agit d'une expression numérique fournissant un entier. Sa valeur minimum est 1.

longueur

Il indique la longueur de la sous-chaîne. Il s'agit d'une expression numérique fournissant un entier, de valeur minimum 1 et de valeur maximum égale à la longueur totale de l'expression alphanumérique.

2.11.2.6 UPPER

Fonction :

Cette fonction convertit les minuscules d'une expression alphanumérique en majuscules.

Format :

```
|--------------------------------------|
|  UPPER (<expression-alphanumérique>)  |
|--------------------------------------|
```

Description :

La valeur obtenue est une expression alphanumérique de même longueur que l'expression spécifiée.

REMARQUE :

Cette fonction ne convertit pas l'expression alphanumérique, mais renvoie une nouvelle expression utilisable dans une affectation, une comparaison ou autre fonction.

Paramètre :

expression-alphanumérique Expression à traiter.

EXEMPLE :

```
LET @A = "abcd"
DISPLAY UPPER (@A)
❑
```

Donne :

```
A : ABCD
```

mais la valeur reste en minuscules dans @A.

2.11.3 Expressions de conversion

Il existe deux **expressions de conversion** :

HEXA
VALUE

Elles sont décrites ci-après.

2.11.3.1 HEXA

Fonction :

Cette fonction permet d'obtenir la valeur hexadécimale d'une variable sous forme alphanumérique.

Format :

```
|------------------|
|  HEXA (<zone>)   |
|------------------|
```

Paramètre :

zone	Ce peut être une zone de n'importe quel type (y compris de type structure ou groupe). La longueur de la chaîne résultante est égale au nombre d'octets occupés par la zone multiplié par deux. La longueur maximum résultante est de 254 caractères.

EXEMPLE :

```
HEXA ($i)
❑
```

Si la valeur de la zone $i, définie par BINARY 15, est 3, le résultat obtenu est : 0003.

2.11.3.2 VALUE

Fonction :

Cette fonction permet de redéfinir dynamiquement les attributs d'une zone. Elle doit constituer la partie droite d'une instruction LET.

Format :

```
|-----------------------------------------------------|
|VALUE (<zone>, <type>, <longueur> [,<nb-décimales>]) |
|-----------------------------------------------------|
```

Paramètres :

zone	Nom de la zone à redéfinir en partie ou en totalité.
type	Nouveau type de la zone, à savoir : 0 = décimal éclaté signé 1 = décimal condensé signé 2 = décimal éclaté non signé 3 = décimal condensé non signé (PACKED-2 en DDL) 4 = décimal condensé non signé du DDL 5 = binaire 6 = alphanumérique
longueur	Longueur de la zone à redéfinir.
nb-décimales	Nombre de chiffres après la virgule (le cas échéant) de la zone. La valeur implicite est 0.

REMARQUE :

Se reporter au tableau 2-12 pour contrôler la correspondance des types entre IQS, DDL et COBOL.

Exemple :

La variable @INTERFACE est définie par CHARACTER 255 ; elle contient une valeur binaire dans les quatres premières positions. Pour obtenir cette valeur, procéder comme suit :

```
LET $I = VALUE (@INTERFACE, 5, 31)
```

5 étant le type de la zone (BINARY) et 31 sa longueur.

2.11.4 Expressions arithmétiques

Une **expression arithmétique** équivaut à une formule algébrique ; elle comporte des opérandes (zones ou constantes numériques) et des opérateurs, dont la combinaison détermine l'opération à effectuer. Il existe quatre opérateurs binaires et un opérateur unaire.

Tableau 2-1. Opérateurs arithmétiques

	OPERATION	OPERATEUR
Binaire	Addition Soustraction Multiplication Division	+ - * /
Unaire	Multiplication par -1	-

2.11.4.1 Règles

Les expressions arithmétiques sont traitées selon les règles de priorité des opérateurs, qui déterminent dans quel ordre sont exécutées les différentes opérations. L'ordre de priorité est le suivant :

1. Multiplication par -1
2. Multiplication et division
3. Addition et soustraction

Les opérations de même priorité sont exécutées consécutivement de la gauche vers la droite. L'ambiguïté n'est donc pas possible. Par exemple, l'expression A / B * C est évaluée comme si elle était écrite (A / B) * C et l'expression A / B / C comme si elle était écrite (A / B) / C.

2.11.4.2 Parenthèses

Il est cependant possible de modifier l'ordre normal d'exécution en utilisant les parenthèses, par exemple : A / (B * C).

2.11.4.3 Combinaison de symboles

Les différentes façons de combiner des symboles sont résumées dans le tableau ci-après, OUI indiquant que la combinaison est autorisée et NON qu'elle ne l'est pas.

Tableau 2-2. Combinaison des symboles dans les expressions arithmétiques

Premier élément	Deuxième élément				
	Opérande	+, -, *, /	- unaire	(	)
opérande	NON	OUI	NON	NON	OUI
+, -, *, /	OUI	NON	OUI	OUI	NON
- unaire	OUI	NON	OUI	OUI	NON
(	OUI	NON	OUI	OUI	NON
)	NON	OUI	OUI	NON	OUI

Une expression arithmétique ne doit jamais commencer ou se terminer par un opérateur sauf s'il s'agit de l'opérateur unaire de multiplication par -1. Le nombre de parenthèses gauches doit être égal au nombre de parenthèses droites.

Les opérations arithmétiques sont traitées de deux façons selon qu'elles ne contiennent que des nombres entiers (variables binaires ou constantes entières) ou non. La différence se situe au niveau de la division : lorsqu'elles ne contiennent que des entiers, le résultat de la division est arrondi au nombre entier immédiatement inférieur (valeur absolue) ; lorsqu'elles contiennent des nombres fractionnaires, les chiffres après la virgule sont conservés.

L'utilisateur doit être prudent lorsqu'il emploie des valeurs décimales élevées, surtout dans des opérations complexes. En effet, la longueur des zones décimales étant limitée à 31 chiffres, il y a risque de débordement et donc de troncation du résultat.

Les expressions arithmétiques peuvent donner lieu aux erreurs suivantes : donnée numérique incorrecte, débordement et division par zéro qui sont respectivement signalées dans la variable système @STATUS par les codes ILLDATA, DATAOV et DIVZR (se reporter au chapitre 3).

2.11.5 Expressions conditionnelles

Les **expressions conditionnelles** permettent de déterminer si une condition, portant généralement sur la nature d'une zone et/ou sur la valeur d'une variable, est réalisée. Elles sont utilisées après le mot-clé WHERE dans les instructions READ et RETRIEVE ou directement dans l'instruction IF.

Il existe deux types d'expressions conditionnelles : les expressions simples et les expressions composées.

2.11.5.1 Expressions conditionnelles simples

```
 -------------------------------------
|   {test de comparaison normal    }   |
|   {test de présence/absence      }   |
|   {tests de comparaison spéciaux }   |
|   {test BETWEEN                  }   |
|   {test AMONG                    }   |
 -------------------------------------
```

Ces différents types de tests sont décrits ci-après.

TEST DE COMPARAISON NORMAL

```
|-------------------------------------------------------|
| <expression>  <opérateur-relationnel>  <expression>   |
|-------------------------------------------------------|
```

Ce test provoque la comparaison de deux opérandes, chacun d'eux étant une expression. Il existe donc deux types de comparaison : comparaison d'opérandes numériques et comparaison d'opérandes non numériques.

Comparaison d'opérandes numériques :

Ce sont les valeurs algébriques des opérandes qui sont comparées. La longueur des opérandes (nombre de chiffres) n'est pas significative.

Comparaison d'opérandes non numériques :

Les caractères des deux opérandes sont comparés un à un de la gauche vers la droite. Les deux opérandes sont considérés comme égaux si tous les caractères dont ils se composent sont égaux un à un. Dans le cas contraire, les deux premiers caractères non égaux détectés sont comparés à l'ordre de classement et l'opérande contenant le caractère le plus élevé dans cet ordre est considéré comme supérieur à l'autre.

Si les deux opérandes sont de longueur inégale, la comparaison est effectuée comme si l'opérande le plus court était complété à droite par des espaces de façon à avoir la même longueur que l'autre.

Les tests et les opérateurs de comparaison sont présentés dans le tableau ci-dessous.

Tableau 2-3. Tests de comparaison et opérateurs correspondants

TEST	OPERATEUR
Egal à	= ou EQ ou EQUAL
Différent de	^ = ou NE ou ^EQUAL ou NOT EQUAL
Supérieur à	> ou GT
Inférieur à	< ou LT
Supérieur ou égal à	> = ou GE
Inférieur ou égal à	< = ou LE

EXEMPLE :

```
(A + B) / C = 4
```

❑

Ici, les termes de la comparaison sont une expression arithmétique et une constante numérique.

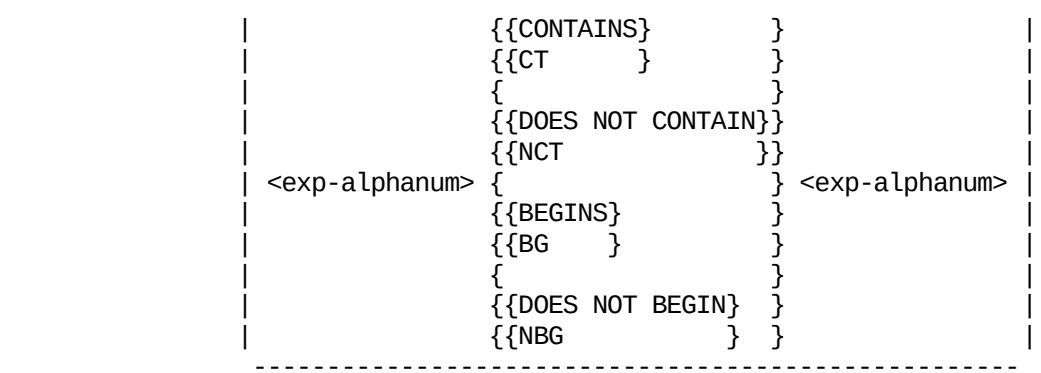
TEST DE PRESENCE/ABSENCE

```
 ---------------------------------------
|                          { {PR     } }  |
|                          { {PRESENT} }  |
|  <expression-alphanum>   {              |
|                          { {AB     } }  |
|                          { {ABSENT } }  |
 ---------------------------------------
```

Ce test détermine si la valeur d'une expression est définie ou non. Il ne s'applique qu'aux expressions alphanumériques, les expressions numériques étant toujours définies.

Une expression alphanumérique est présente si elle contient au moins un caractère différent d'espace. Par exemple, si la zone NOM contient la chaîne de caractères "DUPONT", la condition NOM PRESENT est vérifiée.

TESTS DE COMPARAISON SPECIAUX



Les **tests de comparaison spéciaux** comparent le contenu de deux expressions qui doivent obligatoirement être alphanumériques.

Tableau 2-4. Type de recherche dans les tests de comparaison spéciaux

Mode	Mot-clé relationnelle
Chaîne totale Début chaîne	CONTAINS ou DOES NOT CONTAIN BEGINS ou DOES NOT BEGIN

Test CONTAINS/DOES NOT CONTAIN

Ce test provoque la recherche d'une chaîne de caractères dans une expression. Il ne s'applique qu'à des opérandes alphanumériques. Par exemple, la condition :

```
REGLES CONTAINS "GENERAL"
```

est vérifiée si la zone REGLES contient "LE GRADE DE GENERAL EST SUPERIEUR A CELUI DE CAPITAINE" ou "IL NE FAUT PAS GENERALISER" ou encore "GENERALEMENT". Les espaces à droite dans la deuxième expression sont ignorés sauf si celle-ci est un littéral (ce qui est le cas dans l'exemple).

Test BEGINS/DOES NOT BEGIN

Ce test compare le contenu de deux expressions qui doivent obligatoirement être alphanumériques. La comparaison est effectuée sur le nombre de caractères de la deuxième expression en partant de la gauche. Les espaces à droite dans la deuxième expression sont ignorés sauf si celle-ci est un littéral.

EXEMPLE :

```
   VILLE BEGINS "PARIS"
❑
```

La condition est vérifiée pour une zone de cinq caractères contenant PARIS ou pour une zone de 13 caractères contenant PARIS, FRANCE.

TEST BETWEEN/NOT BETWEEN

```
 ------------------------------------------------------
|         { {BT     }     }                            |
|         { {BETWEEN}     }                            |
| <exp-1> {               } [(]<exp-2> [AND] <exp-3>[)] |
|         { {NBT        } }                            |
|         { {NOT BETWEEN} }                            |
 ------------------------------------------------------
```

La condition est vérifiée si la première expression est comprise (BETWEEN) ou non (NOT BETWEEN) entre les bornes spécifiées par les deuxième et troisième expressions. Elle est également vérifiée si expression-1 est égale à expression-2 ou expression-3.

La valeur de la deuxième expression peut être supérieure ou inférieure à celle de la troisième.

Les trois expressions doivent être du même type, c'est-à-dire toutes alphanumériques ou toutes numériques.

TEST AMONG/NOT AMONG

```
 ------------------------------------------------------
|         {{ AG        } }                             |
|         {{ AMONG     } }                             |
| <exp-1> {{           } } [(]<exp-2> [[,] <exp-3>...[)] |
|         {{ NAG       } }                             |
|         {{ NOT AMONG } }                             |
 ------------------------------------------------------
```

La condition est vérifiée si la première expression est égale à l'une des expressions de la liste (AMONG) ou si elle n'est égale à aucune d'entre elles (NOT AMONG).

Les expressions doivent être toutes du même type, c'est-à-dire toutes alphanumériques ou toutes numériques.

2.11.5.2 Expressions conditionnelles composées

Des expressions conditionnelles simples peuvent être reliées, selon des règles spécifiques, par des opérateurs logiques pour constituer des **expressions conditionnelles composées**.

```
 ------------------------------------------------------------------
|                              {AND}                               |
| [NOT] <condition-simple-1> [{   } [NOT] <condition-simple-2> ] ... |
|                              {OR }                               |
 ------------------------------------------------------------------
```

L'opérateur logique NOT permet d'inverser une condition simple ou composée.

Les conditions simples utilisées pour former une condition composée peuvent appartenir à n'importe lequel des types de condition décrits dans le paragraphe précédent (comparaison simple, présence/absence, comparaison spéciale).
Les parenthèses peuvent être utilisées pour modifier l'ordre implicite de traitement des expressions.

EXEMPLES DE CONDITIONS COMPOSÉES :

```
CL-CODE CT "L" OR CL-NOM EQ "LEBLOND"
PR-QTE > 6 AND PR-QTE < 15
AGE > 25 OR MARIE PRESENT
A < B OR C > D + E
COM-CODE BG 12 AND COM-DATE BETWEEN 900531 AND 910630
A=B AND C=D OR E=F AND G ABSENT OR H < I * J
PR-CODE > 1050 AND (B < 100 OR B > 800)
```

❑

A noter qu'une **condition composée** est toujours une suite de conditions simples reliées par les opérateurs logiques AND et OR (appelés dans ce cas **conjonctions logiques**). Le résultat de la combinaison des conditions simples A et B au moyen des opérateurs AND et OR est indiqué dans le tableau suivant.

Tableau 2-5. Relation entre conditions simples et conditions composées

Conditions simples		Conditions composées	
A	B	A AND B	A OR B
VRAI FAUX VRAI FAUX	VRAI VRAI FAUX FAUX	VRAI FAUX FAUX FAUX	VRAI VRAI VRAI FAUX

Par exemple, si A est vrai et que B est faux, l'expression A AND B est fausse alors que l'expression A OR B est vraie.

Les règles permettant de déterminer si une condition composée est vérifiée ou non sont les suivantes :

1. Si AND est la seule conjonction logique utilisée, l'expression est vraie si et seulement si chacune des conditions simples est vérifiée.
2. Si OR est la seule conjonction logique utilisée, l'expression est vraie si au moins l'une des conditions simples est vérifiée.
3. Si les deux conjonctions logiques AND et OR sont utilisées dans la même expression, deux cas sont à envisager :
   - lorsque des parenthèses sont utilisées, elles doivent l'être de la même façon qu'en algèbre, c'est-à-dire que toute parenthèse ouverte doit être fermée. Dans ce cas, c'est l'expression entre parenthèses qui est traitée en premier. A l'intérieur de parenthèses imbriquées, c'est l'expression la plus interne qui est traitée la première.
   - Lorsque les parenthèses ne sont pas utilisées, les expressions sont traitées dans l'ordre implicite, c'est-à-dire que les conditions AND sont traitées d'abord, de gauche à droite, puis les conditions OR, également de gauche à droite.

Les résultats obtenus avec l'opérateur NOT peuvent être lus dans le tableau 2-5 en remplaçant A par NOT-A et/ou B par NOT-B dans les en-têtes de colonnes.
Par exemple, si NOT est appliqué à la seule condition A, l'en-tête des colonnes doit être remplacé par :

```
|       |   |             |             |
| NOT-A | B | NOT-A AND B | NOT-A OR B |
|       |   |             |             |
```

Les valeurs indiquées dans le tableau restent justes.

En outre, des expressions conditionnelles composées ne contenant pas NOT, peuvent être traitées à partir de conditions simples contenant NOT. Par exemple, les expressions A AND B ou A OR B peuvent être traitées à partir des conditions simples NOT-A et/ou NOT-B. Les différentes combinaisons possibles sont illustrées au tableau 2-6.

Tableau 2-6. Relation entre conditions simples avec NOT et conditions composées sans NOT

Conditions simples		Conditions composées	
NOT-A	**B**	**A AND B**	**A OR B**
VRAI	VRAI	FAUX	VRAI
FAUX	VRAI	VRAI	VRAI
VRAI	FAUX	FAUX	FAUX
FAUX	FAUX	FAUX	VRAI

A	**NOT-B**	**A AND B**	**A OR B**
VRAI	VRAI	FAUX	VRAI
FAUX	VRAI	FAUX	FAUX
VRAI	FAUX	VRAI	VRAI
FAUX	FAUX	FAUX	VRAI

NOT-A	**NOT-B**	**A AND B**	**A OR B**
VRAI	VRAI	FAUX	FAUX
FAUX	VRAI	FAUX	VRAI
VRAI	FAUX	FAUX	VRAI
FAUX	FAUX	VRAI	VRAI

Exemple 1 :

Pour tester la condition composée :

```
C1 AND (C2 OR (C3 OR C4))
```

utiliser la première partie de la règle 3 (cas des parenthèses) et procéder comme suit :

remplacer `C3 OR C4` par `C5`, ce qui donne `C1 AND (C2 OR C5)`,

remplacer `C2 OR C5 par C6`, ce qui donne `C1 AND C6`.

Consulter le tableau 2-6 à chaque phase de l'opération.

Exemple 2 :

Pour tester la condition composée :

```
C1 OR C2 AND C3
```

utiliser la deuxième partie de la règle 3 et placer les parenthèses implicites, ce qui donne :

```
C1 OR (C2 AND C3)
```

puis procéder comme dans l'exemple 1.

Exemple 3 :

Pour tester la condition composée :

```
C1 AND C2 OR C3 AND C4
```

placer les parenthèses implicites, ce qui donne :

```
(C1 AND C2) OR (C3 AND C4)
```

puis procéder comme dans l'exemple 1.

Exemple 4 :

Pour tester la condition composée :

```
C1 AND C2 AND C3 OR C4 OR C5 AND C6 AND C7 OR C8
```

placer les parenthèses implicites, ce qui donne :

```
((C1 AND C2) AND C3) OR C4 OR ((C5 AND C6) AND C7) OR C8
```

puis procéder comme dans l'exemple 1.

Combinaison de symboles

Les différentes possibilités de combinaison des symboles dans les expressions conditionnelles sont indiquées dans le tableau ci-dessous, O signifiant que la combinaison est autorisée et N qu'elle ne l'est pas.

Tableau 2-7. Combinaison de symboles dans les expressions conditionnelles

Premier symbole	Deuxième symbole					
	Condition	OR	AND	(	)	NOT
Condition	N	O	O	N	O	N
OR	O	N	N	O	N	O
AND	O	N	N	O	N	O
(	O	N	N	O	N	O
)	N	O	O	N	O	N
NOT	O	N	N	O	N	N

2.12 Conversions

Le système fournit des possibilités de conversion automatique des valeurs dans les expressions arithmétiques, dans les affectations et dans les comparaisons.

2.12.1 Conversion de valeurs dans les expressions arithmétiques

Les termes des expressions arithmétiques sont convertis de façon à avoir tous le même type. Le type choisi pour la conversion est celui situé au plus haut niveau dans la hiérarchie suivante :

1. Décimal éclaté signé
2. Décimal condensé signé
3. Décimal éclaté non signé
4. Décimal condensé non signé
5. Binaire (15 ou 31)

La conversion des constantes numériques obéit à la règle ci-dessus sachant qu'une constante comportant une marque décimale est toujours convertie en décimal et qu'une constante entière peut être convertie en binaire ou en décimal selon sa longueur.

Le cadrage se fait sur la marque décimale ou, s'il n'y en a pas, sur le chiffre le plus à droite. Si nécessaire, le nombre est complété à gauche et/ou à droite par des zéros. Lorsque le résultat de la conversion est un nombre décimal de plus de 31 chiffres, il y a débordement.

Les conversions d'expressions arithmétiques peuvent donner lieu aux erreurs suivantes : donnée numérique incorrecte et débordement, qui sont respectivement signalées dans la variable système @STATUS par les codes ILLDATA et DATAOV (voir chapitre 3).

2.12.2 Conversion de valeurs dans les affectations

La valeur pouvant être affectée à une zone peut être soit la valeur d'une autre zone, soit la valeur d'une expression. Les types de donnée pouvant être utilisés sont les suivants :

- Alphanumérique
- Décimal éclaté signé
- Décimal éclaté non signé
- Décimal condensé signé
- Décimal condensé non signé
- Binaire

Une valeur alphanumérique ne peut être convertie en numérique que si elle correspond à une valeur numérique éditée ; sinon, le système émet un message d'avertissement.
Le format d'une valeur numérique éditée est le suivant :

```
 ------------------------------------------------------------
|           [   {+}                                        ] |
| [espaces] [ [ { } ] [espaces] chiffres [.chiffres] [espaces] ] |
|           [   {-}                                        ] |
 ------------------------------------------------------------
```

Lorsque la partie droite d'une affectation est une expression arithmétique, celle-ci est convertie en donnée numérique dont le type est celui du terme de l'expression situé au plus haut niveau de la hiérarchie ou celui de la partie gauche de l'affectation s'il est plus élevé (voir "Conversion des expressions arithmétiques").

La conversion s'effectue de droite à gauche avec alignement sur la marque décimale ou, s'il n'y en a pas, sur le chiffre le plus à droite. La valeur convertie est placée dans la zone réceptrice selon le modèle d'édition et le cadrage implicites ou explicites.

La troncation et le remplissage s'effectuent comme suit :

- si la partie gauche de l'affectation est de type binaire, le signe est placé dans la position la plus à gauche ; si un chiffre significatif (c'est-à-dire situé à gauche de la marque décimale) est tronqué, le système émet un message et la variable @STATUS prend la valeur DATAOV (débordement).
- si la partie gauche est de type décimal, la troncation peut également se produire à droite ; lorsque la partie entière est tronquée, le système le signale par un message et @STATUS prend la valeur DATAOV.
- si la partie gauche est de type alphanumérique, la troncation et le remplissage par des espaces se font à droite ; si une partie significative de la valeur est tronquée, le système émet un message d'avertissement.

Une zone de type décimal ne doit pas excéder 31 chiffres. Lorsqu'une valeur dont la longueur est supérieure à 31 chiffres est affectée à une zone de ce type, il y a donc débordement.

Les conversions de valeurs d'affectation peuvent donner lieu aux erreurs suivantes : donnée incorrecte et débordement, qui sont respectivement signalées dans @STATUS par les codes ILLDATA et DATAOV (voir plus haut dans ce chapitre).

2.12.3 Conversion de valeurs dans les comparaisons

Les deux termes d'une comparaison sont des expressions. Les types de donnée pouvant être utilisés sont les mêmes que dans les affectations.

Lorsque l'un des termes d'une comparaison est une expression arithmétique, celle-ci est convertie en donnée numérique dont le type est celui du terme de l'expression situé au plus haut niveau de la hiérarchie.

La comparaison entre alphanumérique et numérique n'est pas possible.

Lorsque les deux termes de la comparaison sont de type alphanumérique, ils ne sont pas convertis.

Lorsque les deux termes sont de type numérique, ils sont tous deux convertis en données du même type (cf. "Conversion des expressions arithmétiques") puis comparés. Lorsque le résultat d'une conversion en décimal est un nombre de plus de 31 chiffres, il y a débordement.

Lorsqu'une erreur est détectée dans une comparaison (ILLDATA ou DATAOV dans @STATUS), c'est l'ensemble des conditions dont elle se compose qui est considéré comme faux et non la seule condition comportant l'erreur.

2.13 Cadrage

2.13.1 Cadrage et édition

Lorsque la clause PICTURE est utilisée, la valeur est éditée selon le modèle spécifié avant d'être cadrée dans la zone réceptrice (selon le cadrage implicite ou selon le cadrage spécifié dans la clause JUSTIFIED).

2.13.2 Cadrage implicite

Lorsqu'une valeur est transférée d'une zone émettrice vers une zone réceptrice, il peut arriver que ces deux zones soient de longueur différente. De plus, la valeur peut comporter des espaces significatifs.

Les fonctions de cadrage, implicite ou explicite (clause JUSTIFIED), permettent de choisir comment doit être placée la valeur dans la zone réceptrice.

Le cadrage implicite se fait toujours à gauche, sauf dans le cas d'une valeur numérique traitée par une instruction PRINT.

REMARQUE :

Les espaces éventuels situés à gauche et à droite de la valeur sont conservés lorsque le cadrage est implicite alors qu'ils sont supprimés lorsque la clause JUSTIFIED est utilisée.

2.13.2.1 Cadrage implicite à gauche

Le premier caractère de la valeur est placé dans la première position de la zone réceptrice.

La troncation, si nécessaire, est effectuée à droite.

2.13.2.2 Cadrage implicite à droite

Le dernier caractère de la valeur est placé dans la dernière position de la zone réceptrice.

La troncation, si nécessaire, est effectuée à gauche.

2.13.3 Clause JUSTIFIED

Fonction :

La clause JUSTIFIED (abréviation JUST) permet d'éviter le cadrage implicite lors du transfert d'une valeur dans une zone alphanumérique.

Format :

```
 -------------------------
|               {LEFT  }  |
|  JUST[IFIED]  {CENTER}  |
|               {RIGHT }  |
 -------------------------
```

Description :

Cette clause s'utilise dans différentes instructions pour indiquer comment doit être placée une valeur dans une zone réceptrice.

La zone réceptrice doit être alphanumérique.

Lorsque cette clause n'est pas spécifiée, le cadrage est implicite (voir paragraphe précédent).

REMARQUE :

Contrairement à ce qui passe dans le cadrage implicite, les espaces à gauche et à droite de la valeur sont ignorés.

Paramètres :

LEFT	Le caractère autre qu'espace le plus à gauche de la valeur est placé dans la position la plus à gauche de la zone réceptrice. La troncation, si nécessaire, se fait à droite.
CENTER	Les espaces à gauche et à droite de la valeur sont ignorés. Le reste de la valeur est placé dans la zone réceptrice de façon à ce que le nombre d'espaces à droite soit égal ou supérieur de 1 au nombre d'espaces à gauche. La troncation, si nécessaire, se fait des deux côtés, le nombre de caractères tronqués à droite étant égal ou supérieur de 1 au nombre de caractères tronqués à gauche.
RIGHT	Le caractère autre qu'espace le plus à droite de la valeur est placé dans la position la plus à droite de la zone réceptrice. La troncation, si nécessaire, se fait à gauche.

2.14 Modèle d'édition

2.14.1 Clause PICTURE

Fonction :

La clause PICTURE (abréviation PIC) s'utilise dans les instructions LET, MODIFY et PRINT pour indiquer comment doit être éditée une valeur avant affectation ou impression.

Format :

```
 ------------------------------------------------------
|            { " <chaîne-de-caractères-protégée> "  }  |
|  PIC[TURE] {                                      }  |
|            { <chaîne-de-caractères-non-protégée>  }  |
 ------------------------------------------------------
```

une chaîne protégée pouvant contenir :

```
 ----------------------------------------------
|  { tout-caractère-sauf-guillemet  }          |
|  {                                } ...      |
|  { " "                            }          |
 ----------------------------------------------
```

et une chaîne non protégée :

```
 ----------------------------------------------
|   tout-caractère-sauf-espace-et-fin-de-ligne  |
 ----------------------------------------------
```

Description :

La chaîne de caractères qui définit le modèle d'édition est une combinaison de symboles d'édition et de caractères normaux. Elle ne doit pas excéder 30 caractères. Les symboles d'édition indiquent comment doit être éditée la valeur avant d'être transférée dans une zone (LET ou MODIFY) ou imprimée (PRINT). La liste de ces symboles est fournie dans le paragraphe suivant.

Les symboles d'édition "V", ".", "CR" et "DB" ne peuvent apparaître qu'une seule fois dans un même modèle. Les autres peuvent être suivis d'un entier non signé entre parenthèses qui indique combien de fois ils se répètent, cet entier devant être inférieur ou égal à 255 et différent de zéro. Par exemple, X(9) équivaut à XXXXXXXXX, la première notation ayant l'avantage de ne comporter que 4 caractères (sur les 30 utilisables) au lieu de 9.

Le modèle d'édition décrit le type de la zone réceptrice : alphanumérique ou numérique. Une valeur alphanumérique ne peut être transférée dans une zone numérique que si elle correspond à une valeur numérique éditée alors qu'une zone numérique peut être transférée dans une zone numérique ou alphanumérique.

L'interprétation de la combinaison de symboles d'édition et de caractères détermine la longueur de la zone réceptrice. Si celle-ci est insuffisante, la valeur est tronquée comme suit :

- à droite si la clause PICTURE spécifie une zone alphanumérique,
- à droite et à gauche après alignement sur la marque décimale (implicite ou explicite) si la clause PICTURE spécifie une zone numérique ; par exemple, la valeur 123.456 après transfert dans une zone dont le modèle d'édition est 9.99 devient 3.45.

Un modèle d'édition doit comporter au moins l'un des symboles "X", "Z", "9" ou "*" ou au moins deux des symboles "+", "-" ou "$".

Du point de vue syntaxique, une clause PICTURE est un mot IQS.

2.14.2 Description des symboles d'édition

Tableau 2-8. Symboles d'édition

SYMBOLE	DESCRIPTION
B	Position de la zone à visualiser ou imprimer devant contenir un espace.
9	Zone numérique : position alphanumérique devant contenir l'un des chiffres de la valeur (y compris zéro).
.	Zone numérique : position devant contenir la marque décimale sur laquelle la valeur sera alignée en sortie ; ce symbole ne doit être utilisé qu'une seule fois. L'option DECIMAL POINT IS COMMA est applicable aux "zones imprimées". Voir annexe C.
$	Position devant contenir la valeur présente dans la variable système @CURRENCY-SIGN au moment de l'édition (voir également le paragraphe "Insertion flottante").
V	Zone numérique : position de la marque décimale implicite ; le V est donc inutile s'il est placé dans la dernière position du modèle ; il ne doit être utilisé qu'une seule fois et ne compte pas dans la longueur puisqu'il ne représente pas un caractère en sortie.
X	Position alphanumérique devant contenir l'un des caractères de la valeur.
Z	Zone numérique : position située au début de la zone à visualiser ou imprimer et devant contenir un chiffre ou, si ce chiffre est égal à zéro, un espace. Ce symbole permet de supprimer les zéros non significatifs à gauche.
*	Zone numérique : position située au début de la zone à visualiser ou imprimer et devant contenir un chiffre ou, si ce chiffre est égal à zéro, un astérisque (*). Ce symbole est souvent utilisé pour les chèques pour des raisons de sécurité.
+	Zone numérique : position devant contenir un signe moins si la valeur est négative, un signe plus si elle est positive ou égale à zéro.
-	Zone numérique : position devant contenir un signe moins si la valeur est négative, un espace si elle est positive ou égale à zéro.
CR DB	Zone numérique : 2 positions devant contenir les caractères CR (crédit) ou DB (débit) si la valeur est négative ou deux espaces si elle est positive. Ces deux symboles ne peuvent être utilisés qu'une seule fois dans un modèle et doivent être placés dans les deux dernières positions de celui-ci (voir également le paragraphe "Insertion fixe").

2.14.3 Modèle d'édition alphanumérique

Un modèle d'édition décrit une zone de type alphanumérique s'il comporte au moins un X. Le modèle peut être utilisé pour modifier la longueur de la valeur ou pour insérer des caractères dans la zone réceptrice (ou les deux).

2.14.3.1 Modification de la longueur

Le modèle d'édition ne comporte que des X dont le nombre détermine la longueur de la zone réceptrice. Chaque X correspond à une position devant être occupée par un caractère de la valeur.

2.14.3.2 Insertion de caractères

Tout caractère autre que X ou B est inséré dans la zone réceptrice à la position qu'il occupe dans le modèle. Le caractère B provoque l'insertion d'un espace. Le caractère X est remplacé par un caractère de la valeur.

2.14.3.3 Transfert dans la zone réceptrice

La valeur est transférée dans la zone réceptrice selon le cadrage implicite (cf."Cadrage implicite") ou explicite (clause JUSTIFIED).

2.14.4 Modèle d'édition numérique

Un modèle d'édition décrit une zone de type numérique s'il ne comporte aucun caractère X.

Les différents types d'édition d'une valeur numérique sont indiqués ci-dessous et décrits plus en détail par la suite.

insertion du signe	insertion du signe (+, -, CR ou DB) dans la zone réceptrice.
insertion simple	insertion, dans la zone réceptrice, des caractères autres que des symboles d'édition.
insertion fixe	insertion du symbole monétaire et/ou des caractères +, -, CR ou DB.
insertion spéciale	insertion de la marque décimale.
suppression des zéros	suppression des zéros non significatifs à gauche et remplacement par des espaces ou des astérisques.
insertion flottante	insertion du symbole monétaire, du signe plus ou du signe moins devant le premier chiffre significatif.

2.14.4.1 Insertion du signe

Les symboles d'insertion du signe sont "+", "-", "CR" et "DB". Leur utilisation provoque l'insertion d'un signe dans la zone réceptrice qui est fonction du signe de la valeur comme le montre le tableau ci-après.

Tableau 2-9. Symboles d'insertion du signe

Symbole	Valeur positive ou égale à zéro	Valeur négative
+	+	-
-	espace	-
CR	deux espaces	CR
DB	deux espaces	DB

2.14.4.2 Insertion simple

Les caractères d'insertion simple sont tous les caractères autres que des symboles d'édition. Ils sont insérés dans la zone réceptrice à la position qu'ils occupent dans le modèle d'édition.

Un caractère d'insertion simple est supprimé s'il doit être placé entre des zéros non significatifs à gauche et que la suppression de ceux-ci est demandée.

Un caractère d'insertion simple est recouvert par un caractère d'insertion flottante ou par le symbole monétaire si la longueur de la zone réceptrice est insuffisante.

Le caractère B est également un caractère d'insertion simple. Il provoque l'insertion d'un espace dans la zone réceptrice.

2.14.4.3 Insertion fixe

Les symboles $, +, -, CR et DB sont des symboles d'insertion fixe lorsqu'ils n'apparaissent qu'une seule fois dans le modèle d'édition. Ils fonctionnent comme des caractères d'insertion simple si ce n'est qu'ils ne peuvent être placés dans le modèle d'édition qu'à certaines positions.

Les symboles CR et DB doivent être placés dans les deux positions les plus à droite du modèle ou, si les dernières positions sont occupées par des caractères d'insertion simple, juste avant ceux-ci.

Les symboles + et - doivent être placés dans la position la plus à gauche ou dans la position la plus à droite du modèle ou, si des caractères d'insertion simple occupent les premières ou dernières positions, juste après ou juste avant ceux-ci.

Le symbole monétaire $ doit être placé dans la position la plus à gauche ou dans la position la plus à droite du modèle d'édition ou, si les premières ou dernières positions sont occupées par des symboles d'insertion du signe ou par des caractères d'insertion simple, juste après ou juste avant ceux-ci.

L'utilisation des symboles +, -, CR et DB en tant que symboles d'insertion fixe provoque le même résultat que leur utilisation en tant que symboles d'insertion du signe (cf. ci-dessus).

Le symbole monétaire provoque l'insertion du caractère présent dans la variable système @CURRENCY-SIGN au moment de l'édition.

2.14.4.4 Insertion spéciale

Le point (.) fonctionne non seulement comme un symbole d'insertion fixe mais aussi comme marque de cadrage, c'est-à-dire qu'il provoque l'insertion de la marque décimale à la position qu'il occupe dans le modèle d'édition et que les nombres sont alignés sur cette marque.

Le symbole "V" qui représente la marque décimale implicite et le symbole "." qui représente la marque décimale réelle ne doivent pas être utilisés dans un même modèle.

2.14.4.5 Suppression des zéros

Les symboles "Z" et "*" indiquent que les zéros non significatifs à gauche doivent être supprimés dans la zone réceptrice et remplacés par des espaces (Z) ou des astérisques (*). Ces deux symboles s'excluent mutuellement.

La suppression est spécifiée par une chaîne d'un ou plusieurs symboles qui représentent les positions dans lesquelles les zéros doivent être supprimés. Les caractères d'insertion simple inclus entre les symboles ou situés immédiatement à leur droite font partie de la chaîne.

La suppression des zéros non significatifs peut être spécifiée de deux façons : 1) en représentant tout ou partie des positions situées à gauche de la marque décimale par des symboles de suppression, 2) en représentant toutes les positions numériques de la zone réceptrice par des symboles de suppression.

Lorsqu'une partie ou la totalité des positions situées à gauche de la marque décimale sont représentées par des symboles de suppression des zéros, ceux-ci sont remplacés par des espaces (Z) ou des astérisques (*) chaque fois qu'ils correspondent à des zéros non significatifs à gauche.

Lorsque toutes les positions numériques de la zone réceptrice sont représentées par des symboles de suppression, trois cas sont à distinguer :

- si la valeur est différente de zéro, seuls les zéros précédant la marque décimale sont supprimés.
- si la valeur est égale à zéro et que le symbole utilisé est Z, toutes les positions à partir du premier symbole Z sont remplacées par des espaces.
- si la valeur est égale à zéro et que le symbole utilisé est *, toutes les positions à partir du premier symbole * sont remplacées par des astérisques à l'exclusion de la position de la marque décimale.

La suppression des zéros et l'insertion flottante s'excluent mutuellement.

2.14.4.6 Insertion flottante

L'insertion flottante consiste à insérer un signe (+ ou -) ou un symbole monétaire juste avant le premier chiffre différent de zéro.

Elle est spécifiée dans le modèle d'édition par une chaîne d'insertion flottante qui comporte au moins deux caractères $, deux signes plus (+) ou deux signes moins (-). Cette chaîne peut également comporter des caractères d'insertion simple après le premier symbole d'insertion flottante. Les trois symboles d'insertion flottante ($, + et -) s'excluent mutuellement.

Les symboles d'insertion flottante et les symboles de suppression des zéros ne doivent pas être utilisés dans un même modèle.

Le caractère le plus à gauche et le caractère le plus à droite de la chaîne d'insertion flottante représentent les positions extrêmes que peut occuper le symbole d'insertion flottante dans la zone réceptrice.

Le second symbole d'insertion flottante en partant de la gauche représente la première position pouvant être occupée par un chiffre dans la zone réceptrice, c'est-à-dire que tous les symboles d'insertion flottante à partir du deuxième peuvent être remplacés dans la zone réceptrice par des chiffres différents de zéro.

L'insertion flottante peut être spécifiée de deux façons : 1) en représentant tout ou partie des positions situées à gauche de la marque décimale par une chaîne d'insertion flottante, 2) en représentant toutes les positions numériques de la zone réceptrice par une chaîne d'insertion flottante.

Lorsqu'une partie ou la totalité des positions situées à gauche de la marque décimale sont représentées par des symboles d'insertion flottante, le premier chiffre différent de zéro, ou la marque décimale si celle-ci n'est précédée que par des zéros, est précédé du symbole monétaire, du signe plus ou du signe moins selon le symbole utilisé. Les positions représentées par des caractères d'insertion flottante et situées à gauche du caractère ainsi inséré sont mises à espaces.

Lorsque toutes les positions numériques de la zone réceptrice sont représentées par des symboles d'insertion flottante, au moins l'un de ceux-ci doit précéder la marque décimale. Si la valeur est différente de zéro, l'édition est effectuée de la même manière que ci-dessus. Si elle est égale à zéro, la zone réceptrice ne contient que des espaces et, éventuellement, des caractères d'insertion simple ou fixe dans les premières positions.

Lorsque le symbole d'insertion flottante utilisé est "+", la valeur est précédée du signe plus si elle est positive ou égale à zéro et du signe moins si elle est négative.

Lorsque le symbole utilisé est "-", la valeur est précédée d'un espace si elle est positive ou égale à zéro et du signe moins si elle est négative.

Lorsque le symbole utilisé est "$", la valeur est précédée du symbole monétaire spécifié dans la variable système @CURRENCY-SIGN.

Pour éviter la troncation, la longueur de la zone réceptrice doit être au moins égale à la longueur de la valeur plus x positions pour les caractères d'insertion autre que flottante apparaissant à l'édition, plus 1 position pour le caractère d'insertion flottante.

2.14.5 Modèle d'édition implicite

Lorsque le modèle d'édition n'est pas spécifié dans les instructions où la clause PICTURE peut être utilisée, il prend une valeur implicite qui est fonction du type de la valeur. Les valeurs implicites sont indiquées dans le tableau ci-dessous.

Tableau 2-10. Modèles d'édition implicite

Type de valeur	Modèle d'édition implicite
CHARACTER (t)	X (t)
UNSIGNED DECIMAL (t) (PACKED ou UNPACKED)	Z (t-1) 9
SIGNED DECIMAL (t) (PACKED ou UNPACKED)	- (t) 9
BINARY 15	- (5) 9
BINARY 31	- (10) 9
UNSIGNED DECIMAL (t, d) (PACKED ou UNPACKED)	Z (e) .9 (d)
SIGNED DECIMAL (t, d) (PACKED ou UNPACKED)	- (e+1) .9 (d)

t = longueur totale de la valeur

d = longueur de la partie décimale de la valeur

e = longueur de la partie entière de la zone

(e est donc égal à t-d)

Tableau 2-11. Exemples de modèles d'édition

PICTURE	12.34	.01	.00	-.01	-12.34
9999.99	0012.34	0000.01	0000.00	0000.01	0012.34
ZZZZ.ZZ	12.34	.01	.00	.01	12.34
ZZZZ.99	12.34	.01	0.00	.01	12.34
ZZZ9.99	12.34	0.01	****.**	0.01	12.34
$$$$.$$	$12.34	$.01	+0000.00	$.01	$12.34
****.**	**12.34	****.01	+	****.01	**12.34
----.--	12.34	.01	+	-.01	-12.34
++++.++	+12.34	+.01	+****.**	-.01	-12.34
+9999.99	+0012.34	+0000.01	0000.00+	-0000.01	-0012.34
+ZZZZ.ZZ	+ 12.34	+ .01	****.***	- .01	- 12.34
+$$$$.$$	+ $12.34	+ $.01	$0000.00	- $.01	- $12.34
+****.**	+**12.34	+****.01	$	-****.01	-**12.34
9999.99+	0012.34+	0000.01+	$****.**	0000.01-	0012.34-
ZZZZ.ZZ+	12.34+	.01+	$	.01-	12.34-
$$$$.$$+	$12.34+	$.01+	$	$.01-	$12.34-
****.**+	**12.34+	****.01+	0000.00	****.01-	**12.34-
$9999.99	$0012.34	$0000.01	****.****	$0000.01	$0012.34
$ZZZZ.ZZ	$ 12.34	$ .01	0,000.00	$ .01	$ 12.34
$****.**	$**12.34	$****.01	*****.**	$****.01	$**12.34
$----.--	$ 12.34	$ .01		$ -.01	$ -12.34.
$++++.++	$ +12.34	$ +.01		$ -.01	$ -12.34.
9999.99CR	0012.34	0000.01		0000.01CR	0012.34CR
ZZZZ.ZZCR	12.34	.01		.01CR	12.34CR
$$$$.$$CR	$12.34	$.01		$.01CR	$12.34CR
****.**CR	**12.34	****.01		****.01CR	**12.34CR
9,999.99	0,012.34	0,000.01		0,000.01	0,012.34
Z,ZZZ.ZZ	12.34	.01		.01	12.34
$,$$$.$$	$12.34	$.01		$.01	$12.34
*,***.**	***12.34	*****.01		*****.01	***12.34
-,---.--	12.34	.01		-.01	-12.34
+,+++.++	+12.34	+.01		-.01	-12.34

2.15 Représentation des données

Ce paragraphe concerne l'interprétation par IQS des zones de données décrites par l'utilisateur et leur représentation en mémoire.

Les zones destinées à recevoir les données (ou valeurs) sont soit alphanumériques (type CHARACTER), soit numériques. Une zone numérique peut être de type binaire (BINARY) ou décimal (DECIMAL), et dans ce dernier cas plusieurs formats sont possibles.

2.15.1 Types de données

Le tableau 2-12 ci-dessous indique les types de données autorisés sous IQS ainsi que les correspondances en COBOL.

Tableau 2-12. Types de données IQS, DDL et COBOL autorisés par IQS

IQS		DDL	COBOL	
			PICTURE	USAGE
[UNPACKED] [SIGNED] DECIMAL	m m,p (m>p) m,p (m=p)	SIGNED UNPACKED m[,p] DECIMAL	S9(m) S9(m-p)V9(p) SV9(m)	
[UNPACKED] UNSIGNED [DECIMAL]	m m,p (m>p) m, p (m=p)	UNSIGNED UNPACKED m[,p] DECIMAL	9(m) 9(m-p)V9(p) V9(m)	
PACKED [SIGNED] [DECIMAL]	m m,p (m>p) m,p (m=p)	SIGNED PACKED m[,p] DECIMAL	S9(m) S9(m-p)V9(p) SV9(m)	COMP
PACKED UNSIGNED [DECIMAL]	m m,p (m>p) m,p (m=p)	UNSIGNED PACKED-2 m[,p] DECIMAL	9(m) 9(m-p)V9(p) V9(m)	COMP-8
		UNSIGNED PACKED m[,p] DECIMAL	9(m) 9(m-p)V9(p) V9(m)	COMP
SIGNED BINARY 15		SIGNED BINARY 15		COMP-1
SIGNED BINARY 31		SIGNED BINARY 31		COMP-2
CHARACTER n		CHARACTER n	X(n)	

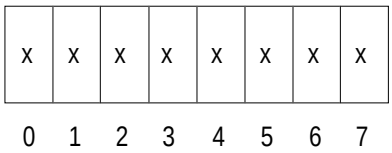
La première colonne (IQS) indique les types de données utilisés dans les instructions DEFINE et PARAMETER du langage de requêtes, pour la définition de zones additionnelles dans le processeur de vues et pour celle des zones d'une structure dans le cas du processeur de structures.

Le format en mémoire des différents types de données est fourni dans les paragraphes suivants.

2.15.2 Format des données en mémoire

Les données sont enregistrées en mémoire sous forme d'octets.

Chaque octet comprend huit positions binaires ou bits qui sont numérotés de 0 à 7 en partant de la gauche. Le format d'un octet est donc le suivant :



x = 0 ou 1

La valeur rangée dans un octet est représentée dans les paragraphes suivants sous forme de deux chiffres hexadécimaux (0 à 9 et A à F).

2.15.3 Chaîne de caractères

Une chaîne de caractères est rangée dans une zone de type alphanumérique définie par CHARACTER n (n représentant la longueur de la chaîne). Cette zone mémoire est formée d'octets contigus contenant chacun un caractère de la chaîne en code EBCDIC. La chaîne peut contenir n'importe quel caractère du jeu EBCDIC (00 à FF).

2.15.4 Valeur décimale éclatée

Une **valeur décimale éclatée** (UNPACKED), qu'elle soit signée (SIGNED) ou non (UNSIGNED), est rangée dans une zone numérique ayant le format suivant :

octet		octet		octet		octet	
quartet gauche	quartet droit	quartet gauche	quartet droit	quartet gauche	quartet droit	quartet gauche	quartet droit
-	chiffre	-	chiffre	-	chiffre	signe	chiffre

Chaque chiffre occupe donc les quatre bits de droite dans chaque octet :

- les valeurs de 0 (0000) à 9 (1001) sont autorisées ;
- les valeurs de A (1010) à F (1111) sont interdites et entrainent une condition d'exception du type "*illegal numeric data*" (donnée numérique incorrecte).

Le contenu des quatre bits de gauche n'est pas contrôlé.

Le signe éventuel est placé dans les quatre bits de gauche du dernier octet :

- les valeurs de A (1010) à F (1111) sont autorisées ;
- les valeurs de 0 (0000) à 9 (1001) sont interdites et entrainent une condition d'exception du type "illegal numeric data" (donnée numérique incorrecte).

La correspondance entre le codage du signe et sa signification est la suivante :

Codage du signe	Signe correspondant
1010	+
1011	-
1100	+
1101	-
1110	+
1111	+

En ce qui concerne les instructions utilisant des zones numériques de type éclaté signé (UNPACKED SIGNED), le signe est représenté comme suit :

Codage du signe	Signe correspondant
1100	+
1101	-

Le contenu des quatre bits de gauche des opérandes origine n'est pas contrôlé et est remplacé par le code F (1111) dans la zone réceptrice. Il en est de même pour la zone réservée au signe dans le cas d'une valeur non signée (UNSIGNED). Dans les deux cas, la valeur numérique à ranger dans la zone ne doit pas comprendre plus de 31 chiffres.

La clause "SIGN IS TRAILING SEPARATE" utilisée en COBOL n'est pas autorisée sous DDL ou IQS bien que le format correspondant soit reconnu par le sous-programme de conversion IQS. Quand cette clause est spécifiée, le signe codé en EBCDIC (4E = +, 60 = -) est placé dans le dernier octet de la zone réceptrice.

En langage DDL et IQS, ce type de donnée doit être déclaré en tant que valeur décimale de format SIGNED UNPACKED m+1,p+1 (m étant le nombre de chiffres précédant le signe et p le nombre de chiffres après la virgule). Le dernier chiffre rangé dans la zone réceptrice sera toujours zéro.

Ce format doit être utilisé avec précaution car il exige une conversion préalable. Cette conversion est effectuée soit lors de l'impression (PRINT) ou de l'affichage (DISPLAY) de la zone, soit lors de son affectation à une zone de type différent. Elle ne peut pas avoir lieu dans une expression arithmétique car le type SIGNED UNPACKED DECIMAL a la priorité la plus élevée parmi tous les types de données.

2.15.5 Valeur décimale condensée

Une **valeur décimale condensée** (PACKED), qu'elle soit signée (SIGNED) ou non (UNSIGNED), occupe une zone mémoire formée d'octets contigus, chaque octet (à l'exception du dernier) permettant d'enregistrer deux chiffres occupant 4 bits chacun. Les quatre bits de droite du dernier octet de la zone réceptrice sont réservés au signe, sauf dans le cas d'une valeur condensée non signée (UNSIGNED PACKED) en DDL où cette zone est occupée par le dernier chiffre. Il est conseillé d'éviter, si possible, de définir des zones DDL de ce type, étant donné que le système doit faire appel à une routine de conversion externe pour traiter ce format.

Une valeur décimale condensée, qu'elle soit signée (SIGNED) ou non (UNSIGNED), est rangée dans une zone numérique ayant le format suivant :

octet		**octet**		**octet**		**octet**	
quartet gauche	quartet droit	quartet gauche	quartet droit	quartet gauche	quartet droit	quartet gauche	quartet droit
chiffre	chiffre	chiffre	chiffre	chiffre	chiffre	chiffre	signe

Chaque chiffre occupe quatre positions binaires ou bits :

- les valeurs de 0 (0000) à 9 (1001) sont autorisées ;
- les valeurs de A (1010) à F (1111) sont interdites et entrainent une condition d'exception du type "*illegal numeric data*" (donnée numérique incorrecte).

Le signe éventuel est placé dans les quatre bits de droite du dernier octet de la zone :

- les valeurs de A (1010) à F (1111) sont autorisées ;
- les valeurs de 0 (0000) à 9 (1001) sont interdites et entrainent une condition d'exception du type "illegal numeric data" (donnée numérique incorrecte).

La correspondance entre le codage du signe et sa signification est la suivante :

Codage du signe	Signe correspondant
1010	+
1011	-
1100	+
1101	-
1110	+
1111	+

Le codage du signe par les instructions est le suivant :

Codage du signe	Signe correspondant
1100	+
1101	-

Une valeur décimale condensée ne doit pas occuper plus de seize octets. La longueur de la zone réceptrice est fonction du nombre de chiffres (m) que comprend la valeur, selon la règle ci-dessous :

```
nombre d'octets = (m + 1) / 2
```

Quand m est un nombre pair, les quatre premiers bits de la zone doivent être à zéro.

2.15.6 Valeur binaire

Une valeur binaire à virgule fixe est rangée dans une zone numérique binaire ayant l'un des deux formats suivants :

- BINARY 15 (16 bits)
- BINARY 31 (32 bits)

Une zone de type BINARY 15 comprend deux octets contigus et une de type BINARY 31 en comprend quatre.

Dans les deux cas, la marque décimale est placée à droite du bit le moins significatif. Les valeurs négatives sont enregistrées sous forme de complément à 2. La fourchette de valeurs autorisées est de -32768 à 32767 pour BINARY 15 et -2147483648 à 2147483647 pour BINARY 31.

3. Variables système IQS

Les variables système IQS sont utilisables aussi bien dans les commandes IQS que dans le langage de requêtes. Elles permettent entre autres :

- de transmettre des valeurs,
- de déterminer le déroulement du traitement,
- de définir la présentation des états,
- de visualiser, vérifier et/ou modifier des valeurs.

La plupart des variables système utilisables dans les conditions et les définitions de zones additionnelles peuvent être lues par les commandes IQS.

Les variables système sont soit numériques, soit alphanumériques et peuvent être classées par type. Les cinq types de variables sont les suivants :

- permanentes
- générales fixes
- générales modifiables
- fichier fixes
- fichier modifiables

Ces différents types sont décrits dans les paragraphes suivants. La liste complète des variables, avec leur description détaillée, est fournie en fin de chapitre.

3.1 Variables permanentes

Une **variable permanente** est une variable qui existe pendant toute la durée d'une session IQS.

```
@CHAR[1..10]
#DEC[1..10]
$MAX-CPU
$MAX-CPU-PER-COMMAND
$MAX-IO
$MAX-IO-PER-COMMAND
$MAX-LINES
$MAX-LINES-PER-COMMAND
$NUM[1..10]
$PRIORITY
@TRIGGER-CODE-ERROR
@TRIGGER-ERROR
```

Il existe 10 variables permanentes binaires : $NUM1 à $NUM10. Leur définition implicite est BINARY 31 (binaire, longueur 31). Celle-ci peut être remplacée par une définition numérique explicite.

Il existe 10 variables permanentes décimales : #DEC1 à #DEC10. Leur définition implicite est PACKED SIGNED (15,2) (condensé signé, longueur 15,2). Celle-ci peut être remplacée par une définition numérique explicite.

Il existe 10 variables permanentes alphanumériques : @CHAR1 à @CHAR10. Leur définition implicite est CHARACTER 32 (caractère, longueur 32). Celle-ci peut être remplacée par une définition alphanumérique explicite, la longueur devant être inférieure ou égale à 32.

Les variables permanentes peuvent être redéfinies à l'intérieur d'une requête au moyen d'une instruction DEFINE, mais cette définition ne prime sur la définition standard que le temps de la requête. La valeur d'une variable permanente est externe à la requête ; ce type de variable peut donc servir à communiquer des informations entre requêtes exécutées au cours de la même session IQS ou entre requêtes et commandes IQS.

REMARQUE :

Une variable permanente ne doit jamais être redéfinie comme zone de type vecteur. Se reporter à l'instruction DEFINE.

Exemple :

Les variables permanentes peuvent être utilisées dans une commande IQS pour la spécification de conditions et la définition de zones additionnelles. Dans l'exemple ci-dessous, une requête transmet la valeur de la variable permanente @CHAR3 à une commande RETAIN où elle est spécifiée comme condition.

```
V: ^,$L
  10 : LET @CHAR3 = "12890G"
  20 : DISPLAY @CHAR3
V: GO
  CHAR3: 12890G
V: RT * FROM CLIENTS WHERE CL-CODE = @CHAR3
V: RV
R: F
  CLIENTS
       CL-CODE: 12890G
       CL-NOM: LEBLOND
       CL-RUE: 57 RUE DE VERDUN
       CL-VILLE: GRENOBLE
R: N
* il n'y a plus d'articles CLIENTS
R:
```

3.2 Variables générales fixes

Les **variables générales fixes** peuvent être lues au cours d'une requête, mais ne peuvent pas être modifiées par celle-ci. Selon leur nature, leur valeur reste la même durant l'exécution de la requête ou évolue au cours de celle-ci.

Ces variables peuvent être lues par des commandes IQS.

```
@ACTION
@CURRENT-VIEW
$DATE
@DATE
$DAY
@DAY
@EXEC-MODE
$MAX-ELAPSE
@MAX-ELAPSE
$MONTH
@MONTH
@PROJECT
@SLLIB
@STATUS
$TIME
@TIME
@TTYPE
$USED-CPU
@USED-CPU
$USED-ELAPSE
@USED-ELAPSE
$USED-IO
$USED-IO-PER-COMMAND
$USED-LINES
$USED-LINES-PER-COMMAND
@USER
$YEAR
```

3.3 Variables générales modifiables

Les **variables générales modifiables** peuvent être lues ou modifiées au cours de l'exécution d'une requête.

Ces variables peuvent être lues par des commandes IQS.

```
@CURRENCY-SIGN
$FUNCTION-KEY
@MESSAGE
@NEXT-QUERY
@NEXT-SCRIPT
```

3.4 Variables fichier fixes

Les **variables fichier fixes** existent pour chacun des fichiers d'une requête. Leur valeur peut être lue mais non modifiée directement par la requête.

```
$LINE (fichiers d'impression uniquement)
$RECORD-LENGTH (autres que fichiers d'impression)
$RECORD-NUMBER (autres que fichiers d'impression)
```

Une variable fichier fixe peut être qualifiée par un nom de fichier si nécessaire. La valeur implicite pour $LINE est PRINTER ; celle pour $RECORD-LENGTH est le nom du fichier lu par l'instruction READ en cours d'exécution, sauf ci ce nom est attribué indirectement (se reporter à l'instruction READ, au chapitre 7). A noter que $RECORD-NUMBER n'a pas de valeur implicite.

3.5 Variables fichier modifiables

Les **variables fichier modifiables** existent pour chacun des fichiers d'impression d'une requête et leur valeur peut être modifiée par la requête. A noter qu'elles peuvent uniquement être lues par les commandes IQS.

```
@COLUMN-BORDER
$COLUMN-SPACING
@LINE-BORDER
$MAX-PAGES
$PAGE
$PAGE-HEIGHT
$PAGE-WIDTH
$STARTING-DETAIL
```

L'utilisateur peut modifier les valeurs implicites au moyen des clauses de description d'état appropriées en mode procédural, ou au moyen des options équivalentes du processeur d'états.

Une variable fichier modifiable peut être qualifiée par un nom de fichier si nécessaire. Le nom de fichier implicite est PRINTER.

3.6 Liste et description des variables système (A-Z)

@ACTION	Générale, fixe ; CHARACTER 2. Elle permet à l'utilisateur de définir le code à utiliser sur la première ligne d'une grille dans le champ action (pour une description plus complète, se reporter à l'instruction ACCEPT).
@CHARn	Permanentes ; CHARACTER 32 ; n = 1 à 10. Elles permettent de transmettre des informations d'une requête à l'autre ou d'une requête à une commande IQS. Elles sont remises à blanc au début de chaque session IQS.
@COLUMN-BORDER	Fichier, modifiable ; CHARACTER 1 ; valeur implicite : espace. Elle indique le caractère à utiliser pour constituer les lignes verticales d'un état IQS (séparateur de colonnes). Sa valeur implicite peut être modifiée au moyen de la clause COLUMN BORDER du processeur d'états.
$COLUMN-SPACING	Fichier, modifiable ; BINARY 15 ; valeur implicite : 1 ; valeur maximum : 255. Elle indique le nombre d'espaces à insérer entre les colonnes (zones) d'un tableau lorsque la clause COLUMN n'a pas été utilisée dans l'instruction PRINT. Sa valeur peut être modifiée au moyen de la clause COLUMN SPACING du processeur d'états.
@CURRENCY-SIGN	Générale, modifiable : CHARACTER 1 ; valeur implicite : $. Elle indique le symbole monétaire à utiliser pour l'édition lorsque les instructions LET, MODIFY ou PRINT spécifient une clause PICTURE contenant le symbole $.
@CURRENT-VIEW	Générale, fixe ; CHARACTER 30. Sa valeur est soit le nom de la vue, du schéma ou de la structure sélectionné par une commande SELECT, soit entièrement à espaces.
$DATE	Générale, fixe ; UNSIGNED UNPACKED DECIMAL 6. Elle contient la date du jour sous la forme AAMMJJ.
@DATE	Générale, fixe ; CHARACTER 12. Elle contient la date du jour sous la forme "MMM JJ, AAAA".
$DAY	Générale, fixe ; UNSIGNED UNPACKED DECIMAL 2. Elle contient le quantième du mois sous la forme JJ.
@DAY	Générale, fixe ; CHARACTER 16. Elle contient le nom du jour de la semaine.

#DECn	Permanentes ; PACKED SIGNED 15,2 ; n = 1 à 10. Elles permettent de transmettre des informations d'une requête à l'autre ou d'une requête à une commande IQS. Elles sont remises à zéro au début de chaque session IQS.
@EXEC-MODE	Générale, fixe ; CHARACTER 3. Elle contient une valeur qui indique le mode de traitement : IOF pour interactif, BTC pour traitement par lots et TDS pour transactionnel.
$FUNCTION-KEY	Générale, modifiable ; UNSIGNED UNPACKED DECIMAL 1 ; valeur implicite : 1. Elle sert à valider ou invalider les fonctions correspondant aux demandes <, >,?, /, etc. utilisables dans le contexte des instructions ACCEPT, ALTER et DISPLAY. Lorsque sa valeur est zéro, les fonctions sont invalidées.
$LINE	Fichier, fixe ; BINARY 31. Elle contient le numéro de ligne courant de la page de l'état. Elle est remise à zéro au début de chaque page.
@LINE-BORDER	Fichier, modifiable ; CHARACTER 1 ; valeur implicite : espace. Elle contient le caractère à utiliser pour constituer les lignes horizontales d'un état IQS (haut, bas et séparation entre les en-têtes de colonnes et les lignes détails). Sa valeur implicite peut être modifiée au moyen de la clause LINE BORDER du processeur d'états.
$MAX-CPU	Permanente ; UNSIGNED UNPACKED DECIMAL 8 ; valeur implicite : 86399000. Elle spécifie le temps CPU maximum autorisé jusqu'à la fin de la session IQS en cours (en millisecondes). Pas disponible sous TDS.
@MAX-CPU	Générale, fixe ; CHARACTER 12. Elle spécifie le temps CPU maximum autorisé jusqu'à la fin de la session IQS en cours, sous la forme HH:MM:SS:ccc. Pas disponible sous TDS.
$MAX-CPU-PER-COMMAND	Permanente ; UNSIGNED UNPACKED DECIMAL 8. Elle spécifie le temps CPU maximum autorisé pour l'exécution d'une commande ou d'une requête (en millisecondes). Pas disponible sous TDS.

$MAX-ELAPSE	Générale, fixe ; UNSIGNED UNPACKED DECIMAL 8. Elle spécifie la durée maximum autorisée pour la session IQS (en millisecondes). La valeur de cette variable peut être spécifiée dans le paramètre ELAPTIME de la commande de lancement d'IQS.
@MAX-ELAPSE	Générale, fixe ; CHARACTER 12. Elle spécifie la durée maximum autorisée pour la session IQS, sous la forme HH:MM:SS:ccc.
$MAX-IO	Permanente ; BINARY 31 ; valeur implicite : 1000000000. Elle spécifie le nombre maximum d'entrées/sorties IQS autorisées pendant la session IQS en cours.
$MAX-IO-PER-COMMAND	Permanente ; BINARY 31. Elle spécifie le nombre maximum d'entrées/sorties IQS autorisées pour chaque commande ou requête exécutée.
$MAX-LINES	Permanente ; BINARY 31 ; valeur implicite : 1000000000. Elle spécifie le nombre maximum de lignes imprimables jusqu'à la fin de la session IQS en cours.
$MAX-LINES-PER-COMMAND	Permanente ; BINARY 31. Elle spécifie le nombre maximum de lignes imprimables pour chaque commande ou requête exécutée.
$MAX-PAGES	Fichier, modifiable ; BINARY 31 ; valeur implicite : pas de maximum. Elle indique le nombre maximum de pages à imprimer. Sa valeur implicite peut être modifiée au moyen de la clause NUMBER OF PAGES du processeur d'états.
@MESSAGE	Générale, modifiable ; CHARACTER 75. Elle contient un message à visualiser sur la dernière ligne de l'écran (voir l'instruction ACCEPT). Let @Message ="SYSOPT DECIMAL POINT IS COMMA" Let @Message ="READVAR Gcl-var-name [(index)]" Let @Message ="MODVAR Gcl-var-name [(index)]text" READVAR affecte le résultat dans la variable et en modifie le contenu. Voir Annexe C.
$MONTH	Générale, fixe ; UNSIGNED UNPACKED DECIMAL 2. Elle contient le numéro du mois dans l'année sous la forme MM.

@MONTH	Générale, fixe ; CHARACTER 16. Elle contient le nom du mois.
@NEXT-QUERY	Générale, modifiable ; CHARACTER 255. Elle indique le nom de la requête suivante. Sous IOF, elle permet de demander l'exécution d'une autre requête à la suite de la requête courante, avec transmission éventuelle de valeurs de paramètres. Sous TDS, elle permet de demander le chaînage de la requête courante soit avec une autre requête (avec transmission de paramètres), soit avec une routine TPR (sans transmission de paramètres). Pour plus de détails sur l'utilisation de cette variable système sous TDS, se reporter au guide utilisateur IQS-V4/TDS (81UR). La variable @NEXT-QUERY est remise à blanc au lancement de chaque requête principale.
@NEXT-SCRIPT	Générale, modifiable ; CHARACTER 255 ; valeur implicite : espaces. Elle permet de demander l'exécution d'un scénario à partir d'une requête. Non applicable en traitement par lots, ni sous TDS.
$NUMn	Permanentes ; BINARY 31 ; n = 1 à 10. Elles permettent de transmettre des informations d'une requête à l'autre ou d'une requête à une commande IQS. Elles sont remises à zéro au début de chaque session IQS.
$PAGE	Fichier, modifiable ; BINARY 31 ; valeur implicite au départ : zéro. Elle contient le numéro de page courant d'un état. Sa valeur est incrémentée au début de chaque nouvelle page. Sa valeur initiale peut être fixée au moyen de la clause de description d'état STARTING PAGE NUMBER ou de l'option NUMERO DE LA PREMIERE PAGE du processeur d'états.
$PAGE-HEIGHT	Fichier, modifiable ; BINARY 15 ; valeur implicite : 56 ; valeur maximum : 255. Elle indique le nombre maximum de lignes que peut contenir une page d'un état. Sa valeur implicite peut être modifiée au moyen du paramètre LINES de la clause PAGE LIMIT du processeur d'états.

$PAGE-WIDTH	Fichier, modifiable ; BINARY 15 ; valeur implicite : fonction de l'appareil utilisé ; valeur maximum : 255. Elle indique le nombre maximum de caractères que peut contenir une ligne d'un état. Sa valeur implicite peut être modifiée au moyen de la clause PAGE WIDTH du processeur d'états.
$PRIORITY	Permanente ; BINARY 15 ; valeur implicite : 0. Elle spécifie la priorité d'exécution relative de la session IQS en cours. La valeur de $PRIORITY s'ajoute à la priorité GCOS 7 en vigueur pour l'exécution des travaux. Il n'est donc pas autorisé de spécifier une priorité supérieure à celle définie au niveau de GCOS 7, mais uniquement une priorité inférieure. Plage de valeurs autorisée : 0 à 3, 0 étant la priorité la plus élevée. Non disponible sous TDS.
@PROJECT	Générale, fixe ; CHARACTER 12. Elle contient le nom de projet GCOS 7 de l'utilisateur .
$RECORD-LENGTH	Fichier, fixe ; BINARY 15. Elle contient la longueur du dernier article lu par une instruction READ. Elle ne peut être utilisée que dans le contexte d'une instruction READ. Le nom de fichier qui qualifie implicitement cette variable est celui spécifié par la dernière instruction READ correctement exécutée. Il peut être remplacé par le nom de l'un des fichiers lus par les instructions READ précédentes.
$RECORD-NUMBER	Fichier, fixe ; BINARY 31 ; pas de valeur implicite. Elle permet de spécifier un numéro d'article dans un fichier relatif UFAS.
@SLLIB	Générale, fixe ; CHARACTER 58. Elle contient le nom de fichier externe de la bibliothèque origine.
@STARTING-DETAIL	Fichier, modifiable ; BINARY 31. Elle indique un entier qui définit le numéro de la première page après l'en-tête de l'état. La valeur peut être modifiée avec l'option STARTING PAGE DETAIL d'un état IQS.
@STATUS	Générale, fixe ; CHARACTER 8. Elle contient le code retour résultant de l'opération précédente. Sa valeur peut être testée au moyen d'une instruction IF ; dans ce cas, les messages d'erreur sont supprimés en sortie. La liste des valeurs de cette variable est fournie au paragraphe suivant.

$TIME	Générale, fixe ; UNSIGNED UNPACKED DECIMAL 8. Elle contient l'heure sous la forme HHMMSSCC, CC représentant les centièmes de seconde.
@TIME	Générale, fixe ; CHARACTER 8. Elle contient l'heure sous la forme HH:MM:SS.
@TRIGGER-CODE-ERROR	Permanente ; CHARACTER 8 ; valeur implicite : "00000000". Chaîne de 8 caractères déterminant le type de traitement à effectuer automatiquement lorsque les limites définies pour une commande sont atteintes. Chacune des huit positions binaires a une signification particulière.
@TRIGGER-ERROR	Permanente ; CHARACTER 30 ; valeur implicite : espaces. Elle spécifie une requête à activer automatiquement lorsque les limites définies pour une commande sont atteintes.
@TTYPE	Générale, fixe ; CHARACTER 8. Elle contient le type du terminal.
$USED-CPU	Générale, fixe ; BINARY 31. Elle contient le temps CPU utilisé depuis le début de la session IQS en cours (en millisecondes).
@USED-CPU	Générale, fixe ; CHARACTER 12. Elle contient le temps CPU utilisé depuis le début de la session IQS en cours sous la forme HH:MM:SS:ccc.
$USED-ELAPSE	Générale, fixe ; BINARY 31. Elle contient le temps écoulé depuis le début de la session IQS en cours (en millisecondes).
@USED-ELAPSE	Générale, fixe ; CHARACTER 12. Elle contient le temps écoulé depuis le début de la session IQS en cours sous la forme HH:MM:SS:ccc.
$USED-IO	Générale, fixe ; BINARY 31. Elle contient le nombre actuel d'E/S IQS depuis le début de la session IQS en cours.
$USED-IO-PER-COMMAND	Générale, fixe ; BINARY 31. Elle contient le nombre d'E/S IQS effectuées depuis le lancement de l'exécution de la commande ou de la requête courante.

$USED-LINES	Générale, fixe ; BINARY 31. Elle contient le nombre de lignes imprimées depuis le début de la session IQS en cours.
$USED-LINES-PER-COMMAND	Générale, fixe ; BINARY 31. Elle contient le nombre de lignes imprimées depuis le lancement de l'exécution de la commande ou de la requête courante.
@USER	Générale, fixe ; CHARACTER 12. Elle contient le nom de l'utilisateur déclaré pour cette session IQS.
$YEAR	Générale, fixe ; UNSIGNED UNPACKED DECIMAL 4. Elle contient l'année sous la forme AAAA.

3.7 Utilisation et valeurs de @STATUS

A l'exécution d'une requête, certaines erreurs conduisent le système à émettre des messages. Il est possible de supprimer l'émission de ces messages en prévoyant un test de la variable système @STATUS par une instruction IF commençant comme suit :

```
IF @STATUS [opérateur relationnel] valeur-status
  THEN
  .
  .
END
```

L'opérateur relationnel doit être = ou ^=. S'il est omis, la valeur implicite est =.

valeur-status peut être l'une des valeurs indiquées dans le tableau ci-après.

Il suffit que @STATUS contienne la valeur spécifiée au moins une fois au cours de la requête pour que le message correspondant soit supprimé pendant le reste de son exécution.

La variable @STATUS évolue à chaque instruction sauf DO premier format, EXIT et REPEAT. Elle prend la valeur DONE lorsque l'instruction a été exécutée correctement.

Des exemples sont fournis dans le manuel du programmeur IQS/V4 (79UR).

Dans le tableau qui suit, la deuxième colonne indique les instructions pouvant être à l'origine de l'erreur (et par conséquent forçant la variable @STATUS à la valeur correspondante) et la troisième colonne indique la cause de l'erreur.

Tableau 3-1. Valeurs de STATUS (1/2)

Valeur de @STATUS	Instructions concernées	Cause de l'erreur
ABSREC	RETRIEVE	Il n'existe pas d'occurrence maîtresse pour un article détail dont la clause INSERTION est MANUAL.
CHKFAIL	INSERT MODIFY	Une zone spécifiée dans INSERT ou MODIFY ne satisfait pas au contrôle demandé dans la définition du schéma par la clause CHECK.
DATALIM	READ RETRIEVE	Il n'y a plus d'articles satisfaisant à la sélection. @STATUS est forcé à la valeur DATALIM en fin de boucle (après le verbe END correspondant). La valeur DATALIM ne provoque pas l'émission d'un message.
DATAOV	Toutes les instructions contenant des expressions arithmétiques	Il s'est produit un dépassement de capacité à la suite d'une opération arithmétique ou bien une perte de chiffres significatifs à la suite d'une conversion ou d'une affectation de valeur.
DIVZR	"	Une expression arithmétique contient une division par zéro.
DONE	ACCEPT ALTER CONNECT CREATE DELETE DISCONNECT DISPLAY DO FOOTING HEADING INSERT LET MODIFY PRINT READ RECONNECT RETRIEVE REWIND SORT SPACE WRITE	L'instruction a été correctement exécutée.
FILEMPTY	READ REWIND SORT	Le fichier d'entrée existe, mais il est vide.

Tableau 3-2. Valeurs de STATUS (2/2)

Valeur de @STATUS	Instructions concernées	Cause de l'erreur
FILENCR	READ REWIND SORT	Le fichier d'entrée n'a pas été créé (fichiers temporaires uniquement).
FUNCNAV	CHECKPOINT COMMIT RESTART ROLLBACK	Tentative d'exécution d'une instruction COMMIT ou ROLLBACK sous TDS ou d'une instruction CHECKPOINT ou RESTART sous IOF. Se reporter au guide utilisateur IQS-V4/TDS (81UR).
ILLCTR	INSERT MODIFY	La valeur de la zone spécifiée dans une clause OCCURS...DEPENDING ON... est incorrecte.
ILLDATA	LET RETRIEVE Toute instr. traitant des données décimales.	Des zones spécifiées dans l'expression contiennent des caractères non numériques (erreur détectée au cours d'une conversion) ou bien une zone de type décimal contient des caractères non décimaux (par exemple, des espaces).
KEYUNKN	RETRIEVE	Il n'existe pas d'article comportant la clé spécifiée. KEYUNKN ne provoque pas l'émission d'un message.
MAXCTR	ACCEPT ALTER DISPLAY PRINT	L'indice spécifié pour une zone est supérieur au nombre maximum d'occurrences de cette zone (défini par la clause OCCURS...DEPENDING ON...).
MEMBERS	DELETE	L'article possède des articles détails.
NODUP	INSERT MODIFY	Les clés en double ne sont pas autorisées pour le fichier dont l'article spécifié fait partie (UFAS).
NODUP1	INSERT MODIFY	Les clés CALC en double ne sont pas autorisées (IDS/II).
NOTCON	DISCONNECT	L'article ne fait pas partie de l'occurrence d'ensemble spécifiée.
RECCON	CONNECT	L'article est déjà inclus dans l'occurrence d'ensemble spécifiée.
RECNFD	RETRIEVE	L'occurrence détail n'a pas d'occurrence maîtresse. Cette erreur ne se produit qu'avec un fichier séquentiel indexé. Elle peut survenir par exemple lorsqu'aucune occurrence de UNIVERSITE ne correspond à une occurrence de COLLEGE (voir Annexe A).

4. Commandes de l'éditeur de texte IQS

Ce chapitre fournit une description détaillée des commandes de l'éditeur de texte IQS qui constituent un sous-ensemble de l'éditeur de texte standard de GCOS 7, et sont utilisables directement au niveau C: et au niveau V:.

Si l'utilisateur souhaite disposer de la totalité des commandes de l'éditeur de texte GCOS 7, il peut appeler ce dernier sous IQS, au moyen de la commande EDIT. Il peut alors se servir du jeu complet de commandes décrit dans le manuel consacré à Text Editor (05UP) ; il peut également travailler avec l'assistance de textes HELP.

Les commandes A (Adjonction), AUTO, I (Insertion) et C (Remplacement) placent l'utilisateur en mode saisie pour l'introduction de lignes nouvelles. Pour quitter le mode saisie et revenir au niveau commande, il suffit de frapper le caractère barre oblique ("/") au début d'une ligne.

En mode saisie, les commandes IQS ne sont pas traitées comme des commandes mais comme des données d'entrée. La seule exception à cette règle est la commande d'interruption (BREAK) qui permet de quitter le mode saisie pour revenir au niveau commande.
On trouvera dans les paragraphes qui suivent un certain nombre de concepts de base propres aux éditeurs de texte, puis une description des commandes utilisables à l'intérieur d'IQS.

4.1 Adressage dans l'espace de travail origine

L'espace de travail origine est constitué d'un certain nombre de lignes identifiées par un numéro. Les commandes de l'éditeur de texte peuvent opérer sur une seule ligne ou sur un ensemble de lignes identifié par les numéros de la première et de la dernière ligne. Les trois méthodes d'adressage sont les suivantes :

- adressage par numéro de ligne
- adressage relatif
- adressage par contexte.

4.1.1 Adressage par numéro de ligne

Chaque ligne de l'espace de travail peut être identifiée par un numéro unique de la forme :

```
nnnnn.ddd
```

nnnnn est un entier (5 chiffres décimaux au maximum) spécifiant le numéro de ligne absolu. S'il est unique dans l'espace de travail, il suffit pour identifier la ligne.

ddd est un entier (3 chiffres décimaux au maximum) à utiliser seulement lorsque le numéro absolu de la ligne n'est pas unique. Cet entier spécifie un numéro relatif, à savoir une position relative par rapport à la première ligne portant le numéro absolu (ligne nnnnn ou nnnnn.0).

Exemples de numéros de ligne : 10, 10.1, 11.625, 100, 150, 0.1, 0.656. Les zéros non significatifs peuvent être omis ; ainsi 00064, 064, 64.0 spécifient tous la même ligne.

A noter que les caractères accent circonflexe (^), point (.) et dollar ($) permettent respectivement de désigner la première ligne, la ligne courante et la dernière ligne de l'espace de travail sans citer leur numéro. Cette particularité est utilisable pour l'adressage relatif analysé dans le paragraphe qui suit.

4.1.2 Adressage relatif

Au lieu d'identifier une ligne directement par son numéro, l'utilisateur peut spécifier son adresse relative en donnant sa position par rapport à une autre ligne. Cette méthode est nécessaire pour les lignes dont le numéro absolu n'est pas unique (voir ci-dessus), mais elle peut également s'appliquer à toutes les lignes de l'espace de travail.

Le format d'une adresse relative est le suivant :

```
               { + }
[numéro-ligne] {   }  position-relative
               { - }
```

numéro-ligne est la ligne de référence.
position-relative est un entier compris entre 0 et 32000 ; il spécifie un nombre de lignes à compter de la ligne de référence. Le signe + ou - indique le sens du déplacement (progression ou régression).
Si numéro-ligne est omis, la valeur implicite est la ligne courante (voir ci-après).

Exemples d'adressage relatif :

`10+3`	(3ème ligne après la ligne 10)
`25-1`	(ligne précédant la ligne 25)
`^+5`	(5ème ligne après la première ligne de l'espace de travail)
`$-3`	(3ème ligne avant la dernière ligne de l'espace de travail)
`.+6`	(6ème ligne après la ligne courante)
`.-2`	(2ème ligne avant la ligne courante)

4.1.3 Adressage par contexte

L'adressage par contexte s'effectue en spécifiant un critère, à savoir une chaîne de caractères à rechercher. Ce critère identifie la première ligne qui contient la chaîne considérée.

La recherche de la chaîne s'effectue depuis la ligne suivant celle spécifiée jusqu'à la dernière ligne de l'espace de travail, puis de la première ligne de l'espace à la ligne spécifiée. En d'autres termes, elle s'effectue de la ligne spécifiée +1 à la ligne $, puis de la ligne ^ à la ligne spécifiée.

Le format du critère de recherche est le suivant :

```
[numéro-ligne] /critère/
```

La chaîne est une combinaison quelconque de caractères EBCDIC délimitée par des barres obliques. Si le numéro de ligne est omis, la valeur implicite est la ligne courante. Si une barre oblique fait partie intégrante de la chaîne, sa signification de délimiteur doit être annulée en la faisant précéder des caractères [C. Ainsi, /[C/*/ spécifie la chaîne /*. La barre oblique peut également être remplacée par un autre délimiteur, un point d'interrogation par exemple ; ainsi, ?/*? spécifie également la chaîne /*.

L'éditeur de texte mémorise le dernier critère spécifié, ce qui permet à l'utilisateur de le réemployer en frappant simplement "//" (critère vide).

Exemples d'adressage par contexte :

```
10/LIGNE/
```

première ligne contenant la chaîne "LIGNE" après la ligne 10.

```
/LIGNE/
```

première ligne contenant la chaîne "LIGNE" après la ligne courante.

Après exécution correcte d'une commande, la dernière ligne spécifiée dans l'adressage devient généralement la ligne courante.
Dans la description des commandes, qui constitue la deuxième partie de ce chapitre, la position de la ligne courante en fin d'exécution sera précisée à chaque fois.

Après la commande LOAD, la ligne courante est la dernière ligne de l'unité de bibliothèque origine chargée dans l'espace de travail. Il peut arriver que la ligne courante ne soit pas définie (par exemple, lorsque l'espace de travail est vide).

Si l'exécution d'une commande est incorrecte, la ligne courante reste inchangée.

Le caractère point (.) identifie la ligne courante. Lorsqu'il est suivi d'une adresse relative ou d'un critère, il peut être omis.

EXEMPLES :

```
.+1         }
. 1         }   ligne suivant la ligne courante
+1          }

.-1         }
            }   ligne précédant la ligne courante
-1          }

/LINE/      }   première ligne contenant "LIGNE" après
            }   la ligne courante
```

❑

4.2 Combinaison des méthodes d'adressage

Les trois techniques d'adressage décrites ci-dessus peuvent être combinées selon les formats suivants :

```
[numéro-ligne][position-relative][/critère/]
```

ou

```
[numéro-ligne][/critère/][position-relative]
```

numéro-ligne, position-relative et critère ont été décrits précédemment. Pour simplifier, dans la suite du chapitre, ces deux formats d'adressage seront désignés sous l'appellation abrégée "numéro-ligne".

EXEMPLES :

Si la ligne 10 est la ligne courante :

	(l'absence d'adressage équivaut à la ligne 10)
`/LIGNE/+4`	(4ème ligne après la première ligne contenant "LIGNE" à la suite de la ligne courante)
`20+5/LIGNE/`	(première ligne contenant "LIGNE" après la 5ème ligne suivant la ligne 20)

❑

4.3 Désignation de la ligne courante

Lorsqu'une ligne de commande n'est constituée que d'une adresse (sous l'une des formes indiquées au paragraphe précédent), l'éditeur se positionne sur la ligne de texte ainsi spécifiée, elle devient la ligne courante et son contenu est visualisé.

Par exemple :

la ligne de commande :

```
/DEBUT/
```

A l'exécution, l'éditeur localise une ligne contenant la chaîne DEBUT ; cette ligne devient la ligne courante et il en visualise le contenu.

4.4 Mode commande

Les commandes de l'éditeur de texte ne sont admises qu'en mode commande. En environnement interactif (IOF), l'utilisateur est informé qu'il se trouve à ce niveau par le caractère de guidage C:, ou par le caractère V: s'il a sélectionné un schéma, une vue ou une structure (au moyen d'une commande SELECT).
En environnement traitement par lots, il n'y a pas de guidage et l'utilisateur doit s'assurer du bon séquencement des commandes.

4.5 Mode saisie

Les commandes A (Append), AUTO, I (Insert) et C (Change) placent l'utilisateur en mode saisie pour l'introduction de lignes nouvelles.
En interactif (IOF), le guidage :

```
numéro-ligne :
```

indique à l'utilisateur qu'il est dans ce mode. Pour le quitter, il suffit de frapper le caractère barre oblique ("/") au début d'une ligne. Les lignes blanches transmises sont enregistrées et numérotées dans l'espace de travail.

En traitement par lots, chaque article introduit après A (Append), AUTO, I (Insert) ou C (Change) est traité comme une ligne d'entrée tant qu'un caractère barre oblique en première position et seul sur la ligne n'a pas été détecté.

4.6 Format général d'une commande

Le format général d'une commande de l'éditeur de texte est le suivant :

```
[numéro-ligne-1] [, numéro-ligne-2] commande éditeur [paramètres]
```

Certaines commandes ne nécessitent aucun numéro de ligne ; d'autres en exigent un ou deux, mais l'utilisateur peut les omettre. L'éditeur de texte affecte alors automatiquement une valeur implicite aux numéros de ligne manquants. Ces valeurs sont décrites ci-dessous.

4.6.1 Adresses implicites

Dans le format de commande ci-dessus, la valeur implicite de numéro-ligne-2 est numéro-ligne-1 et celle de numéro-ligne-1 est la ligne courante.
Ainsi, pour les commandes exigeant deux numéros de ligne, les formats suivants sont équivalents :

```
numéro-ligne-1 commande éditeur
```

et :

```
numéro-ligne-1, numéro-ligne-1 commande éditeur
```

De même, les formats suivants :

```
commande éditeur
```

et

```
ligne-courante, ligne-courante commande éditeur
```

Pour les commandes ne nécessitant qu'un seul numéro de ligne, les formats suivants sont équivalents :

```
commande éditeur
```

et :

```
ligne-courante commande éditeur
```

4.7 Utilisation des espaces dans les commandes

En général, les espaces figurant avant le nom de la commande, entre ce dernier et le premier paramètre, entre les paramètres et après le dernier paramètre ne sont pas pris en compte. L'utilisateur peut donc, au choix, insérer un ou plusieurs espaces ou ne pas en mettre. Les espaces sont interdits à l'intérieur des noms de commande et des numéros de ligne absolus. Par exemple, AU TO et 11 0 sont tous deux incorrects. Les espaces à l'intérieur d'une chaîne de caractères (entre barres obliques) sont significatifs. Ainsi, la chaîne /X Y Z/ contient 5 caractères et est différente de /XYZ/ qui en contient 3.

Il ne peut y avoir qu'une seule commande par ligne (en interactif) ou par article (en traitement par lots).

Une commande ne peut déborder sur la ligne suivante (en interactif) ou sur l'article suivant (en traitement par lots).

4.8 Bibliothèques

4.8.1 Bibliothèque origine

La bibliothèque origine est une bibliothèque standard (type SL) contenant les objets en langage origine (requêtes, macros, scénarios, descriptions d'état). Lorsqu'une unité de bibliothèque est créée, IQS lui affecte automatiquement un type en fonction de l'objet spécifié dans la commande (par exemple, SAVE QUERY, SAVE MACRO, SAVE SCRIPT, REPLACE REPORT).

Les différents types possibles sont indiqués ci-dessous.

Texte origine	Type
macro	QRM
requête	QRY
description d'état	QRP
scénario	SCR

REMARQUE :

Lorsqu'il emploie EDIT ou FSE (éditeur plein écran) à partir d'IQS pour créer une unité de bibliothèque, l'utilisateur doit s'assurer que le type de langage approprié est spécifié dans la commande W de EDIT ou dans la commande CREATE de FSE.

Lorsque le contenu de l'espace de travail origine est rangé dans la bibliothèque origine, la correspondance entre les numéros de ligne s'établit comme suit :

- les lignes de l'espace de travail identifiées par leur seul numéro absolu (sans numéro relatif) reçoivent ce même numéro dans la bibliothèque.
- les lignes identifiées par un numéro relatif (différent de 0) reçoivent le numéro zéro dans la bibliothèque.

Inversement, lorsque le contenu d'une unité de bibliothèque origine est chargé dans l'espace de travail origine, la correspondance s'établit comme suit :

- les lignes de la bibliothèque dont le numéro est différent de zéro reçoivent ce même numéro dans l'espace de travail.
- les lignes portant le numéro zéro reçoivent le numéro absolu de la dernière ligne chargée complété par un numéro relatif. La valeur initiale de ce numéro relatif est 1 et elle est incrémentée de 1 par ligne tant qu'une ligne de numéro différent de zéro n'est pas détectée dans la bibliothèque.
  Si la bibliothèque contient plus de 999 lignes successives avec le numéro zéro, une renumérotation doit être effectuée avant chargement. Le tableau 4-1 ci-après donne un exemple de correspondance entre numéros de ligne de l'espace de travail et de la bibliothèque.

Tableau 4-1. Correspondance entre numéros de ligne

Espace de travail origine	Bibliothèque origine
0.1	0
0.2	0
0.3	0
1	1
2	2
3	3
10	10
10.1	0
10.2	0
10.3	0
10.4	0
12	12
13	13
13.1	0
20	20
30	30

4.8.2 Bibliothèque résultante

La bibliothèque résultante est une bibliothèque binaire standard (type BIN). Chaque requête, macro ou description d'état compilée est rangée dans une unité résultante portant le même nom que l'unité origine correspondante. IQS assure la cohérence entre version origine et version résultante d'un même objet.

Les requêtes utilisables sous TDS sont rangées, après compilation, dans une bibliothèque standard d'unités compilées (type CU). Pour plus de détails, se reporter au guide utilisateur IQS-V4/TDS.

4.9 Commandes de l'éditeur de texte IQS

Les commandes de l'éditeur de texte utilisables à l'intérieur d'IQS sont présentées dans les pages qui suivent, en ordre alphabétique.

4.9.1 A (APPEND)

Fonction :

Adjonction d'un ensemble de lignes après une ligne spécifiée.

Format :

```
[<numéro-ligne>] A
```

Description de la commande :

Cette commande permet d'ajouter un ensemble de lignes après une ligne spécifiée. Cette ligne peut être identifiée par un numéro de ligne ou être la ligne courante. Après exécution de la commande, l'utilisateur se trouve en mode saisie et les nouvelles lignes sont automatiquement numérotées.

Pour quitter le mode saisie et revenir au niveau commande, il suffit de frapper le caractère barre oblique ("/") au début d'une ligne.
Les nouvelles lignes reçoivent des numéros absolus non attribués compris entre les numéros des deux lignes existantes qui bornent l'insertion. Une fois ces numéros absolus épuisés, des numéros relatifs viennent compléter le dernier numéro absolu attribué. Cette opération peut provoquer un décalage de la numérotation relative des lignes existantes. L'adjonction ne peut modifier le numéro absolu des lignes existantes.

Si l'exécution de la commande est correcte, la ligne courante est la dernière ligne ajoutée (avant sortie du mode saisie). Si une erreur est détectée dans la commande ou si aucune ligne n'est ajoutée, la ligne courante reste la même.

Si l'espace de travail origine est vide, $A ou A est équivalent à AUTO 10,10.

Description du paramètre :

numéro-ligne	Ligne après laquelle les nouvelles lignes sont à ajouter. Si ce paramètre est spécifié, il doit renvoyer à une ligne existante (par l'une des méthodes d'adressage autorisées). S'il est omis, la ligne courante (si elle est définie) ou le début de l'espace de travail (s'il est vide) constitue la valeur implicite.

Exemple 1 :

Soit un espace de travail origine contenant les lignes suivantes :

```
10  :   LIGNE 10
20  :   LIGNE 20
30  :   LIGNE 30
```

La commande :

```
$A
```

peut être utilisée pour ajouter les lignes suivantes :

```
31  :   LIGNE 31
32  :   LIGNE 32
33  :   LIGNE 33
34  :   /
```

La ligne 34 contient le caractère barre oblique permettant de sortir du mode saisie. Elle n'est pas incluse dans l'espace de travail. La ligne courante est maintenant la ligne 33. A noter que les nouvelles lignes portent les numéros absolus supérieurs à 30 qui était le dernier numéro attribué. Avec la commande 10A, les nouveaux numéros de ligne attribués auraient été 11, 12 et 13.

Soit maintenant la commande :

```
20A
```

Les 9 premières lignes ajoutées sont numérotées de 21 à 29. Les 999 lignes suivantes sont numérotées de 29.1 à 29.999. Au-delà, toute adjonction devient impossible et la commande RENUMBER doit être utilisée pour effectuer une renumérotation.

Exemple 2 :

Soit un espace de travail origine contenant les lignes suivantes :

```
10  :   LIGNE 10
11  :   LIGNE 11
12  :   LIGNE 12
```

La commande 11A peut être utilisée pour ajouter des lignes supplémentaires entre les lignes 11 et 12 comme suit :

```
C: 11A

11.1  :   NOUVELLE LIGNE 1
11.2  :   NOUVELLE LIGNE 2
11.3  :   NOUVELLE LIGNE 3
11.4  :   NOUVELLE LIGNE 4
11.5  :   /
```

La ligne 11.5 provoque la sortie du mode saisie. Après cette opération, la ligne 11.4 est la ligne courante et l'utilisateur se trouve au niveau commande.

4.9.2 AUTO (AT)

Fonction :

Remise à l'état initial de l'espace de travail origine, avec affichage d'un numéro de ligne, pour introduction d'informations nouvelles.

Format :

```
 ------------------------------------------
|        {10              }   {10   }       |
| AUTO [ {                } [,{     } ] ]   |
|        {<première-ligne>}  ,{<pas>}       |
 ------------------------------------------
```

Description de la commande :

Cette commande supprime le contenu éventuel de l'espace de travail origine et instaure le mode saisie permettant à l'utilisateur d'introduire de nouvelles informations. Les lignes sont numérotées automatiquement. Pour sortir du mode saisie (et retourner au niveau commande), il suffit de frapper le caractère barre oblique ("/") au début d'une ligne. Avant de supprimer le contenu, l'éditeur de texte commande à l'utilisateur s'il souhaite sauvegarder son espace de travail.

L'utilisateur a la possibilité de spécifier le numéro de la première ligne et le pas de progression. Si l'exécution de AUTO est correcte, la ligne courante est la dernière ligne introduite (avant la frappe du caractère barre oblique). S'il y a une erreur dans la commande, celle-ci n'est pas exécutée. L'ancien contenu de l'espace de travail est conservé (s'il y en a un) et la ligne courante reste la même. Si AUTO est exécutée sans introduction de texte, l'espace de travail est vide et la ligne courante n'est pas définie.

Description des paramètres :

première-ligne	Entier positif spécifiant le numéro de la première ligne à introduire. La valeur maximale est 90000 et la valeur implicite 10.
pas	Entier positif spécifiant le pas de progression (incrément entre deux lignes). La valeur maximale est 1000 et la valeur implicite 10.

EXEMPLE 1 :

```
AUTO 100,1
❑
```

Cette commande remet l'espace de travail origine à l'état initial avec passage au mode saisie. Le numéro de la première ligne saisie sera 100 et l'incrément sera de 1.

Les lignes suivantes sont introduites :

```
100  :   PREMIERE LIGNE
101  :   DEUXIEME LIGNE
102  :   TROISIEME LIGNE
103  :   QUATRIEME LIGNE
104  :   /
```

Le caractère barre oblique de la ligne 104 n'est pas introduit dans l'espace de travail. La dernière ligne introduite (numéro 103) est la ligne courante.

EXEMPLE 2 :

```
AUTO
❑
```

Cette commande remet l'espace de travail à l'état initial avec passage au mode saisie. Le numéro de la première ligne saisie sera 10 (valeur implicite) et l'incrément sera de 10 (valeur implicite).

Les lignes suivantes sont introduites :

```
10  :   PREMIERE LIGNE
20  :   DEUXIEME LIGNE
30  :   /
```

Le caractère barre oblique de la ligne 30 provoque la sortie du mode saisie et le retour au niveau commande.
La ligne courante est la ligne 20.

4.9.3 C (CHANGE)

Fonction :

Remplacement d'un ensemble de lignes dans l'espace de travail origine.

Format :

```
[<numéro-ligne-1>] [,<numéro-ligne-2>] C
```

Description de la commande :

Cette commande permet de supprimer un ensemble de lignes dans l'espace de travail origine pour le remplacer par un nouvel ensemble.
L'ensemble à supprimer est identifié en spécifiant sa première et sa dernière ligne.
A l'exécution, l'utilisateur se trouve en mode saisie et peut introduire les nouvelles lignes. Pour quitter le mode saisie et revenir au niveau commande, il lui suffit de frapper le caractère barre oblique ("/") au début d'une ligne.
^,$C est équivalent à AUTO 10,10 (sauf que la possibilité de sauvegarder le contenu de l'espace de travail n'existe pas).

Les nouvelles lignes sont automatiquement numérotées en utilisant les numéros des lignes supprimées, jusqu'à épuisement. Une fois tous ces numéros utilisés, le processus est le même que pour la commande Append : les numéros absolus non attribués sont affectés jusqu'à épuisement et les lignes suivantes reçoivent le dernier numéro absolu utilisé complété par un numéro relatif.

Si l'exécution de la commande est correcte, la ligne courante est la dernière ligne insérée (celle précédant le caractère barre oblique). Si aucune ligne n'est insérée, la ligne courante est celle qui suit la dernière ligne supprimée ; si c'est la dernière ligne de l'espace de travail, la dernière ligne définie dans l'espace de travail devient la ligne courante.
Si l'exécution de la commande est incorrecte, la ligne courante reste la même.

La commande Change ne peut être utilisée si l'espace de travail origine est vide (sauf s'il s'agit de la commande ^,$C).

Description des paramètres :

numéro-ligne-1	Première ligne de l'ensemble à remplacer. Si ce paramètre est spécifié, il doit renvoyer à une ligne existante (par une des méthodes d'adressage autorisées). S'il est omis, la ligne courante est la valeur implicite.
numéro-ligne-2	Dernière ligne de l'ensemble à remplacer. Si ce paramètre est spécifié, il doit renvoyer à une ligne existante (par l'une des méthodes d'adressage autorisées) et être supérieur ou égal à numéro-ligne-1. S'il est omis, sa valeur implicite est numéro-ligne-1. Toutes les lignes comprises entre numéro-ligne-1 et numéro-ligne-2 (inclus) sont remplacées.

EXEMPLE :

Soit un espace de travail contenant les lignes suivantes :

```
1    :   LIGNE 1
1.1  :   LIGNE 1.1
1.2  :   LIGNE 1.2
3    :   LIGNE 3
6    :   LIGNE 6
7    :   LIGNE 7
8    :   LIGNE 8
```

❑

la commande :

```
1,1.2C
```

supprime les trois premières lignes (c'est-à-dire 1, 1.1 et 1.2). Les trois premières lignes de remplacement sont numérotées respectivement 1, 1.1 et 1.2. La quatrième ligne de remplacement porte le numéro 2 (premier numéro absolu non attribué). Les lignes de remplacement suivantes portent les numéros 2.1, 2.2, 2.3, etc.

Le même résultat pourrait être obtenu avec les commandes suivantes (si le numéro de la ligne courante est 1) :

```
^,+2C
```

```
^,/3/-1C
```

```
^,$-4C
```

Après exécution de la commande, la ligne courante est la dernière ligne introduite.

4.9.4 D (DELETE)

Fonction :

Suppression d'un ensemble de lignes dans l'espace de travail origine.

Format :

```
[<numéro-ligne-1>] [,<numéro-ligne-2>] D
```

Description de la commande :

Cette commande permet de supprimer une ou plusieurs lignes dans l'espace de travail origine. Si un seul numéro de ligne est spécifié, la ligne correspondante est supprimée. Si deux numéros de ligne sont spécifiés, les deux lignes correspondantes ainsi que toutes les lignes intermédiaires sont supprimées. Si aucun numéro de ligne n'est spécifié, c'est la ligne courante qui est supprimée.

Si l'exécution de la commande est correcte, la ligne courante est celle qui suit la dernière ligne spécifiée. Si c'est la dernière ligne de l'espace de travail qui est supprimée, la dernière ligne définie dans l'espace de travail devient la ligne courante. Si l'exécution de la commande Delete est incorrecte, la ligne courante reste la même.

Description des paramètres :

numéro-ligne-1	Première ligne à supprimer. Si ce paramètre est spécifié il doit renvoyer à une ligne existante (par l'une des méthodes d'adressage autorisées). S'il est omis, la ligne courante est la valeur implicite.
numéro-ligne-2	Dernière ligne à supprimer. Si ce paramètre est spécifié, il doit renvoyer à une ligne existante (par l'une des méthodes d'adressage autorisées) et être supérieur ou égal à numéro-ligne-1. S'il est omis, numéro-ligne-1 est la valeur implicite. Toutes les lignes comprises entre numéro-ligne-1 et numéro-ligne-2 (inclus) sont supprimées.

EXEMPLE :

Soit un espace de travail origine contenant les lignes suivantes :

```
 0.1  :   LIGNE 0.1
 0.2  :   LIGNE 0.2
 3    :   LIGNE 3
10    :   LIGNE 10
12    :   LIGNE 12
12.1  :   LIGNE 12.1
12.2  :   LIGNE 12.2
```

❑

La ligne 10 peut être supprimée par l'une des commandes suivantes :

```
10D
10,10D
3+1D           (première ligne suivant la ligne 3)
/LIGNE 10/D    (première ligne contenant la chaîne spécifiée)
12-1D          (première ligne avant la ligne 12)
$-3D           (troisième ligne avant la dernière ligne)
```

Après la suppression, la ligne courante est la ligne 12 (ligne suivant la dernière ligne supprimée).

Les lignes 12, 12.1 et 12.2 peuvent être supprimées de la manière suivante :

```
,+2D         (ligne courante et les deux suivantes)
12,12.2D     (lignes 12 à 12.2)
12,$D        (de la ligne 12 à la fin de l'espace de travail)
3+1,$D       (de la première ligne suivant la ligne 3 jusqu'à la
              dernière ligne)
$-2,$D       (de la dernière ligne moins 2 à la dernière ligne)
,$D          (de la ligne courante à la dernière ligne)
```

La dernière ligne de l'espace de travail ayant été supprimée (ligne 12.2), la ligne courante devient la ligne 3 (dernière ligne définie).

4.9.5 I (INSERT)

Fonction :

Insertion d'un ensemble de lignes avant une ligne spécifiée.

Format :

```
[<numéro-ligne>] I
```

Description de la commande :

Cette commande permet d'insérer un ensemble de lignes avant une ligne spécifiée. Cette dernière peut être identifiée par un numéro ou être la ligne courante. Après exécution de la commande, l'utilisateur se trouve en mode saisie et les nouvelles lignes sont automatiquement numérotées.
Pour quitter le mode saisie et revenir au niveau commande, il suffit de frapper le caractère barre oblique ("/") au début d'une ligne.

Les nouvelles lignes reçoivent des numéros absolus non attribués compris entre les numéros des deux lignes existantes qui bornent l'insertion. Une fois ces numéros absolus épuisés, des numéros relatifs viennent compléter le dernier numéro absolu attribué. Cette opération peut provoquer un décalage de la numérotation relative des lignes existantes. L'insertion ne peut modifier le numéro absolu des lignes existantes.

Si l'exécution de la commande est correcte, la ligne courante est la dernière ligne insérée (avant sortie du mode saisie). Si une erreur est détectée dans la commande ou si aucune ligne n'est insérée, la ligne courante reste la même.

Si l'espace de travail origine est vide, $I ou I est équivalent à AUTO 10,10.

Description du paramètre :

numéro-ligne	Ligne avant laquelle les nouvelles lignes sont à insérer. Si ce paramètre est spécifié, il doit renvoyer à une ligne existante (par l'une des méthodes d'adressage autorisées). S'il est omis, la ligne courante est la valeur implicite.

EXEMPLE :

Soit un espace de travail origine contenant les lignes suivantes :

```
10  :   LIGNE 10
20  :   LIGNE 20
30  :   LIGNE 30
❑
```

La commande :

```
10I
```
ou
```
^I
```

permet d'effectuer les insertions suivantes :

```
1  :   NOUVELLE LIGNE 1
2  :   NOUVELLE LIGNE 2
3  :   NOUVELLE LIGNE 3
4  :   /
```

La ligne 4 contient le caractère barre oblique permettant de sortir du mode saisie et de retourner au niveau commande. Elle n'est pas incluse dans l'espace de travail. La ligne courante est maintenant la ligne 3.

La commande :

```
1I
```
ou
```
^I
```
permet d'effectuer les insertions suivantes :

```
0.1  :   NOUVELLE LIGNE 0.1
0.2  :   NOUVELLE LIGNE 0.2
0.3  :   LIGNE SPECIALE 0.3
0.4  :   /
```

La ligne courante est maintenant la ligne 0.3.

La commande :

```
0.3I
```
permet d'insérer les lignes suivantes :

```
0.3  :   NOUVELLE LIGNE 0.3
0.4  :   NOUVELLE LIGNE 0.4
0.5  :   NOUVELLE LIGNE 0.5
0.6  :   /
```

La ligne courante est maintenant la ligne 0.5.

Après exécution de ces trois opérations, le contenu de l'espace de travail est le suivant :

```
 0.1  :   NOUVELLE LIGNE 0.1
 0.2  :   NOUVELLE LIGNE 0.2
 0.3  :   NOUVELLE LIGNE 0.3
 0.4  :   NOUVELLE LIGNE 0.4
 0.5  :   NOUVELLE LIGNE 0.5
 0.6  :   LIGNE SPECIALE 0.3
 1    :   NOUVELLE LIGNE 1
 2    :   NOUVELLE LIGNE 2
 3    :   NOUVELLE LIGNE 3
10    :   LIGNE 10
20    :   LIGNE 20
30    :   LIGNE 30
```

REMARQUE :

L'ancienne ligne 0.3 (contenant LIGNE SPECIALE 0.3) devient la ligne 0.6 après l'insertion. Une autre commande ^I permettrait d'insérer une nouvelle ligne 0.1 et les lignes 0.1 à 0.6 existantes seraient décalées et deviendraient respectivement les lignes 0.2 à 0.7.

4.9.6 L (LIST)

Fonction :

Listage d'un ensemble de lignes de l'espace de travail origine.

Format :

```
[<numéro-ligne-1>] [,<numéro-ligne-2>] L
```

Description de la commande :

Cette commande permet de lister une ou plusieurs lignes de l'espace de travail. Si un seul numéro de ligne est spécifié, la ligne correspondante est visualisée. Si deux numéros sont spécifiés, les deux lignes correspondantes ainsi que toutes les lignes intermédiaires sont visualisées.
Si aucun numéro n'est spécifié, c'est la ligne courante qui est visualisée.

Si l'exécution de la commande est correcte, la ligne courante est la dernière ligne visualisée. Si l'exécution est incorrecte, la ligne courante reste la même. Si l'utilisateur effectue une interruption (BREAK) en cours d'exécution, la ligne courante est la dernière ligne visualisée.

Description des paramètres :

numéro-ligne-1	Première ligne à lister. Si ce paramètre est spécifié, il doit renvoyer à une ligne existante (par l'une des méthodes d'adressage autorisées). S'il est omis, la ligne courante est la valeur implicite.
numéro-ligne-2	Dernière ligne à lister. Si ce paramètre est spécifié, il doit renvoyer à une ligne existante (par l'une des méthodes d'adressage autorisées), et être supérieur ou égal à numéro-ligne-1. S'il est omis, numéro-ligne-1 est la valeur implicite. Toutes les lignes comprises entre numéro-ligne-1 et numéro-ligne-2 (inclus) sont visualisées.

EXEMPLE 1 :

Soit un espace de travail origine contenant les lignes suivantes :

```
 0.1  :   LIGNE 0.1
 0.2  :   LIGNE 0.2
 1    :   LIGNE 1
 2    :   LIGNE 2
10    :   LIGNE 10
11    :   LIGNE 11
12    :   LIGNE 12
12.1  :   LIGNE 12.1
12.2  :   LIGNE 12.2
12.3  :   LIGNE 12.3
12.4  :   LIGNE 12.4
```

❑

Ces lignes peuvent être listées au moyen de la commande :

```
^,$L
```

La commande :

```
/LIGNE 11/,/12.2/L
```

permet de lister les lignes suivantes :

```
11    :   LIGNE 11
12    :   LIGNE 12
12.1  :   LIGNE 12.1
12.2  :   LIGNE 12.2
```

La ligne courante est maintenant la ligne 12.2 (la dernière listée).

La commande :

```
+1,+2L
```

ou

```
+1,$L
```

permet de lister les lignes suivantes :

```
12.3  :   LIGNE 12.3
12.4  :   LIGNE 12.4
```

REMARQUE :

A noter l'utilisation de l'adressage relatif, +1 spécifie la ligne courante plus une (c'est-à-dire la ligne suivant la ligne 12.2) et +2 spécifie la ligne courante plus deux (c'est-à-dire la ligne 12.4). La ligne 12.4 étant la dernière ligne, le caractère $ peut également être utilisé pour la désigner. Après exécution de la commande, la ligne courante est la ligne 12.4.

La commande :

```
^,-8L
```

permet de lister les lignes suivantes :

```
0.1  :   LIGNE 0.1
0.2  :   LIGNE 0.2
1    :   LIGNE 1
```

-8 spécifie la huitième ligne avant la ligne courante (qui est la ligne 12.4 avant exécution de la commande). Après exécution, la ligne courante est la ligne 1.

EXEMPLE 2 :

Le contenu de l'espace de travail est le même que dans l'exemple 1.

```
10L
❑
```

permet de lister la ligne suivante :

```
10  :   LIGNE 10
```

La ligne courante est maintenant la ligne 10.

4.9.7 RENUMBER (RB)

Fonction :

Renumérotation des lignes dans l'espace de travail origine.

Format :

```
 ------------------------------------------------
|            { 10             }    { 10  }       |
| RENUMBER [ {                }  [,{     } ] ]   |
|            {<première-ligne>}    {<pas>}       |
 ------------------------------------------------
```

Description de la commande :

Cette commande permet de renuméroter les lignes dans l'espace de travail origine. Elle peut être utilisée en cas de rupture de la numérotation à la suite de suppressions ou d'insertions ou lorsqu'une insertion entre deux lignes existantes devient impossible (plus de 999 numéros relatifs entre deux numéros absolus successifs).

L'utilisateur a la possibilité de spécifier le numéro de la première ligne et le pas de progression.
Si l'exécution de RENUMBER est correcte, la dernière ligne de l'espace de travail devient la ligne courante. Si l'exécution est incorrecte, l'espace de travail n'est pas renuméroté et la ligne courante reste la même.

Description des paramètres :

première-ligne	Entier positif spécifiant le nouveau numéro de la première ligne dans l'espace de travail. La valeur maximale est 90000 et la valeur implicite 10.
pas	Entier positif spécifiant le nouveau pas de progression (incrément entre deux lignes). La valeur maximale est 1000 et la valeur implicite est 10.

EXEMPLE :

Soit un espace de travail contenant les lignes suivantes :

```
10    :   LIGNE A
20    :   LIGNE B
29    :   LIGNE C
29.1  :   LIGNE D
29.2  :   LIGNE E
29.3  :   LIGNE F
29.4  :   LIGNE G
29.5  :   LIGNE H
29.6  :   LIGNE I
30    :   LIGNE J
31    :   LIGNE K
32    :   LIGNE L
33    :   LIGNE M
❑
```

La commande :

```
RENUMBER
```

permet de renuméroter les lignes de la manière suivante :

```
10  :   LIGNE A
20  :   LIGNE B
30  :   LIGNE C
40  :   LIGNE D
.          .
.          .
.          .
00  :   LIGNE J
10  :   LIGNE K
20  :   LIGNE L
30  :   LIGNE M
```

Aucun paramètre n'étant spécifié dans la commande, le numéro de la première ligne est 10 et l'incrément est 10 (valeurs implicites). Après exécution, la ligne 130 devient la ligne courante.

La commande :

```
RENUMBER 1,2
```

permet de renuméroter les lignes comme suit : 1, 3, 5, 7, 9, 11 ....25.

Après exécution, la ligne 25 devient la ligne courante.

4.9.8 S (SUBSTITUTE)

Fonction :

Substitution d'une chaîne de caractères à une autre dans un ensemble de lignes.

Format :

```
|-----------------------------------------------------|
|    [<ligne-1>] [,<ligne-2>] S/<critère>/<chaîne>/   |
|-----------------------------------------------------|
```

Description de la commande :

Cette commande permet de substituer une chaîne spécifiée à une autre chaîne spécifiée dans un ensemble de lignes. L'ensemble est identifié par ses première et dernière lignes. Si aucun numéro de ligne n'est spécifié, la commande s'applique à la ligne courante.

Le nombre total de substitutions effectuées est visualisé.

Si aucune substitution n'est effectuée (si le critère spécifié est introuvable dans l'ensemble de lignes), un message d'avertissement apparaît.

Si l'exécution de la commande est correcte, la ligne courante est la dernière ligne lue. Si l'exécution est incorrecte (erreur ou critère non trouvé), la ligne courante reste la même.

Description des paramètres :

ligne-1	Numéro de la première ligne de l'ensemble. Si ce paramètre est spécifié, il doit renvoyer à une ligne existante (par l'une des méthodes d'adressage autorisées). S'il est omis, la ligne courante est la valeur implicite.
ligne-2	Numéro de la dernière ligne de l'ensemble. Si ce paramètre et spécifié, il doit renvoyer à une ligne existante (par l'une des méthodes d'adressage autorisées) et être supérieur ou égal à ligne-1. S'il est omis, ligne-1 est la valeur implicite. Toutes les lignes comprises entre ligne-1 et ligne-2 (inclus) sont lues pour rechercher le critère spécifié.

critère	Chaîne de caractères à remplacer par <chaîne>. Elle doit être entre barres obliques. Elle peut être vide (//) et prend alors la valeur du critère spécifié dans la commande précédente. Si une barre oblique fait partie intégrante de la chaîne à remplacer, elle doit être précédée des deux caractères [C (par exemple, la chaîne A/B doit s'écrire A[C/B). A l'intérieur de cette chaîne, les majuscules et les minuscules ne sont pas équivalentes ; par exemple, /a b c d/ est différent de /A B C D/. Les espaces sont significatifs ; par exemple /X Y Z/ est différent de /XYZ/.
chaîne	Chaîne de caractères devant remplacer <critère>. Elle doit se terminer par une barre oblique. Elle peut être vide (//) et signifie alors : remplacer <critère> par rien. Si <chaîne> contient le caractère &, ce caractère est remplacé par la chaîne spécifiée par <critère>. La signification spéciale du caractère & peut être annulée en le faisant précéder des deux caractères [C. Toutes les autres remarques faites à propos de <critère> s'appliquent également à <chaîne>.

EXEMPLE :

La ligne courante est la ligne 1 et l'espace de travail contient les lignes suivantes :

```
1  :   ABCDEFGHIJKLMN
2  :   ABCD
3  :   KLMN
4  :   KLMNABCD
5  :   ABCDKLMNABCD
6  :   WXYZABCD
```

❑

La commande :

```
S/ABCD/WXYZ/
```

modifie la ligne 1 et donne le message suivant :

```
1 SUBSTITUTION
```

La nouvelle valeur de la ligne 1 est :

```
1  :   WXYZEFGHIJKLMN
```

Après exécution, la ligne courante est toujours la ligne 1.

La commande :

```
+3,+4S//RST/
```

donne le message suivant :

```
3 SUBSTITUTIONS
```

L'ensemble de lignes dans lequel la substitution est effectuée est réduit aux lignes 4 et 5 (ligne courante +3 et ligne courante +4).
Le critère vide (//) renvoie au critère "ABCD" spécifié dans la commande précédente. Après exécution, la ligne courante est la ligne 5 qui est la dernière ligne de l'ensemble traité. A noter que si la ligne 5 n'avait pas contenu le critère "ABCD", la ligne courante serait quand même la ligne 5 (il y a eu une substitution dans la ligne 4). Le contenu des lignes 4 et 5 est maintenant le suivant :

```
4  :   KLMNRST
5  :   RSTKLMNRST
```

La commande :

```
S/RST/UV
```

donne le message suivant :

```
2 SUBSTITUTIONS
```

La nouvelle valeur de la ligne 5 est maintenant :

```
5  :   UVKLMNUV
```

La ligne courante demeure la ligne 5 car aucun numéro de ligne n'apparaissant dans la commande, la substitution ne s'applique qu'à la ligne courante.

La commande :

```
S/ABC/XYZ/
```

donne le message suivant :

```
Chaîne non trouvée
```

puisque la ligne 5 (ligne courante) ne contient pas la chaîne ABC.

5. Instructions du langage de requêtes IQS (A-D)

5.1 Introduction

Le langage de requêtes est traité dans les chapitres 5 à 7. Le présent chapitre fournit une description des instructions de la lettre A à la lettre D incluse. Cette description dans l'ordre alphabétique est précédée d'une présentation générale.

En ce qui concerne la création de requêtes utilisables sous TDS, se reporter au guide utilisateur IQS-V4/TDS.

Pour tous renseignements complémentaires en matière de programmation, consulter le manuel d'initiation au langage de requêtes et le guide du programmeur.

Une requête comprend un ensemble d'instructions et des commentaires facultatifs.

5.1.1 Format général d'une instruction

En général, une instruction du langage de requêtes se compose des éléments suivants :

- Un nom d'instruction (obligatoire)
- Un ou plusieurs paramètres (facultatifs)
- Un ou plusieurs mots-clés (facultatifs)

PAR EXEMPLE :

```
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
  WHERE E-NOM = "SARRE"
END
❑
```

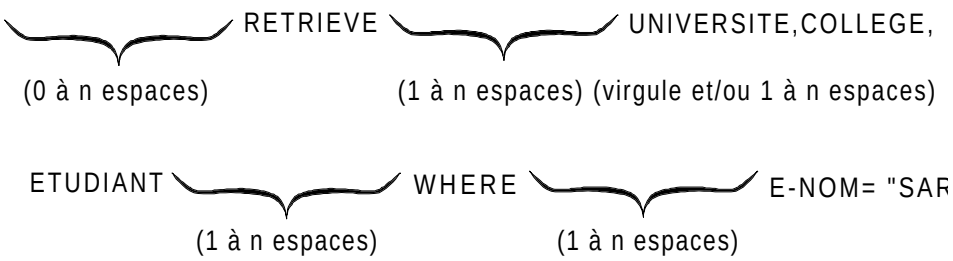
RETRIEVE et END sont des instructions ; WHERE est un mot-clé ; et les articles UNIVERSITE, COLLEGE et ETUDIANT ainsi que la condition E-NOM = "SARRE" sont des paramètres.

5.1.2 Utilisation des virgules

Les séparateurs utilisés dans les listes de paramètres sont la virgule et les espaces. La virgule ne peut figurer qu'aux endroits où elle apparaît dans le format détaillé des instructions. A noter qu'elle est obligatoire dans les instructions DEFINE et PARAMETER lorsque plusieurs éléments sont spécifiés.

5.1.3 Utilisation des espaces

Les espaces (un ou plusieurs) sont obligatoires entre le nom de l'instruction et le premier paramètre et entre le dernier paramètre et les mots-clés qui suivent. Ils sont interdits à l'intérieur d'un nom d'instruction, d'un nom de paramètre ou d'un mot-clé. Le nom d'instruction peut être précédé d'un ou plusieurs espaces. Plusieurs instructions peuvent être spécifiées sur une même ligne d'entrée (en interactif) ou dans un même article d'entrée (en traitement par lots). A l'inverse, une même instruction peut occuper plusieurs lignes ou articles d'entrée, à condition toutefois que le nom d'instruction, un nom de paramètre ou un mot-clé ne soient pas coupés.



5.1.4 Utilisation de décalages dans la présentation

Les espaces à gauche permettent des décalages qui font ressortir les différents blocs et facilitent donc la lecture des programmes.

```
RETRIEVE UNIVERSITE
.
.
.  RETRIEVE COLLEGE
.  .
.  .
.  .  RETRIEVE ETUDIANT
.  .  .
.  .  END
.  .
.  .
.  END
.
END
```

Il est possible d'utiliser la commande TRANSLATE QUERY pour mettre en forme la requête ou le contenu de l'espace de travail courant.

REMARQUE :

Dans ce cas, les commentaires sont perdus. Se reporter à la commande TRANSLATE.

5.1.5 Ecriture d'une instruction sur plusieurs lignes

```
RETRIEVE UNIVERSITE,
   COLLEGE,
   ETUDIANT
      WHERE E-NOM = "SARRE"
```

Le nom d'instruction RETRIEVE, le nom de paramètre ETUDIANT, le mot-clé WHERE, etc. ne peuvent pas être coupés.

5.1.6 Instructions emboîtées

Les emboîtements sont admis dans les instructions suivantes :

- Instructions de contrôle
- CREATE
- DO
- IF
- READ
- RETRIEVE

Le nombre maximum d'emboîtements est fonction de la complexité des différentes instructions utilisées. Il est conseillé de signaler les différents niveaux d'emboîtement par des décalages successifs (voir plus haut).

5.1.7 Eléments du langage associés

Les divers éléments du langage (littéral, clause PICTURE par exemple) utilisés dans la description des instructions et des paramètres sont définis au chapitre 2. Les variables système sont décrites au chapitre 3.

5.1.8 Remarques sur les exemples fournis

Les exemples fournis dans ce chapitre se réfèrent aux schémas de l'annexe A. Sauf indication contraire, ils sont tous compatibles avec les trois schémas, ceux-ci étant similaires. A noter que la zone TYP ne figure pas dans le schéma IDS/II et qu'elle n'occupe pas la même position dans les deux schémas UFAS. Lorsque cette particularité entraîne une différence en sortie, le schéma utilisé comme référence est le schéma UFAS séquentiel.

A noter que, dans les exemples qui accompagnent chaque instruction, les informations à fournir par l'utilisateur sont toujours soulignées.

5.1.9 Description des instructions

Les instructions du langage de requêtes sont traitées dans les chapitre 5 à 7, dans l'ordre alphabétique.

Pour chaque instruction, les rubriques sont les suivantes : fonction, format, description de l'instruction, description des paramètres.

5.2 ACCEPT

Fonction :

Demande d'introduction de données au terminal (soit en mode ligne, soit par l'intermédiaire de grilles).
En traitement par lots, les données sont prises dans COMFILE.

Format 1 :

```
|-----------------------------------------------------------|
|    ACCEPT [WITH CHECKPOINT] <nom-zone>                    |
|                                                           |
|       [PROMPT <expression-alphanumérique>]                |
|                                                           |
|   [,<nom-zone> [PROMPT <expression-alphanumérique>]]...   |
|                                                           |
|      [THRU <nom-grille> [,<nom-grille>]...]               |
|-----------------------------------------------------------|
```

Format 2 :

```
|-------------------------------------------------|
|  ACCEPT [WITH CHECKPOINT]                       |
|                                                 |
|     [<nom-article>[,<nom-article>]...]          |
|                                                 |
|     [THRU <nom-grille> [,<nom-grille>]...]      |
|-------------------------------------------------|
```

Format 3 :

```
|-------------------------------------------------|
|    ACCEPT [WITH CHECKPOINT]                     |
|                                                 |
|       [<nom-fichier>]                           |
|                                                 |
|       [THRU <nom-grille> [,<nom-grille>]...]    |
|-------------------------------------------------|
```

Description de l'instruction :

L'instruction ACCEPT permet de transférer des données introduites au terminal dans des articles et/ou des zones.
Le contenu des articles ou des zones est remplacé par les données introduites.
L'instruction ACCEPT ne modifie que les données se trouvant en mémoire. Les données se trouvant dans un fichier ou dans une base de données ne peuvent être modifiées que par les instructions WRITE, INSERT ou MODIFY.

L'option WITH CHECKPOINT (avec point de reprise) est réservée à TDS.

Les données sont introduites soit au moyen de grilles utilisateur (sur les terminaux plein écran), soit en mode ligne.

INTRODUCTION EN MODE GRILLE :

- Le nombre maximum de grilles par instruction ACCEPT est de 8.
- Il ne doit y avoir qu'une seule grille par écran, autrement dit, il n'est pas possible de visualiser plusieurs grilles à la fois (par adjonction ou superposition).
- Le nombre de lignes d'une grille utilisateur ne doit pas excéder le nombre de lignes de l'écran moins 2 (généralement 22).
- La première ligne de la grille (qui contient -->__) est fournie par IQS ; elle n'a donc pas à être définie dans la grille utilisateur. Lorsque la grille est visualisée, cette ligne permet à l'utilisateur d'introduire un code action qui peut être soit un code standard, soit un code défini par l'utilisateur. Les codes standard sont les suivants :

  <n demande de saut en arrière de n grilles
  >n demande de saut en avant de n grilles
  =n demande de visualisation de la énième grille
  / retour au niveau commande
  ; exécution de l'instruction ACCEPT avec les valeurs déjà introduites.

Un code défini par l'utilisateur ne doit pas être un mot réservé ou un mot-clé connu du système. Il peut être lu dans une requête par l'intermédiaire de la variable système @ACTION. L'utilisation d'un tel code met fin à l'instruction ACCEPT même s'il reste des grilles. Cependant, si la variable $FUNCTION-KEY a été positionnée à 0 avant la soumission de l'instruction, les codes action ne sont pas pris en compte et toutes les grilles sont visualisées.

- La dernière ligne de la grille est fournie par IQS ; elle n'a donc pas à être définie dans la grille utilisateur. Elle permet d'envoyer un message par l'intermédiaire de la variable système @MESSAGE.

INTRODUCTION EN MODE LIGNE :

Lorsque le premier format de l'instruction est utilisé et que les zones spécifiées sont des scalaires, l'introduction de données se déroule comme suit :

- Le nom de la zone (sans préfixe @, # ou $ dans le cas de variables temporaires, de variables système ou de paramètres) ou la valeur de l'expression alphanumérique (lorsque l'option PROMPT est utilisée) est visualisé au terminal.
- Il est suivi de deux points (:) après lesquels doivent être introduites les données ; les données introduites sont transmises au système en appuyant sur la touche TRANSMIT.
- Le processus se répète pour toutes les zones spécifiées, chacune étant traitée dans l'ordre où elle apparaît dans l'instruction.

Les codes action standard sont les suivants :

< retour au début de l'instruction ACCEPT (c'est-à-dire à la première valeur à introduire) ; dans ce cas, les valeurs antérieurement introduites sont visualisées entre parenthèses.
> (ou fin-de-ligne) conservation de l'ancienne valeur
/ retour au niveau commande sans exécution des modifications demandées
; exécution de l'instruction avec les valeurs déjà introduites

A noter que si la variable système $FUNCTION-KEY est positionnée à 0 avant la soumission de l'instruction ACCEPT, les codes ci-dessus ne sont pas pris en compte.

Pour quitter IQS immédiatement et retourner au niveau S:, introduire sous IOF la chaîne EOS (voir exemple plus loin).

Avec le deuxième ou le troisième format de l'instruction ou avec le premier format lorsque la zone spécifiée est de type structure, l'introduction de données se déroule comme suit :

Le processus est le même que précédemment, mais le nom de l'article ou de la zone de type structure est d'abord visualisé seul sur une ligne ; lorsque toutes les zones de l'article ou tous les éléments de la zone ont été traités, le nom de l'article suivant ou de la zone suivante est visualisé et l'introduction de données se poursuit de la même manière. Le changement de niveau de données est indiqué par un décalage de cinq positions vers la droite.

Dans le cas d'une zone de type vecteur ou groupe, le numéro d'occurrence est visualisé après le nom de la zone et une itération automatique est effectuée pour chaque occurrence.

Lorsque la touche TRANSMIT est utilisée juste après l'apparition des deux points ou lorsque la valeur introduite n'est composée que d'espaces, la zone conserve sa valeur précédente et le nom de la zone suivante est visualisé.

Lorsque la zone est de type alphanumérique, l'introduction de la valeur ... (trois points) la rend non définie, c'est-à-dire entièrement à espaces. Si la valeur introduite est trop longue, elle est tronquée à droite.

Lorsque la zone est de type numérique, la valeur introduite est refusée si elle n'est pas conforme à sa définition. Si la valeur est trop longue, elle est tronquée à droite uniquement lorsque la troncation n'affecte que la partie fractionnaire du nombre ; sinon, elle est refusée.

Lorsque des valeurs sont introduites au moyen de l'instruction ACCEPT, les clauses CHECK spécifiées pour les zones à la définition du schéma ne sont pas appliquées. Elles ne le sont que lorsque l'instruction INSERT ou MODIFY est exécutée.

Les zones (autres que les variables temporaires) et les articles spécifiés dans ACCEPT doivent être définis lorsque l'instruction est exécutée. Par conséquent, les zones et les articles d'une base de données ne peuvent être cités dans une instruction ACCEPT que si celle-ci figure dans un bloc CREATE ou RETRIEVE.

Description des paramètres :

nom-zone	Nom de la zone pour laquelle une valeur doit être introduite. Il est possible de spécifier plusieurs noms de zones. (cf. format 1)
WITH CHECKPOINT	Réservé à TDS. Demande de consolidation de la ou des bases de données et de constitution d'un point de reprise pour la requête.
PROMPT <expression-alphanumérique>	Si l'option PROMPT est déclarée, le guidage spécifié dans <expression-alphanumérique> est visualisé pour demander l'introduction d'une valeur pour une zone. Si cette option est omise, c'est le nom de la zone qui est utilisé comme guidage. Pour les expressions alphanumériques, se reporter au chapitre 2.
nom-grille	Nom de la grille utilisateur à employer pour l'introduction de données. L'utilisateur peut spécifier 8 grilles au maximum. Si des valeurs implicites ont été définies pour la grille utilisée, celles-ci sont visualisées et transmises telles quelles, sauf si elles sont modifiées. La recherche des grilles en bibliothèque s'effectue dans l'ordre suivant : #BLIB, #BINLIB1, #BINLIB2 et #BINLIB3.
nom-article	Nom d'un article déjà défini (dans un bloc RETRIEVE ou CREATE). Des données doivent être demandées pour chacune des zones de cet article. Lorsque l'instruction ACCEPT est utilisée sans aucun paramètre, des valeurs sont demandées pour les zones de tous les articles spécifiés dans la dernière instruction RETRIEVE ou CREATE. (cf. format 2)
nom-fichier	Nom du fichier actuellement lu par une instruction READ. Des données sont demandées pour chacune des zones de l'article du fichier. Lorsqu'aucun nom de fichier n'est spécifié, des valeurs sont demandées pour toutes les zones du premier article décrit par la première instruction WRITE qui a défini le fichier. (cf. format 3)

EXEMPLE 1 (ZONES TEMPORAIRES) :

```
ACCEPT @NOM,$NO-TEL
```

Cette instruction demande d'introduire des valeurs pour les zones @NOM et $NO-TEL. La valeur de la première zone est demandée par le guidage :

```
NOM :
```

et celle de la seconde par :

```
NO-TEL :
```

❑

Les valeurs doivent être introduites après les deux points, sur la même ligne.

EXEMPLE 2 (ZONES DE DONNEES) :

```
ACCEPT UNIVERSITE
```

Cette instruction demande d'introduire des valeurs pour toutes les zones de l'article UNIVERSITE. Les guidages sont les suivants :

```
UNIVERSITE
     TYP :
     U-NOM :
     VILLE :
```

❑

EXEMPLE 3 (ACCEPT DANS UNE REQUETE) :

```
ACCEPT @UNIVERSITE, @GRADE
SPACE
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
   WHERE U-NOM=@UNIVERSITE AND GRADE=@GRADE
   PRINT E-NOM, GRADE
   ACCEPT GRADE
   MODIFY GRADE
END
```

❑

Cette requête donne lieu au dialogue suivant (les valeurs introduites par l'utilisateur sont soulignées) :

```
UNIVERSITE : PARIS-1
GRADE : LICENCE

ARNAUD              LICENCE
GRADE : MAITRISE
MARTIN              LICENCE
GRADE : LICENCE
SARRE               LICENCE
GRADE : MAITRISE
```

EXEMPLE 4 (MODIFICATION DU GUIDAGE) :

Si la première ligne de l'exemple précédent est remplacée par :

```
ACCEPT @UNIVERSITE, @GRADE PROMPT "GRADE DE L'ETUDIANT"
❑
```

Les guidages sont les suivants :

```
UNIVERSITE : PARIS-1
GRADE DE L'ETUDIANT : LICENCE
```

Les valeurs introduites par l'utilisateur sont soulignées.

EXEMPLE 5 (UTILISATION DE LA CHAINE EOS POUR QUITTER IQS) :

L'introduction de la chaîne EOS à la suite d'un guidage affiché par ACCEPT provoque la sortie d'IQS et le retour au niveau S:.

```
V: EXEC
   VILLE : /
V: EXEC
   VILLE : EOS
<<<16:51
S:
❑
```

5.3 ALTER

Fonction :

Demande d'introduction de données au terminal.
En traitement par lots, les données sont prises dans le fichier COMFILE.

Format 1 :

```
|-----------------------------------------------------------|
|    ALTER  [WITH CHECKPOINT] <nom-zone>                     |
|                                                           |
|       [PROMPT <expression-alphanumérique>]                 |
|                                                           |
|   [,<nom-zone> [PROMPT <expression-alphanumérique>]]...    |
|                                                           |
|      [THRU <nom-grille> [,<nom-grille>]...]                |
|-----------------------------------------------------------|
```

Format 2 :

```
|-------------------------------------------------|
|  ALTER  [WITH CHECKPOINT]                       |
|                                                 |
|     [<nom-article>[,<nom-article>]...]          |
|                                                 |
|     [THRU <nom-grille> [,<nom-grille>]...]      |
|-------------------------------------------------|
```

Format 3 :

```
|-------------------------------------------------|
|    ALTER  [WITH CHECKPOINT]                     |
|                                                 |
|       [<nom-fichier>]                           |
|                                                 |
|       [THRU <nom-grille> [,<nom-grille>]...]    |
|-------------------------------------------------|
```

Description de l'instruction :

L'instruction ALTER permet de transférer des données introduites au terminal dans des articles et/ou des zones. L'ancienne valeur des zones, si elle est différente d'espaces, est visualisée entre parenthèses lorsque la nouvelle valeur est demandée.

Le contenu des articles ou des zones est remplacé par les données introduites. L'instruction ALTER ne modifie que les données se trouvant en mémoire. Les données se trouvant dans un fichier ou dans une base de données ne peuvent être modifiées que par les instructions WRITE, INSERT ou MODIFY.

L'option WITH CHECKPOINT est réservée à TDS.

Les données sont introduites soit au moyen de grilles utilisateur (sur les terminaux fonctionnant en mode grille), soit en mode ligne.

INTRODUCTION EN MODE GRILLE :

- Le nombre maximum de grilles par instruction ALTER est de 8.
- Il ne doit y avoir qu'une seule grille par écran, autrement dit, il n'est pas possible de visualiser plusieurs grilles à la fois.
- Le nombre de lignes d'une grille utilisateur ne doit pas excéder le nombre de lignes de l'écran moins 2 (généralement 22).
- La première ligne de la grille (qui contient -->__) est fournie par IQS ; elle n'a donc pas à être définie dans la grille utilisateur. Lorsque la grille est visualisée, cette ligne permet à l'utilisateur d'introduire un code action qui peut être soit un code standard, soit un code défini par l'utilisateur. Les codes standard sont les suivants :

  <n demande de saut en arrière de n grilles
  >n demande de saut en avant de n grilles
  =n demande de visualisation de la énième grille
  / retour au niveau commande
  , exécution de l'instruction ALTER avec les valeurs déjà introduites.

  Un code défini par l'utilisateur ne doit pas être un mot réservé ou un mot-clé connu du système. Il peut être lu dans une requête par l'intermédiaire de la variable système @ACTION. L'utilisation d'un tel code met fin à l'instruction ALTER même s'il reste des grilles. Cependant, si la variable $FUNCTION-KEY a été positionnée à 0 avant la soumission de l'instruction, les codes action ne sont pas pris en compte et toutes les grilles sont visualisées.
- La dernière ligne de la grille est fournie par IQS ; elle n'a donc pas à être définie dans la grille utilisateur. Elle permet d'envoyer un message par l'intermédiaire de la variable système @MESSAGE.

INTRODUCTION EN MODE LIGNE :

Lorsque le premier format de l'instruction est utilisé et que les zones spécifiées sont des scalaires, l'introduction de données se déroule comme suit :

- Le nom de la zone (sans préfixe @, # ou $ dans le cas des noms de variables temporaires, de variables système ou de paramètres) ou la valeur de l'expression alphanumérique (lorsque l'option PROMPT est utilisée) est visualisé au terminal.
- Si l'ancienne valeur de la zone est différente d'espaces, elle est visualisée entre parenthèses sur la même ligne.
- Deux points (:) sont ensuite visualisés sur la même ligne ; les données doivent être introduites après ces deux points et transmises au système en appuyant sur la touche TRANSMIT.
- Le processus se répète pour toutes les zones spécifiées, chacune étant traitée dans l'ordre où elle apparaît dans l'instruction.

Les codes action standard sont les suivants :

<	retour au début de l'instruction ALTER (c'est-à-dire à la première valeur à introduire)
>	(ou fin-de-ligne) conservation de l'ancienne valeur
/	retour au niveau commande sans exécution des modifications demandées
,	exécution de l'instruction avec les valeurs déjà introduites

A noter que si la variable système $FUNCTION-KEY est positionnée à 0 avant la soumission de l'instruction ALTER, les codes ci-dessus ne sont pas pris en compte.

Pour quitter IQS immédiatement et retourner au niveau S:, introduire sous IOF la chaîne EOS (voir exemple plus loin).

La valeur d'une zone peut également être modifiée au moyen de la demande de l'éditeur de texte S/ancienne-valeur/nouvelle-valeur/. Pour plus de détails sur cette demande, se reporter au chapitre 4.

Avec le deuxième ou le troisième format de l'instruction ou avec le premier format lorsque la zone spécifiée est de type structure, l'introduction de données se déroule comme suit :

Le processus est le même que précédemment, mais le nom de l'article ou de la zone de type structure est d'abord visualisé seul sur une ligne ; lorsque toutes les zones de l'article ou tous les éléments de la zone ont été visualisés, le nom de l'article suivant ou de la zone suivante est visualisé et l'introduction de données se poursuit de la même manière. Le changement de niveau de données est indiqué par un décalage de cinq positions vers la droite.

Dans le cas d'une zone de type vecteur ou groupe, le numéro d'occurrence est visualisé après le nom de la zone et une itération automatique est effectuée pour chaque occurrence.

Lorsque la touche TRANSMIT est utilisée juste après l'apparition des deux points ou lorsque la valeur introduite n'est composée que d'espaces, la zone conserve sa valeur précédente et le nom de la zone suivante est visualisé.

Lorsque la zone est de type alphanumérique, l'introduction de la valeur ... (trois points) la rend non définie, c'est-à-dire entièrement composée d'espaces. Si la valeur introduite est trop longue, elle est tronquée à droite.

Lorsque la zone est de type numérique, la valeur introduite est refusée si elle n'est pas conforme à sa définition. Si la valeur est trop longue, elle est tronquée à droite uniquement lorsque la troncation n'affecte que la partie fractionnaire du nombre ; sinon, elle est refusée.

Lorsque des valeurs sont introduites au moyen de l'instruction ALTER, les clauses CHECK spécifiées pour les zones à la définition du schéma ne sont pas appliquées. Elles ne le sont que lorsque l'instruction INSERT ou MODIFY est exécutée.

Les zones (autres que les variables temporaires) et les articles spécifiés dans ALTER doivent être définis lorsque l'instruction est exécutée. Par conséquent, les zones et les articles d'une base de données ne peuvent être cités dans une instruction ALTER que si celle-ci figure dans un bloc CREATE ou RETRIEVE.

Description des paramètres :

nom-zone	Nom de la zone pour laquelle une valeur doit être introduite. Il est possible de spécifier plusieurs noms de zones (cf. format 1). Si une zone numérique contient une valeur non autorisée (donnée numérique incorrecte), la valeur visualisée est ???.
WITH CHECKPOINT	Réservé à TDS. Demande de consolidation de la ou des bases de données et de constitution d'un point de reprise pour la requête.
PROMPT <expression-alphanumérique>	Si l'option PROMPT est déclarée, le guidage spécifié dans <expression-alphanumérique> est visualisé pour demander l'introduction d'une valeur pour une zone. Si cette option est omise, c'est le nom de la zone qui est utilisé comme guidage. Pour les expressions alphanumériques, se reporter au chapitre 2.
nom-grille	Nom de la grille utilisateur à employer pour l'introduction de données. L'utilisateur peut spécifier 8 grilles au maximum. La recherche des grilles en bibliothèque s'effectue dans l'ordre suivant : #BLIB, #BINLIB1, #BINLIB2 et #BINLIB3. La grille est remplie avec les valeurs des zones avant d'être visualisée.
nom-article	Nom d'un article déjà défini (dans un bloc CREATE ou RETRIEVE). Des données doivent être demandées pour chacune des zones de cet article. Lorsque l'instruction ALTER est utilisée sans aucun paramètre, des valeurs sont demandées pour les zones de tous les articles spécifiés dans la dernière instruction CREATE ou RETRIEVE. (cf. format 2)
nom-fichier	Nom d'un fichier lu par une instruction READ. Toutes les zones de données de l'article sont acceptées. Si aucun nom n'est spécifié, les valeurs implicites sont celles des zones de données du premier article décrit par la première instruction WRITE définissant le fichier de l'instruction READ précédente. (cf. format 3)

EXEMPLE 1 :

La requête suivante permet d'introduire au terminal des valeurs pour UNIVERSITE et GRADE, pour rechercher les articles ETUDIANT correspondant aux valeurs fournies. La valeur de la zone E-NOM de ces articles est alors visualisée précédée de GRADE DE et la valeur courante de la zone GRADE est visualisée sur la même ligne pour être éventuellement modifiée. A noter l'utilisation de l'expression SUBSTRING pour rendre les guidages plus explicites.

```
ACCEPT @UNIVERSITE
       @GRADE
SPACE
RETRIEVE UNIVERSITE,
         COLLEGE,
         ETUDIANT
         WHERE U-NOM = @UNIVERSITE
               AND GRADE = @GRADE
   LET @A = "GRADE DE"
   LET SUBSTRING (@A,10,20) = E-NOM
   ALTER GRADE PROMPT @A
   MODIFY ETUDIANT
END
❑
```

La requête est exécutée et les grades de ARNAUD et SARRE sont modifiés :

```
UNIVERSITE : PARIS-1
GRADE : LICENCE

GRADE DE ARNAUD (LICENCE) : MAITRISE
GRADE DE MARTIN (LICENCE) :
GRADE DE SARRE (LICENCE) : MAITRISE
```

Du fait que l'instruction MODIFY a été utilisée, ces modifications sont enregistrées dans la base de données. Pour la remettre à l'état antérieur, il faudrait introduire les valeurs suivantes :

```
UNIVERSITE : PARIS-1
GRADE : MAITRISE

GRADE DE ARNAUD (MAITRISE) : LICENCE
GRADE DE SERREAU (MAITRISE) :
GRADE DE WAGNER (MAITRISE) :
GRADE DE MARTY (MAITRISE) :
GRADE DE SARRE (MAITRISE) : LICENCE
GRADE DE WIART (MAITRISE) : /
```

EXEMPLE 2 :

La requête suivante permet de sélectionner un article ETUDIANT et d'en modifier certaines ou toutes les zones :

```
ACCEPT @NOM-ETUDIANT
RETRIEVE ETUDIANT
         WHERE E-NOM = NOM-ETUDIANT
   ALTER
   MODIFY ETUDIANT
END
❑
```

La requête est exécutée et l'utilisateur apporte quelques modifications mais, arrivé au guidage DIPLOME 2, il s'aperçoit qu'il a oublié de modifier le numéro de sécurité sociale. Il lui suffit d'introduire < pour revenir au premier guidage sans que les modifications qu'il a déjà apportées soient perdues :

```
NOM-ETUDIANT : ARNAUD
ETUDIANT
     TYP (E) :
     E-U-NOM (PARIS-1) :
     E-C-NOM (MEDECINE) :
     E-NOM (ARNAUD) :
     SEC-SOCIALE (136 45 7341) :
     GRADE (LICENCE) : MAITRISE
     DATE-NAISSANCE
         N-ANNEE (61) :
         N-MOIS (10) :
         N-JOUR (9) :
     FRAIS-SCOL (2000) : 1500
     DIPLOME 1 : FRANCAIS
     DIPLOME 2 : <
ETUDIANT
     TYP (E) :
     E-U-NOM (PARIS-1) :
     E-C-NOM (MEDECINE) :
     E-NOM (ARNAUD) :
     SEC-SOCIALE (136 45 7341) : S/45/46/
     SEC-SOCIALE (136 46 7341) :
     GRADE (MAITRISE) : S/MAITRISE/LICENCE/
     GRADE (LICENCE) :
     DATE-NAISSANCE
         N-ANNEE (61) :
         N-MOIS (10) :
         N-JOUR (9) :
     FRAIS-SCOL (1500) :
     DIPLOME 1 (FRANCAIS) : ...
     DIPLOME 2 :
     DIPLOME 3 : ;
```

Au cours de la seconde visualisation des guidages, l'utilisateur corrige le numéro de sécurité sociale et le grade au moyen de la demande S/ / / de l'éditeur de texte ; la modification apportée est visualisée immédiatement. Il annule également la valeur de DIPLOME 1 en introduisant trois points (...).

5.4 ASSIGN

Fonction :

Association d'un fichier de travail à un fichier séquentiel (UFAS ou sous-fichier bibliothèque), ou d'un fichier d'impression destiné à recevoir les sorties générées par une instruction PRINT ou SPACE, à un fichier permanent ou à SYSOUT (pour sortie sur l'imprimante ligne).

Format 1 :

```
.-----------------------------------------.
|    ASSIGN PRT <nom-fichier-impression>  |
|                                         |
|       { IFN <nom-fichier-interne>     } |
|    TO {                               } |
|       { <expression-alphanumérique-1> } |
 -----------------------------------------
```

Format 2 :

```
.-----------------------------------------------.
|          { <*expression-alphanumérique-2> }   |
|   ASSIGN {                                }   |
|          {  <nom-fichier-travail>         }   |
|                                               |
|      { IFN <nom-fichier-interne>     }        |
|  [TO {                               } ]      |
|      { <expression-alphanumérique-3> }        |
 -----------------------------------------------
```

Description de l'instruction :

L'instruction ASSIGN permet d'affecter un fichier de travail dont le nom peut être prédéfini ou spécifié dans l'instruction. Le fichier peut être temporaire ou associé à un nom de fichier externe connu.

A noter qu'une fois affecté, le fichier reste connu d'IQS pendant toute la session et peut être réutilisé par d'autres requêtes ou commandes.

L'instruction ASSIGN PRT (format 1) sert à envoyer les sorties dans un fichier permanent ou directement sur l'imprimante ligne via le fichier SYSOUT.

Pour l'élaboration de requêtes exécutables sous TDS et comportant des instructions ASSIGN, se reporter au guide utilisateur IQS-V4/TDS (81UR).

Description des paramètres :

nom-fichier-impression	Nom logique du fichier d'impression.
expression-alphanumérique-1	Nom externe du fichier d'impression.
nom-fichier-interne	Identification du fichier par son nom de fichier interne (ifn) spécifié lors de son affectation au moyen du GCL ou du JCL. (Sous IQS/IOF, l'affectation a été effectuée lors du lancement d'IQS ; sous IQS/TDS, lors du lancement de TDS).
*expression-alphanumérique-2	Nom désignant un fichier de travail existant. Lorsque cette option est utilisée, le nom du fichier de travail est le résultat de l'expression considérée.
nom-fichier-travail	Spécification directe d'un fichier de travail qui sera utilisé dans des instructions WRITE, SORT ou READ ultérieures.
expression-alphanumérique-3	Expression indiquant le nom de fichier externe du fichier de travail. Lorsque cette option est omise, un fichier temporaire est associé au fichier de travail.

5.5 CANCEL FORMAT

Fonction :

Suppression du lien existant entre un fichier de travail et un format de présentation.

Format :

```
|--------------------------------------|
|    CANCEL FORMAT ON <nom-fichier>    |
|--------------------------------------|
```

Description de l'instruction :

L'instruction CANCEL FORMAT permet de dissocier du fichier de travail spécifié le format de présentation qui lui était éventuellement lié.

Cette instruction peut être déclarée n'importe où à l'intérieur d'une requête. En outre, elle peut être spécifiée plusieurs fois dans une même requête.

Description du paramètre :

nom-fichier	Nom du fichier de travail auquel était associé un format de présentation.

5.6 CANCEL REPORT

Fonction :

Suppression du lien existant entre un fichier d'impression et une description d'état.

Format :

```
 ------------------------------------------------------
|                     { PRINTER                  }     |
| CANCEL REPORT  [ ON {                          } ]   |
|                     { <nom-fichier-impression> }     |
 ------------------------------------------------------
```

Description de l'instruction :

L'instruction CANCEL REPORT permet de dissocier du fichier d'impression spécifié la description d'état qui lui était liée.

CANCEL REPORT clôt la description d'état en cours en imprimant les titres de bas de page (FOOTING) avant d'exécuter l'instruction.

Cette instruction peut être déclarée n'importe où à l'intérieur d'une requête. En outre, elle peut être spécifiée plusieurs fois au sein d'une même requête.

Description du paramètre :

nom-fichier-impression	Nom d'un fichier séquentiel spécifié comme fichier d'impression dans une instruction PRINT. La valeur implicite est PRINTER.

5.7 CANCEL TITLE

Fonction :

Annulation de l'effet d'une instruction HEADING et/ou FOOTING précédente pour un fichier d'impression.

Format :

```
 ------------------------------------------------------
|                      { PRINTER                   }    |
| CANCEL TITLE   [ ON  {                           } ]  |
|                      { <nom-fichier-impression>  }    |
 ------------------------------------------------------
```

Description de l'instruction :

L'instruction CANCEL TITLE clôt d'abord le tableau en cours, puis annule la définition des titres haut et/ou bas de colonnes spécifiée par une précédente instruction HEADING et/ou FOOTING ou par une instruction PRINT WITH TITLE.

Description du paramètre :

nom-fichier-impression	Nom du fichier d'impression pour lequel les définitions de titres doivent être annulées. La valeur implicite est PRINTER.

EXEMPLE :

La requête suivante imprime un tableau dont les colonnes portent les titres NOM et FRAIS-SCOL (en haut), puis annule la définition de ces titres et imprime une nouvelle ligne TOTAL FRAIS SCOLARITE =.

```
LET @COLUMN-BORDER = "!"
LET @LINE-BORDER = "-"
LET $COLUMN-SPACING = 3
HEADING PR1
RETRIEVE UNIVERSITE,
         COLLEGE,
         ETUDIANT
         WHERE U-NOM = "PARIS-1"
  LET $SUM = SUM(FRAIS-SCOL)
PR1.
  PRINT E-NOM TITLE "NOM",
        FRAIS-SCOL
END
CANCEL TITLE
SPACE
PRINT "TOTAL FRAIS-SCOLARITE =",
      $SUM JUSTIFIED LEFT COLUMN + 1
SPACE TOP
```

❑

Cette requête produit le tableau suivant :

```
|-----------------------|--------------|
| NOM                   |   FRAIS-SCOL |
|-----------------------|--------------|
| ARNAUD                |         2000 |
| SERREAU               |         1900 |
| WAGNER                |         1900 |
| MARTIN                |          800 |
| MARTY                 |         1200 |
| SARRE                 |         1100 |
| WIART                 |         1500 |
|-----------------------|--------------|

TOTAL FRAIS-SCOLARITE = 10400
```

5.8 CHECKPOINT

Fonction :

Exécution d'une consolidation (commitment) d'une ou plusieurs bases de données et constitution d'un point de reprise pour la requête.
Utilisable uniquement sous TDS et en traitement par lots.

Format :

```
|-----------------|
|   CHECKPOINT    |
|-----------------|
```

Description de l'instruction :

L'instruction CHECKPOINT permet d'exécuter une consolidation d'une ou plusieurs bases de données et de constituer un point de reprise pour la requête.

Lorsque cette instruction est utilisée sous IOF, la variable système @STATUS est forcée à la valeur "FUNCNAV". Dans ce cas, la requête s'arrête prématurément, à moins que l'utilisateur n'ait prévu un test sur la valeur de @STATUS.

Pour plus de détails sur le concept de point de reprise, se reporter au guide utilisateur IQS-V4/TDS (81UR).

5.9 COMMIT

Fonction :

Exécution d'une consolidation (angl. commitment).
Utilisable uniquement sous IOF ou en traitement par lots.

Format :

```
|-----------|
|  COMMIT   |
|-----------|
```

Description de l'instruction :

L'instruction COMMIT permet d'exécuter une consolidation. Les mises à jour effectuées depuis la dernière instruction COMMIT sont définitivement enregistrées dans la base de données.

Cette instruction peut être utilisée pour les fichiers UFAS et les bases de données IDS/II gérés par GAC (General Access Control).

Elle est interdite à l'intérieur des blocs RETRIEVE, READ et CREATE.

Si l'utilisateur désire ne pas enregistrer les mises à jour effectuées depuis la dernière instruction COMMIT, il doit utiliser l'instruction ROLLBACK décrite plus loin.

5.10 CONNECT

Fonction :

Rattachement d'une occurrence d'article détail IDS/II à une occurrence de son article maître.

Format :

```
|-----------------------------------------------------|
|CONNECT <nom-article-réel-1> TO <nom-article-réel-2> |
|                                                     |
|        [VIA <nom-ensemble>]                         |
|-----------------------------------------------------|
```

Description de l'instruction :

L'instruction CONNECT permet de relier un article à l'un de ses articles maîtres par l'intermédiaire d'un ensemble. Dans cet ensemble, nom-article-réel-1 doit être défini comme article détail (MEMBER) et nom-article-réel-2 comme article maître (OWNER) avec la clause INSERTION MANUAL.

CONNECT ne peut être utilisée avec une vue.

Elle doit être exécutée dans le contexte d'une instruction RETRIEVE ou CREATE (après INSERT) et ne s'applique qu'aux articles IDS/II.

Pour plus de détails sur les règles d'insertion, se reporter à l'instruction INSERT.

Description des paramètres :

nom-article-réel-1	Nom ou synonyme de l'article dont l'occurrence considérée doit être rattachée à celle de l'article maître. nom-article-réel-1 doit avoir été défini comme article détail (MEMBER) dans l'ensemble considéré et avoir été spécifié dans une instruction RETRIEVE ou INSERT précédente. L'aire dont il fait partie doit avoir été ouverte en mode mise à jour (UPDATE).
nom-article-réel-2	Nom ou synonyme de l'article dont l'occurrence considérée doit devenir maîtresse de celle de nom-article-réel-1. Il doit avoir été spécifié comme article maître (OWNER) de l'ensemble considéré et avoir été spécifié dans une instruction RETRIEVE précédente. L'aire dont il fait partie doit avoir été ouverte en mode mise à jour (UPDATE).
nom-ensemble	Nom de l'ensemble reliant nom-article-réel-1 et nom-article-réel-2. La clause VIA est obligatoire lorsque les deux articles sont reliés par plusieurs ensembles.

EXEMPLE 1 :

Cas d'un article obtenu par RETRIEVE :

```
RETRIEVE UNIVERSITE,
         COLLEGE,
         ENSEIGNANT
         WHERE U-NOM = "PARIS-1"
               AND C-NOM = "SCIENCE"
               AND PROFESSEUR = "JULY"
  RETRIEVE DEPARTEMENT VIA C-D
           WHERE NOM-DEPART = "MATHEMATIQUE"
    CONNECT ENSEIGNANT TO DEPARTEMENT
  END
END
```

❑

Résultat :

```
RETRIEVE ENSEIGNANT,
         DEPARTEMENT
         WHERE PROFESSEUR = "JULY"
  DISPLAY NOM-DEPART
END
NOM-DEPART  : MATHEMATIQUE
```

EXEMPLE 2 :

Cas d'un article obtenu par INSERT :

```
RETRIEVE UNIVERSITE,
         COLLEGE,
         WHERE U-NOM = "PARIS-1"
               AND C-NOM = "MEDECINE"
  CREATE ENSEIGNANT
    ACCEPT
    INSERT ENSEIGNANT
    RETRIEVE DEPARTEMENT
             WHERE NOM-DEPART = "PATHOLOGIE"
      CONNECT ENSEIGNANT TO DEPARTEMENT
    END
  END
END

ENSEIGNANT
  PROFESSEUR : JOURDAIN
  DEGRE : 25
  SEC-SOC : 250 13 2538
❑
```

Demande de visualisation du résultat de la requête ci-dessus :

```
RETRIEVE DEPARTEMENT,
         ENSEIGNANT
         WHERE NOM-DEPART = "PATHOLOGIE"
  DISPLAY
END
```

Résultat obtenu :

```
DEPARTEMENT
  NOM-DEPART : PATHOLOGIE
ENSEIGNANT
  PROFESSEUR : JOURDAIN
  DEGRE : 25
  SEC-SOC : 250 13 2538
```

5.11 CREATE

Fonction :

Initialisation du processus de création d'un article.

Format :

```
|-----------------------------------------------------|
| [<étiquette>.] CREATE [<synonyme>=] <nom-article>   |
|                                                     |
|                     <séquence-d'instructions> ...   |
|                                                     |
|                END [<étiquette>]                    |
|-----------------------------------------------------|
```

Description de l'instruction :

L'instruction CREATE introduit une séquence d'instructions destinée à créer un nouvel article en mémoire. Chaque fois que l'instruction CREATE est détectée, les zones de l'article sont remises à espaces si elles sont alphanumériques et à zéro si elles sont numériques.
CREATE ne permet pas d'enregistrer l'article dans la base de données ; pour ce faire, il faut utiliser l'instruction INSERT.

Un bloc CREATE...END peut contenir plusieurs instructions INSERT.

Pour plus de détails sur les règles d'insertion, se reporter à l'instruction INSERT.

Description des paramètres :

étiquette	Etiquette facultative fournie par l'utilisateur pour identifier l'instruction CREATE et devant figurer dans l'instruction END correspondante. Cette étiquette peut également être spécifiée dans une instruction EXIT ou REPEAT.
synonyme	Nom fourni par l'utilisateur pour remplacer le nom de l'article dans la séquence d'instructions.
nom-article	Nom de l'article à créer.
séquence-d'instructions	Séquence d'instructions à exécuter. A noter que l'article spécifié par CREATE n'est pas pris en compte dans la détermination du chemin d'accès à utiliser dans les instructions RETRIEVE éventuellement présentes dans la séquence.

EXEMPLE :

Création d'un ou plusieurs articles COLLEGE, dont le nom est introduit au terminal, pour chaque article UNIVERSITE du fichier UFAS séquentiel indexé. Les blocs RETRIEVE et CREATE sont identifiés par des points reliant l'instruction de début de la boucle et celle de fin (END) :

```
RETRIEVE UNIVERSITE WHERE U-NOM BEGINS "T"
.  DISPLAY
.  CREATE COLLEGE
.  .  ACCEPT
.  .  IF C-NOM ABSENT
.  .  THEN
.  .    EXIT
.  .  END
.  .  INSERT COLLEGE
.  .  REPEAT
.  END
END
```

❑

Le résultat obtenu est le suivant :

```
UNIVERSITE
    TYP : U
    U-NOM : TOULON-3
    FIL2 : ...
    FIL3 : ...
    FIL4 : ...
    VILLE : ...
COLLEGE
  C-NOM : SCIENCE
COLLEGE
  C-NOM : LETTRES
COLLEGE
  C-NOM : MEDECINE
COLLEGE
  C-NOM : ;
UNIVERSITE
    TYP : U
    U-NOM : TOULOUSE-2
    FIL2 : ...
    FIL3 : ...
    FIL4 : ...
    VILLE : TOULOUSE
COLLEGE
  C-NOM : SCIENCE
COLLEGE
  C-NOM : /
```

5.12 DEFINE (DEF)

Fonction :

Définition de variables permanentes ou temporaires ou redéfinition de variables temporaires.

Format :

```
.-------------------------------------------------------------.
|                           {<variable-permanente>          } |
| DEF[INE] [<numéro-niveau>] {                              } |
|                           {<variable-temporaire> [(<occ>)] } |
|                                                             |
|   ||REDEF[INES] <zone> ||                                   |
|   ||                   ||                                   |
|   ||<attribut>         ||                                   |
|                                                             |
|                      {<variable-permanente>          }      |
| [,[<numéro-niveau>] {                              }       |
|                      {<variable-temporaire> [(<occ>)] }     |
|                                                             |
|       ||REDEF[INES] <zone> ||                              |
|       ||                   ||   ]...                       |
|       ||<attribut>         ||                              |
 -------------------------------------------------------------
```

```
.-------------------------------------------------------------.
| Le format de <attribut> est le suivant :                    |
|                                                             |
| 1) IDEM <nom-zone>                                          |
|                                                             |
| 2) Lorsque la zone est alphanumérique :                     |
|                                                             |
|                {(<longueur>) }                              |
|    CHAR[ACTER] {             }                              |
|                { <longueur>  }                              |
|                                                             |
| 3) Lorsque la zone est numérique :                          |
|                                                             |
|{           {(<longueur>)}                                  }|
|{  BIN[ARY] { <longueur> }                                  }|
|{                                                           }|
|{ {UNPACKED}  {SIGNED  }           {(<longueur>[,<décimales>])}}|
|{[{PACKED  }][{UNSIGNED}] DEC[IMAL]{<longueur>[,<décimales>]  }}|
 -------------------------------------------------------------
```

Description de l'instruction :

Cette instruction non exécutable permet de définir des variables permanentes ou temporaires ou de redéfinir des variables temporaires. Le nom de la variable peut être précédé du préfixe @ (zone alphanumérique), $ ou # (zones numériques). Plusieurs variables peuvent être définies dans la même instruction DEFINE ; dans ce cas, leurs définitions doivent obligatoirement être séparées par des virgules.

Des variables permanentes ne peuvent être définies que sous forme de scalaires.

Des variables temporaires peuvent être définies ou redéfinies sous les formes suivantes :

scalaire	Un scalaire est une zone élémentaire isolée.
vecteur	Un vecteur est un ensemble de zones élémentaires ayant la même définition (c'est-à-dire un tableau unidimensionnel de zones élémentaires). Chaque élément est identifié de manière unique par le nom du vecteur suivi d'un indice (zone numérique ou constante entre parenthèses).
structure	Une structure est un ensemble hiérarchisé de zones. Le niveau le plus bas de la hiérarchie se compose de scalaires ou de vecteurs ; le niveau le plus haut est représenté par le nom de la structure. Entre ces deux niveaux, des structures intermédiaires peuvent subdiviser progressivement la structure. L'organisation de la structure est spécifiée dans une instruction DEFINE au moyen de numéros de niveau. Plus le niveau hiérarchique est bas, plus le numéro est élevé ; des niveaux successifs ne portent pas obligatoirement des numéros consécutifs. La fin de la définition d'une structure n'est pas indiquée explicitement. Elle est détectée à l'apparition d'une instruction, d'une définition de structure portant un numéro de niveau inférieur ou égal au sien ou d'une définition de scalaire ou de vecteur sans numéro de niveau. Un nom de structure peut être précédé du préfixe @. Une structure ne peut pas avoir d'attributs (IDEM, CHAR, BIN ou DEC).
groupe	Un groupe est un ensemble de structures identiques (c'est-à-dire un tableau unidimensionnel de structures). Chaque élément du groupe est identifié de manière unique par un indice. Le nombre maximum de groupes emboîtés est 3. Un groupe ne doit pas avoir d'attributs (IDEM, CHAR, BIN ou DEC).

Description des paramètres :

numéro-niveau	Constante entière non signée différente de zéro ; les zéros à gauche sont ignorés. Les scalaires et les vecteurs peuvent être définis avec un numéro de niveau égal à 1 ou sans numéro de niveau, auquel cas celui-ci prend la valeur implicite 1. Les noms de structure et de groupe doivent être précédés du numéro de niveau 1 ; les noms des éléments dont ils se composent doivent être précédés d'un numéro de niveau indiquant leur position dans la hiérarchie.
variable-permanente	Nom de la variable permanente à définir. La définition n'est valable que pour la durée de la requête. Il existe dix variables permanentes binaires : $NUM1 à $NUM10, dix variables permanentes alphanumériques : @CHAR1 à @CHAR10 et dix variables permanentes décimales : #DEC1 à #DEC10. Une variable permanente doit être définie avant toute instruction exécutable de la requête dans laquelle elle est utilisée.
variable-temporaire	Une variable temporaire peut être redéfinie au cours de la même requête. Son nom n'a pas à être précédé d'un préfixe $, @ ou #.
occ	Ce paramètre est utilisé pour définir des vecteurs et des groupes. Il s'agit d'une constante entière qui indique le nombre d'occurrences d'une variable temporaire de type vecteur ou groupe.
REDEFINES <nom-zone>	Nom de la variable temporaire à redéfinir. Lorsque l'option REDEFINES est utilisée, variable-temporaire doit avoir un numéro de niveau égal à 1 (ou pas de numéro de niveau) et sa longueur doit être inférieure ou égale à celle de <zone>. <zone> ne doit pas être qualifié ou indicé. Les variables système ne peuvent pas être redéfinies.
IDEM <nom-zone>	Nom de la zone dont la description est à utiliser pour variable-temporaire. Sa description doit déjà exister (dans une structure, un article de la vue courante ou une instruction DEFINE ou PARAMETER précédente). Il peut s'agir du nom d'un article, d'une structure, d'un vecteur, d'un groupe ou d'un scalaire. Lorsque le nom d'un vecteur ou d'un groupe est utilisé, le nombre d'occurrences (occ) peut être modifié. <nom-zone> ne peut pas être qualifié ou indicé.

CHARACTER	Ce paramètre indique que la zone est de type alphanumérique. <longueur> indique le nombre maximum de caractères que peut contenir la zone ; sa valeur maximum est 32767 pour les variables temporaires et 32 pour les variables permanentes.

BINARY	Ce paramètre indique que la zone est de type binaire. Les deux longueurs autorisées sont 15 et 31 bits. Les valeurs possibles sont les suivantes :
	`BINARY 15 -32768 à 32767`
	`BINARY 31 -2**31 à (2**31) -1`

SIGNED/UNSIGNED	Ces deux paramètres s'appliquent aux zones de type DECIMAL uniquement. Ils indiquent si une position doit être prévue pour le signe lorsque la valeur est imprimée. La valeur implicite est SIGNED.
PACKED/UNPACKED	Ces deux paramètres s'appliquent aux zones de type DECIMAL uniquement. PACKED indique que la valeur doit être rangée sous forme condensée (2 chiffres décimaux par octet) et UNPACKED qu'elle doit l'être sous forme éclatée (1 chiffre décimal par octet). La valeur implicite est UNPACKED.
DECIMAL	Ce paramètre indique que la zone est de type décimal. <longueur> spécifie le nombre maximum de chiffres décimaux que peut contenir la zone ; sa valeur ne doit pas dépasser 31 et ne comprend pas le signe.
longueur	Constante numérique indiquant la longueur maximum de la valeur que peut contenir la zone. L'unité dans laquelle la longueur est exprimée dépend du type de la zone (voir CHARACTER, BINARY et DECIMAL ci-dessus).
décimales	Ce paramètre ne s'applique qu'aux zones de type DECIMAL. Il s'agit d'une constante numérique qui indique le nombre de chiffres devant figurer à droite de la marque décimale. Sa valeur implicite est zéro. Les paramètres longueur et décimales doivent être séparés par au moins un espace ou une virgule. Le nombre de chiffres après la marque décimale doit être inclus dans la longueur.

EXEMPLE 1 : INSTRUCTIONS SIMPLES

```
DEFINE $NUM DECIMAL 2
```

$NUM est une zone numérique pouvant contenir 2 chiffres décimaux plus un signe.

```
DEFINE #FACTEUR DECIMAL 6 2
```

#FACTEUR est une zone numérique pouvant contenir 6 chiffres décimaux plus un signe. La marque décimale est placée avant l'avant-dernier chiffre.

```
DEFINE @UNIVERSITE CHARACTER 20
```

@UNIVERSITE peut contenir 20 caractères.

```
DEFINE TOTAL BINARY 31
```

TOTAL peut contenir 31 chiffres binaires plus un signe.

```
DEFINE $BUDGET SIGNED UNPACKED DECIMAL 15
```

$BUDGET peut contenir 15 chiffres décimaux plus un signe.

```
DEFINE @NOM(5) CHARACTER 30
```

@NOM est une zone de type vecteur qui se répète 5 fois. Chaque occurrence de cette zone peut contenir 30 caractères.

❑

EXEMPLE 2 : Utilisation de IDEM

```
DEF ETUD-TEMP IDEM ETUDIANT DISPLAY ETUD-TEMP
```
❑

L'affichage est le suivant :

```
ETUD-TEMP
     TYP : ...
     E-U-NOM : ...
     E-C-NOM : ...
     E-NOM : ...
     SEC-SOCIALE : ...
     GRADE : ...
```

etc., les zones de ETUD-TEMP étant identiques à celles de l'article ETUDIANT du schéma.

EXEMPLE 3 : Utilisation des niveaux

```
DEFINE
       1  AA,
          2 BB CHARACTER 12,
          2 CC DECIMAL 12,2,
          2 DD,
            3 EE BINARY 15,
            3 FF BINARY 31,
          2 GG PACKED SIGNED 10,2
ACCEPT AA
PRINT WITH TITLE AA
❑
```

Des valeurs sont demandées pour chacun des niveaux dont se compose AA :

```
AA
     BB : ABCDEF
     CC : 123.456
     DD
          EE : -1200
          FF :  135000
     GG : -528.48

BB            CC           EE         FF          GG

ABCDEF    123.45        -1200     135000     -528.48
```

EXEMPLE 4 : Utilisation de REDEFINES

```
DEFINE SECU-SOC CHARACTER 11
DEFINE 1 SEC-SOC-DETAIL REDEFINES SECU-SOC,
          2 ZONE-1 CHARACTER 3,
          2 ESPACE-1 CHARACTER 1
          2 ZONE-2 CHARACTER 2,
          2 ESPACE-2 CHARACTER 1
          2 ZONE-3 CHARACTER 4
ACCEPT SEC-SOC-DETAIL
SPACE
RETRIEVE ETUDIANT WHERE SEC-SOCIALE=SECU-SOC
   PRINT E-NOM
END

SEC-SOC-DETAIL
     ZONE-1 : 140
     ESPACE-1 :
     ZONE-2 : 26
     ESPACE-2 :
     ZONE-3 : 1344

MARTIN
❑
```

5.13 DELETE

Fonction :

Suppression d'une ou plusieurs occurrences d'article dans la base de données.

Format :

```
|-------------------------------------------------|
|     DELETE <nom-article-réel> [WITH MEMBERS]    |
|-------------------------------------------------|
```

Description de l'instruction :

Cette instruction supprime une occurrence d'article et, éventuellement, tous ses articles détail.

Description des paramètres :

nom-article-réel	Nom ou synonyme de l'article à supprimer. Il doit avoir été spécifié seul dans une instruction RETRIEVE précédente dont aucune autre instruction RETRIEVE ne dépend. L'aire dont fait partie l'article doit avoir été ouverte en mode mise à jour (UPDATE). Il ne doit pas s'agir d'un article d'une vue IDS/II. Ce doit être un article UFAS ou IDS/II d'un schéma ou un article UFAS d'une vue. Dans ce dernier cas, l'article de la vue doit être identique à l'article réel du schéma et ses droits d'accès doivent permettre la suppression.
WITH MEMBERS	Lorsque cette option est spécifiée, toutes les occurrences détail reliées à l'occurrence d'article maître nom-article-réel sont également supprimées. L'aire dont elles font partie doit avoir été ouverte en mode mise à jour (UPDATE). Lorsque cette option est omise, nom-article-réel ne doit pas avoir d'occurrences détail, sinon une erreur est signalée à l'exécution de la requête.

EXEMPLE :

```
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
  WHERE C-NOM = "MEDECINE"
  PRINT E-NOM
END

ARNAUD
SERREAU
WAGNER

RETRIEVE UNIVERSITE, COLLEGE
      WHERE C-NOM = "MEDECINE"
  RETRIEVE ETUDIANT
      WHERE E-NOM = "WAGNER"
      DELETE ETUDIANT
  END
END
```

❑

Une nouvelle exécution de la première requête donne le résultat suivant :

```
ARNAUD
SERREAU
```

5.14 DISCONNECT

Fonction :

Suppression de la liaison existant entre une occurrence d'article détail IDS/II et une occurrence de son article maître.

Format :

```
|----------------------------------|
|  DISCONNECT <article-réel-1>     |
|                                  |
|      FROM <article-réel-2>       |
|                                  |
|      [VIA <nom-ensemble>]        |
|----------------------------------|
```

Description de l'instruction :

Cette instruction supprime la liaison établie entre un article membre et l'un de ses articles maîtres par l'intermédiaire d'un ensemble. Dans cet ensemble, article-réel-1 doit être défini comme article détail (MEMBER) et article-réel-2 comme article-maître (OWNER) avec la clause RETENTION OPTIONAL.

DISCONNECT ne peut être utilisée avec une vue.

Elle doit être exécutée dans le contexte d'une instruction RETRIEVE spécifiant article-réel-1 et article-réel-2.

Description des paramètres :

article-réel-1	Nom ou synonyme de l'article dont l'occurrence considérée doit être détachée de celle de l'article maître. article-réel-1 doit avoir été défini comme article détail (MEMBER) dans l'ensemble considéré et avoir été spécifié par une instruction RETRIEVE précédente. L'aire dont il fait partie doit avoir été ouverte en mode mise à jour (UPDATE).
article-réel-2	Nom ou synonyme de l'article dont l'occurrence considérée doit cesser d'être maîtresse de celle de nom article-réel-1. Il doit avoir été spécifié comme article maître (OWNER) de l'ensemble considéré. L'aire dont il fait partie doit avoir été ouverte en mode mise à jour (UPDATE).
nom-ensemble	Nom de l'ensemble reliant article-réel-1 et article-réel-2. Si les deux articles sont reliés par plusieurs ensembles, la clause VIA nom-ensemble est obligatoire.

EXEMPLE :

Supposons que le professeur JOURDAIN, dont la création et l'inclusion dans une occurrence de l'ensemble D-F (DEPARTEMENT-ENSEIGNANT) ont été illustrées dans le deuxième exemple de l'instruction CONNECT, doive être exclu de cette occurrence.

Il faut, pour ce faire, utiliser la requête suivante :

```
RETRIEVE UNIVERSITE, COLLEGE, DEPARTEMENT, ENSEIGNANT
      WHERE U-NOM = "PARIS-1"
      AND C-NOM = "MEDECINE"
      AND NOM-DEPART = "PATHOLOGIE"
      AND PROFESSEUR = "JOURDAIN"
  DISCONNECT ENSEIGNANT FROM DEPARTEMENT
END
```

❑

L'article ENSEIGNANT conserve une occurrence dans laquelle la zone PROFESSEUR a la valeur JOURDAIN, mais cette occurrence n'a plus pour article maître l'article DEPARTEMENT dans lequel la valeur de NOM-DEPART est PATHOLOGIE.

5.15 DISPLAY

Fonction :

Visualisation de données au terminal (ou sortie dans le fichier PRTFILE en traitement par lots).

Format 1 :

```
|-------------------------------------------------------------|
| DISPLAY [<expression>                                       |
|                                                             |
|         [PROMPT <expression-alphanumérique>]                |
|                                                             |
| [, <expression> [PROMPT <expression-alphanumérique>]]...]   |
|                                                             |
|       [THRU <nom-grille> [,<nom-grille>]...]                |
|-------------------------------------------------------------|
```

Format 2 :

```
|-----------------------------------------------------|
| DISPLAY [<nom-article> [,<nom-article>]...]         |
|                                                     |
|         [THRU <nom-grille> [,<nom-grille>]...]      |
|-----------------------------------------------------|
```

Format 3 :

```
|-----------------------------------------------------|
| DISPLAY [<nom-fichier>]                             |
|                                                     |
|      [THRU <nom-grille> [,<nom-grille>]...]         |
|-----------------------------------------------------|
```

Description de l'instruction :

L'instruction DISPLAY permet de visualiser des données (zones, articles, littéraux ou fichiers) au terminal. En traitement par lots, ces données vont dans le fichier PRTFILE.

VISUALISATION EN MODE GRILLE :

- Le nombre maximum de grilles par instruction DISPLAY est de 8.
- Il ne doit y avoir qu'une seule grille par écran, autrement dit, il n'est pas possible d'utiliser plusieurs grilles à la fois.
- Le nombre de lignes d'une grille utilisateur ne doit pas excéder le nombre de lignes de l'écran moins 2 (généralement 22).
- La première ligne de la grille (qui contient -->__) est fournie par IQS ; elle n'a donc pas à être définie dans la grille utilisateur. Lorsque la grille est visualisée, cette ligne permet à l'utilisateur d'introduire un code action qui peut être soit un code standard, soit un code défini par l'utilisateur. Les codes standard sont les suivants :

  <n demande de saut en arrière de n grilles
  >n demande de saut en avant de n grilles
  =n demande de visualisation de la énième grille
  / retour au niveau commande
  ; exécution de l'instruction DISPLAY avec les valeurs déjà introduites.

  Un code défini par l'utilisateur ne doit pas être un mot réservé ou un mot-clé connu du système. Il peut être lu dans une requête par l'intermédiaire de la variable système @ACTION. L'utilisation d'un tel code met fin à l'instruction DISPLAY même s'il reste des grilles. Cependant, si la variable $FUNCTION-KEY a été positionnée à 0 avant la soumission de l'instruction, les codes action ne sont pas pris en compte et toutes les grilles sont visualisées.
- La dernière ligne de la grille est fournie par IQS ; elle n'a donc pas à être définie dans la grille utilisateur. Elle permet d'envoyer un message par l'intermédiaire de la variable système @MESSAGE.

VISUALISATION EN MODE LIGNE :

L'option PROMPT permet d'insérer un texte avant la donnée visualisée.

Une seule expression est visualisée par ligne. Si l'expression est un nom de zone, le nom de la zone (sans préfixe @, $ ou #) est visualisé suivi d'un espace, de deux points, d'un espace et de la valeur de la zone. Si l'expression est une expression arithmétique, seul le résultat de l'opération est visualisé.
Le processus se répète pour chaque expression spécifiée. Lorsque la longueur des informations visualisées sur une même ligne excède la longueur de la ligne, il y a troncation à droite. Lorsque la valeur d'une zone alphanumérique n'est pas définie (c'est-à-dire à espaces), elle est visualisée sous la forme : ... (trois points).

Lorsque l'instruction DISPLAY est utilisée pour visualiser des articles, le nom de l'article est visualisé seul sur une ligne, puis les zones dont il se compose et leur valeur sont visualisées sur les lignes suivantes. Le changement de niveau de données est indiqué par un décalage de cinq positions vers la droite. Pour visualiser des articles, l'instruction DISPLAY peut spécifier soit le nom d'articles de la base de données, soit le nom d'un fichier. Dans le premier cas, DISPLAY doit être exécutée dans un bloc CREATE ou RETRIEVE et dans le second cas dans un bloc READ.

Lorsque l'instruction DISPLAY est utilisée sans paramètres, elle est associée au bloc CREATE, READ ou RETRIEVE de niveau immédiatement supérieur et toutes les zones de l'article correspondant sont visualisées.

Les données sont visualisées selon le modèle d'édition et le cadrage implicites (cf. Chapitre 2).

A noter que dans le cas d'une zone de type vecteur ou groupe, le numéro d'occurrence est visualisé après le nom de la zone et une itération automatique est effectuée pour chaque occurrence.

Description des paramètres :

expression	Expression à visualiser (format 1). Ce peut être un nom de variable permanente ou une expression alphanumérique ou numérique (cf. Chapitre 2). Si une expression numérique ou une zone contient des données numériques incorrectes, la valeur visualisée est ???.
PROMPT expression-alphanumérique	Expression alphanumérique à visualiser avant la donnée.
nom-grille	Nom de la grille utilisateur à utiliser pour visualiser les données. Le nombre maximum de grilles par instruction DISPLAY est de 8. Il ne doit y avoir qu'une seule grille par écran. Le nombre de lignes d'une grille utilisateur ne doit pas excéder le nombre de lignes de l'écran moins 2. La recherche des grilles en bibliothèque s'effectue dans l'ordre suivant : #BLIB, #BINLIB1, #BINLIB2 et #BINLIB3.
nom-article	Nom d'un article spécifié dans une instruction READ, RETRIEVE ou CREATE précédente. Toutes les zones dont il se compose sont à visualiser. L'instruction DISPLAY ne peut être exécutée qu'à la suite d'une instruction CREATE, READ ou RETRIEVE. Lorsque ce paramètre est omis, tous les articles spécifiés par le bloc RETRIEVE ou READ de niveau immédiatement supérieur ou l'article défini pour le bloc CREATE de niveau immédiatement supérieur sont visualisés. (cf. format 2)
nom-fichier	Nom du fichier lu par la précédente instruction READ. Toutes les zones du premier article défini pour ce fichier sont visualisées. Ce paramètre peut être omis lorsque l'instruction READ précède immédiatement l'instruction DISPLAY ; dans ce cas, la valeur implicite est le nom du fichier lu. (cf. format 3)

EXEMPLE 1 :

Instructions simples

```
DISPLAY E-NOM, @ADRESSE, FRAIS-SCOL
```
❑

Cette instruction permet de visualiser la valeur des trois zones spécifiées (remarquer l'absence du préfixe @ pour la variable temporaire @ADRESSE) :

```
E-NOM : MARTY
ADRESSE : 17 RUE MARTINVILLE
FRAIS-SCOL : 1200
```

L'instruction ci-dessous permet de visualiser la valeur de toutes les zones de l'article UNIVERSITE. Elle n'est valide qu'à la suite d'une instruction RETRIEVE (ayant spécifié cet article) :

```
DISPLAY UNIVERSITE
```

Résultat :

```
UNIVERSITE
     TYP : U
     U-NOM : TOULOUSE-2
     VILLE : TOULOUSE
```

EXEMPLE 2 :

Visualisation des occurrences de l'article ETUDIANT répondant aux critères spécifiés.

```
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
     WHERE U-NOM = "PARIS-1"
     AND C-NOM = "SCIENCE"
     AND N-ANNEE OF DATE-NAISSANCE LT 60
   DISPLAY ETUDIANT
END
```
❑

Résultat :

```
ETUDIANT
      TYP : E
      E-NOM : MARTY
      SEC-SOCIALE : 243 25 1313
      GRADE : MAITRISE
      DATE-NAISSANCE
           N-ANNEE : 58
           N-MOIS : 4
           N-JOUR : 19
      FRAIS-SCOL : 1200
      DIPLOME 1 : ALGEBRE
      DIPLOME 2 : MECANIQUE
      DIPLOME 3 : ...
      DIPLOME 4 : ...

ETUDIANT
      TYP : E
      E-NOM : WIART
      SEC-SOCIALE : 144 26 1316
      GRADE : MAITRISE
      DATE-NAISSANCE
           N-ANNEE : 56
           N-MOIS : 3
           N-JOUR : 27
      FRAIS-SCOL : 1500
      DIPLOME 1 : ALGEBRE
      DIPLOME 2 : ANALYSE
      DIPLOME 3 : GREC
      DIPLOME 4 : ...
```

EXEMPLE 3 :

Visualisation du résultat d'un calcul.

```
DISPLAY 123+456
```

❑

Résultat :

```
579
```

et

```
DISPLAY 123+456 PROMPT "RESULTAT"
```

Résultat :

```
RESULTAT : 579
```

EXEMPLE 4 :

Demande de visualisation des nom et grade de chaque étudiant.

```
RETRIEVE ETUDIANT
  DISPLAY E-NOM, GRADE
    THRU E-LISTE
END
```

❑

Cette requête permet de visualiser le nom et le grade de chaque étudiant au moyen de la grille E-LISTE.

5.16 DO

Fonction :

Définition d'une boucle d'itération.

Format 1 :

```
 ---------------------------------
|  [<étiquette>.] DO              |
|    [<séquence-d'instructions>]  |
|  END [<étiquette>]              |
 ---------------------------------|
```

Format 2 :

```
 ------------------------------------------------------------------
|                                      { FROM }                    |
|  [<étiquette>.] DO <variable-index> {      } <expression-1>      |
|                                      { =    }                    |
|             { TO }                                              |
|             {    } <expression-2>                               |
|             { ,  }                                              |
|                                                                 |
|             { BY } { +1           }                             |
|           [ {    } { -1           } ]                           |
|             { ,  } {<expression-3> }                            |
|                                                                 |
|  [<séquence-d'instructions>]                                    |
|                                                                 |
|  END [<étiquette>]                                              |
 ------------------------------------------------------------------
```

Description de l'instruction :

L'instruction DO définit une séquence d'instructions à exécuter plusieurs fois. La séquence commence immédiatement après l'instruction DO et se termine avec l'instruction précédant END. Une instruction REPEAT exécutée à l'intérieur de la séquence a le même effet que l'instruction END, c'est-à-dire qu'elle provoque un retour à l'instruction DO, ou plus exactement à la séquence de contrôle de boucle qui précède la première instruction de la séquence définie par DO.

Dans le premier format de DO, le nombre d'itérations est déterminé par les instructions REPEAT de la séquence. L'instruction EXIT permet de sortir de la boucle avec branchement sur la première instruction suivant END.

Dans le second format de DO, le nombre d'itérations est déterminé par la valeur d'une variable index. A la première exécution, cette variable est forcée à la valeur de <expression-1> et à chaque itération elle est augmentée de la valeur de <expression-3>, cet incrément pouvant être positif ou négatif. L'exécution de la boucle s'arrête lorsque la valeur de la variable index est supérieure (pour un incrément positif) ou inférieure (pour un incrément négatif) à celle de <expression-2>. Lorsque la valeur initiale de la variable index (expression-1) interdit l'exécution de la séquence, un branchement est immédiatement effectué sur l'instruction qui suit END. Une instruction EXIT exécutée à l'intérieur de la séquence provoque également une sortie de la boucle quelle que soit la valeur de la variable index.

Une instruction DO peut être emboîtée dans une autre instruction DO ou dans une instruction READ ou RETRIEVE ; dans ce cas, la totalité de l'instruction emboîtée (de DO à END) doit être contenue dans l'autre instruction. A noter que les instructions READ ou RETRIEVE fonctionnent également de manière itérative.

L'instruction DO peut être préfixée par une étiquette, auquel cas, celle-ci doit également figurer dans l'instruction END correspondante.

Description des paramètres :

Étiquette	Nom identifiant l'instruction DO à laquelle se rapporte une instruction END, EXIT ou REPEAT ultérieure. A noter que l'étiquette doit être suivie d'un point dans l'instruction DO, mais pas dans l'instruction END.
Séquence-d'instructions	Séquence d'instructions à exécuter itérativement. Elle peut comporter n'importe quelles instructions du langage de requêtes, y compris des instructions DO. Dans le format 1 de DO, le nombre d'itérations est déterminé par les instructions EXIT et REPEAT. Dans le format 2, il est déterminé par la variable index et les expressions spécifiées, mais les instructions EXIT et REPEAT peuvent également être utilisées. REPEAT renvoie au début de la boucle pour traitement de la variable index (incrémentation ou décrémentation selon expression-3 et comparaison avec expression-2).
Variable-index	Variable index forcée à la valeur de expression-1 au début de la boucle et incrémentée à chaque itération. La valeur de cette variable peut être modifiée par une instruction LET en cours d'exécution, cette modification pouvant influer sur le nombre d'itérations. A la fin de la boucle, la valeur de la variable index n'est plus définie. La variable index ne peut être qu'une zone de type numérique.
FROM expression-1	expression-1 est une expression numérique qui fournit la valeur initiale de la variable index. Cette expression étant évaluée au début de la boucle, les modifications apportées à ses éléments en cours d'exécution n'affectent ni le nombre d'itérations, ni la valeur de la variable index.
TO expression-2	expression-2 est une expression numérique qui indique la valeur limite de la variable index. Cette expression étant évaluée au début de la boucle, les modifications apportées à ses éléments en cours d'exécution n'affectent pas le nombre d'itérations. L'exécution de la boucle se termine lorsque la valeur de la variable index est supérieure (si expression-3 est positive) ou inférieure (si expression-3 est négative) à la valeur de expression-2. Lorsque la valeur initiale de la variable index (expression-1) interdit l'exécution de la séquence, un branchement est immédiatement effectué sur l'instruction qui suit END.

BY expression-3	expression-3 est une expression numérique qui représente la valeur de l'incrément applicable à la variable index à chaque itération. Cette expression étant évaluée au début de la boucle, les modifications apportées à ses éléments en cours d'exécution n'affectent pas la valeur de l'incrément. La valeur de expression-3 peut être positive, négative ou égale à zéro. Dans ce dernier cas, la fin de la boucle doit être provoquée par une instruction EXIT. Ce paramètre est facultatif et sa valeur implicite est +1.

EXEMPLE 1 :

Nombre d'itérations déterminé par des instructions REPEAT et EXIT.

```
ETIQ1. DO
    .
    .
    .
    IF $NBR < 0
    THEN REPEAT ETIQ1
    ELSE
       ETIQ2. DO
            .
            .
            .
       END ETIQ2
       EXIT ETIQ1
    END
END ETIQ1
```

❑

La boucle extérieure commence à ETIQ1. DO et se termine à END ETIQ1. Son nombre d'itérations est déterminé par un test de la variable $NBR. Lorsque celle-ci n'est plus négative, la boucle intérieure ETIQ2 est lancée ; le nombre d'itérations de ETIQ2 peut être déterminé par des instructions EXIT et REPEAT qui ne figurent pas dans l'exemple. La fin de ETIQ2 est indiquée par END ETIQ2. L'instruction EXIT ETIQ1 met fin à l'exécution de la boucle extérieure ETIQ1. En fait, cette instruction est superflue puisque, de toute façon, l'exécution passe à l'instruction END ETIQ1.

EXEMPLE 2 :

```
    ACCEPT @DIPLOME
    SPACE
    RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
          WHERE U-NOM = "PARIS-1"

       L ET $N = 1
       E1.DO
          IF $N > 4 THEN EXIT E1 END
          IF DIPLOME($N) = @DIPLOME
          THEN
             PRINT E-NOM COL 9
             EXIT E1
          ELSE
             LET $N = $N + 1
             REPEAT E1
          END
       END E1
    END
```

❑

Résultat (les données introduites par l'utilisateur sont soulignées) :

```
DIPLOME : ALGEBRE

          SERREAU
          MARTY
          WIART
```

Cette requête pourrait également s'écrire de la manière suivante :

```
ACCEPT @DIPLOME
SPACE
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
      WHERE U-NOM = "PARIS-1"
   E1.DO $N = 1 TO 4
      IF DIPLOME($N) = @DIPLOME
      THEN
         PRINT E-NOM COL 9
         EXIT E1
      END
   END E1
END
```

Nombre d'itérations déterminé par des expressions :

EXEMPLE 3 :

```
DO $I = 2 TO 10 BY 2
   .
   .
END
```
❑

La boucle est exécutée 5 fois.

Si l'on remplace l'instruction DO par :

```
DO $I = 10 TO 2 BY -2
```

elle est également exécutée 5 fois.

EXEMPLE 4 :

Une boucle définie par :

```
DO $I = 10 TO 12 BY -2
```
ou
```
DO $I = 12 TO 10
```
❑

n'est pas exécutée.

EXEMPLE 5 :

Les quatre instructions DO suivantes sont équivalentes :

```
DO $I = 1 TO 10
DO $I = 1,10
DO $I = 1 TO 10 BY 1
DO $I = 1, 10, 1
```
❑

EXEMPLE 6 :

Nombre d'itérations déterminé par des expressions et des instructions EXIT et REPEAT.

```
DO $I = 1 TO 100
   .
   .
   IF $NBR > $CAL
   THEN REPEAT
   ELSE LET $I = $VAL
   END
   .
   .
   IF $TOTAL > $ANC-TOTAL THEN EXIT END
   .
   .
END
```

❑

Si la valeur de $NBR est supérieure à celle de $CAL, l'instruction REPEAT est exécutée et renvoie au début de la boucle. La variable index $I est incrémentée et comparée à 100 ; si elle est inférieure ou égale à 100, l'exécution de la boucle se poursuit. Si la valeur de $NBR est inférieure ou égale à celle de $CAL, la clause ELSE est exécutée et la variable index prend la valeur de $VAL. A noter que cette modification affecte le nombre d'itérations et qu'il faut prévoir un moyen de sortir de la boucle.

Si la valeur de $TOTAL est supérieure à celle de $ANC-TOTAL, l'instruction EXIT met fin à la boucle quelle que soit la valeur de $I.

6. Instructions du langage de requêtes IQS (E-P)

6.1 Introduction

Le présent chapitre fournit une description des instructions du langage de requêtes de la lettre E à la lettre P incluse. Pour les autres instructions se reporter au chapitre précédent et suivant.

Une requête comprend un ensemble d'instructions et des commentaires facultatifs. Se reporter à la présentation générale du langage de requêtes fournie au début du chapitre 5.

En ce qui concerne la création de requêtes utilisables sous TDS, se reporter au guide utilisateur IQS-V4/TDS.

Pour tous renseignements complémentaires en matière de programmation, consulter le manuel d'initiation au langage de requêtes et le guide du programmeur.

6.2 EJECT

Fonction :

Effacement de l'écran, avec positionnement du curseur en début d'écran.

Format :

```
EJECT
```

Description de l'instruction :

L'instruction EJECT permet d'effacer l'écran et de positionner le curseur en haut du nouvel écran. Cette instruction est équivalente à l'instruction SPACE TOP (saut de page) lors de l'impression d'un état (instruction PRINT).

L'exécution de cette instruction ne change pas la valeur de la variable système $PAGE.

Si les sorties sont envoyées dans un fichier d'impression, la variable $LINE est forcée à 1.

6.3 END

Fonction :

Clôture d'un bloc CREATE, DO, IF, READ ou RETRIEVE, ou d'une séquence d'instructions figurant à la suite d'une instruction de contrôle.

Format :

```
END [<étiquette>]
```

Description de l'instruction :

L'instruction END est obligatoire pour indiquer la fin d'un bloc commençant par une instruction CREATE, DO, IF, READ ou RETRIEVE, ou une instruction de contrôle.

La détection d'une instruction EXIT, dans une boucle définie par l'un des blocs ci-dessus, provoque une sortie de la boucle avec branchement sur la première instruction suivant END.

En cas de boucles emboîtées, l'utilisation du paramètre étiquette est recommandée pour une meilleure lisibilité.

Description des paramètres :

étiquette	Paramètre facultatif servant à identifier l'instruction début de bloc correspondante.

6.4 EXECUTE (EXEC)

Fonction :

Appel d'une autre requête (sous-requête) pour exécution.

Format :

```
 ------------------------------------------------
|          { *(<expression-alphanumérique>) }    |
|  EXECUTE {                                 }    |
|          { <nom-requête-appelée>           }    |
|                                                |
|       [(<argument> [,<argument>]...)]          |
 ------------------------------------------------
```

Description de l'instruction :

Cette instruction permet d'appeler une autre requête pour exécution. Chaque argument spécifié est associé au paramètre correspondant défini dans l'instruction PARAMETER de la requête appelée et un branchement est effectué sur la première instruction exécutable.

A noter que l'occurrence courante de l'article éventuellement spécifié dans la requête appelante n'est pas prise en compte dans la requête appelée (voir exemple 2).

Description des paramètres :

*(expression-alphanumérique)	Cette option permet de spécifier un nom de requête par l'intermédiaire d'une expression alphanumérique, au lieu de le fournir directement. Les parenthèses sont obligatoires même si l'expression n'est constituée que d'un nom de variable.
nom-requête-appelée	Nom d'une requête rangée dans la bibliothèque binaire BINLIB.
argument	Nom d'une zone accessible à la requête appelée et déclarée dans cette dernière au moyen de l'instruction PARAMETER. Les paramètres spécifiés dans PARAMETER correspondent terme à terme aux arguments spécifiés dans EXECUTE, le premier argument correspondant au premier nom de zone (de niveau 1 ou sans numéro de niveau) spécifié dans PARAMETER et ainsi de suite. Le nombre d'arguments doit être égal au nombre de paramètres et la définition de chaque argument doit être identique à celle du paramètre qui lui correspond.

Exemple 1 :

REQUETE-1 permet d'examiner un certain nombre d'occurrences de l'article ETUDIANT et d'imprimer le nom des étudiants dont le grade correspond à celui spécifié. Elle appelle REQUETE-2 qui lui fournit les valeurs à utiliser. REQUETE-1 est donc une requête "générale" alors que REQUETE-2 permet de spécifier le nombre d'occurrences à examiner et de sélectionner le grade.

REQUETE-1 (requête principale) :

```
LET $I = 0
DEFINE $NOMBRE UNSIGNED DECIMAL 4
DEFINE @GRADE CHARACTER 20
   EXECUTE REQUETE-2 ($NOMBRE,@GRADE)
RETRIEVE ONLY $NOMBRE ETUDIANT
         WHERE GRADE=@GRADE
   PRINT E-NOM
END
```

REQUETE-2 (sous-requête) :

```
PARAMETER $NOMBRE UNSIGNED DECIMAL 4,
          @GRADE CHARACTER 20
ACCEPT $NOMBRE PROMPT "EXAMEN DE COMBIEN D'ARTICLES"
ACCEPT @GRADE PROMPT "QUEL GRADE"
RETURN
```

Résultat de REQUETE-1 avec les valeurs 5 et MAITRISE :

```
EXAMEN DE COMBIEN D'ARTICLES : 5
QUEL GRADE : MAITRISE
SERREAU
WAGNER
MARTY
```

Voir également l'instruction PARAMETER.

Exemple 2 : Utilisation incorrecte d'EXECUTE

REQUETE-1 permet de rechercher des occurrences de l'article UNIVERSITE et de les imprimer. Elle appelle REQUETE-2, au moyen de l'instruction EXECUTE spécifiée à l'intérieur du bloc RETRIEVE, pour rechercher et imprimer les occurrences de l'article COLLEGE.

REQUETE-1 :

```
RETRIEVE UNIVERSITY
  PRINT UNIVERSITY
  EXEC REQUETE-2
END
```

REQUETE-2 (sous-requête) :

```
RETRIEVE COLLEGE
  PRINT COLLEGE
END
```

Résultat :

Les deux requêtes étant indépendantes, l'occurrence de l'article UNIVERSITE sélectionnée dans REQUETE-1 n'est pas connue de REQUETE-2 : toutes les occurrences de l'article COLLEGE sont donc imprimées à chaque exécution de la sous-requête.

Pour sélectionner les occurrences de COLLEGE se rapportant à l'occurrence courante de l'article UNIVERSITE, la deuxième instruction RETRIEVE doit être emboîtée dans le premier bloc RETRIEVE, comme dans REQUETE-3 ci-dessous.

REQUETE-3 :

```
RETRIEVE UNIVERSITY
  PRINT UNIVERSITY
  RETRIEVE COLLEGE
  PRINT COLLEGE
  END
END
```

6.5 EXIT

Fonction :

Sortie d'une boucle d'itération définie par un bloc CREATE, DO, READ ou RETRIEVE.

Format :

```
EXIT [<étiquette>]
```

Description de l'instruction :

L'instruction EXIT met fin à l'exécution d'une boucle définie par un bloc CREATE, DO, READ ou RETRIEVE. Un branchement est effectué sur l'instruction qui suit l'instruction END de la boucle.

Une même boucle peut contenir plusieurs instructions EXIT ; c'est la première exécutée qui provoque la sortie de la boucle.

Lorsque l'instruction qui définit la boucle comporte une étiquette, cette dernière doit également figurer dans EXIT. Une instruction EXIT sans étiquette est considérée comme se rapportant à l'instruction CREATE, DO, READ ou RETRIEVE de niveau immédiatement supérieur. L'instruction EXIT doit se trouver à l'intérieur du bloc auquel elle s'applique.

Une instruction EXIT ne doit pas porter la même étiquette qu'une instruction de contrôle.

Description des paramètres :

étiquette	Lorsque ce paramètre est spécifié, il doit correspondre à l'étiquette d'une instruction CREATE, DO, READ ou RETRIEVE précédente. Lorsqu'il n'est pas spécifié, l'instruction EXIT est associée à l'instruction CREATE, DO, READ ou RETRIEVE de niveau immédiatement supérieur.

EXEMPLE :

```
ET1. RETRIEVE UNIVERSITE
   .
   .
   ET2. RETRIEVE COLLEGE
      .
      .
      EXIT ET1
      .
      .
   END ET2
   .
   .
END ET1
```

❑

L'instruction EXIT ET1 provoque la sortie de la boucle lancée par l'instruction RETRIEVE portant l'étiquette ET1, et par conséquent de la boucle emboîtée lancée par l'instruction RETRIEVE portant l'étiquette ET2.

6.6 FOOTING, HEADING, HEADING AND FOOTING

Fonction :

Demande d'impression des titres d'un état en haut de page (HEADING), en bas de page (FOOTING), ou en haut et en bas de page (HEADING AND FOOTING).

Format :

```
 ------------------------------------------------------------
|  { FOOTING             }                                   |
|  { HEADING             }   <étiquette> [,<étiquette>] ...  |
|  { HEADING AND FOOTING }                                   |
 ------------------------------------------------------------
```

Description de l'instruction FOOTING :

L'instruction FOOTING permet de demander l'impression des titres en bas des colonnes. Ces titres sont ceux spécifiés par les instructions PRINT dont l'étiquette est citée dans FOOTING. Ces instructions PRINT doivent concerner le même fichier de sortie. Lorsque les titres se chevauchent, il y a recouvrement.

L'effet de FOOTING est similaire à celui d'une instruction PRINT WITH TITLE comportant tous les éléments spécifiés dans les instructions PRINT dont l'étiquette est citée, à la différence que les titres sont imprimés en bas des colonnes et non en haut.

REMARQUE :

Le titre peut comporter jusqu'à 3 lignes. Le nombre et le contenu de ces lignes sont définis par les éléments spécifiés dans les instructions PRINT dont l'étiquette est citée (cf. instruction PRINT).

Description de l'instruction HEADING :

L'instruction HEADING permet de demander l'impression des titres en haut des colonnes. Ces titres sont ceux spécifiés par les instructions PRINT dont l'étiquette est citée dans HEADING. Toutes ces instructions PRINT doivent concerner le même fichier de sortie. Lorsque les titres se chevauchent, il y a recouvrement.

L'effet de HEADING est similaire à celui d'une instruction PRINT WITH TITLE comportant les éléments spécifiés dans les instructions PRINT dont l'étiquette est citée.

REMARQUE :

Le titre peut comporter jusqu'à 3 lignes. Le nombre et le contenu de ces lignes sont définis par les éléments spécifiés dans les instructions PRINT dont l'étiquette est citée (cf. instruction PRINT).

Description de l'instruction HEADING AND FOOTING :

Cette instruction combine les fonctions des instructions HEADING et FOOTING, c'est-à-dire qu'elle permet de demander l'impression des titres à la fois en haut et en bas des colonnes.

Lorsque les options TITLE des instructions PRINT dont les étiquettes sont spécifiées dans HEADING et/ou FOOTING contiennent des expressions, tous les éléments dont celles-ci se composent doivent avoir une valeur lorsque HEADING et/ou FOOTING est utilisée.

Le paramètre NCOL de l'instruction PRINT permet d'affecter une donnée à une colonne autre que celle déterminée par l'instruction HEADING et/ou FOOTING, NCOL 1 spécifiant la colonne la plus à gauche, NCOL 2 la colonne suivante, etc. Son utilisation est illustrée dans l'exemple ci-dessous.

EXEMPLE :

```
SPACE TOP
LET @COLUMN-BORDER = "!"
LET @LINE-BORDER = "-"
LET $COLUMN-SPACING = 3
HEADING L1,
        L2,
        L3
RETRIEVE UNIVERSITE WHERE VILLE PRESENT
   LET $NBR-COLLEGE = 0
   LET $NBR-ETUD = 0
L1.
   PRINT VILLE TITLE "UNIVERSITE"
   BEFORE UNIVERSITE CHANGE
     PRINT (20)"-" NCOL 2
        PRINT "NOMBRE DE COLLEGES" NCOL 1,
              $NBR-COLLEGE NCOL 2
        PRINT "NOMBRE D'ETUDIANTS" NCOL 1,
              $NBR-ETUD NCOL 3
     SPACE
   END
   RETRIEVE COLLEGE
     LET $NBR-COLLEGE = COUNT
     LET $NBR-ETUDIANT = 0
```

```
L2.  PRINT C-NOM TITLE "COLLEGE"
     BEFORE COLLEGE CHANGE
        PRINT (20) "-" NCOL 3
        PRINT "NOMBRE D'ETUDIANTS" NCOL 2,
              $NBR-ETUDIANT NCOL 3
     END

      RETRIEVE ETUDIANT
         LET $NBR-ETUD = COUNT
         LET $NBR-ETUDIANT = COUNT
L3.      PRINT E-NOM TITLE "ETUDIANT"
      END
   END
END
```

❑

Le tableau produit est le suivant :

UNIVERSITE	COLLEGE	ETUDIANT
PARIS		
	MEDECINE	
		ARNAUD
		SERREAU
		WAGNER
	NOMBRE D'ETUDIANTS	3
	PHARMACIE	
	NOMBRE D'ETUDIANTS	0
	SCIENCE	
		MARTIN
		MARTY
		SARRE
		WIART
	NOMBRE D'ETUDIANTS	4
NOMBRE DE COLLEGES	3	
NOMBRE D'ETUDIANTS		7
TOULOUSE		
	ECONOMIE	
	NOMBRE D'ETUDIANTS	0
NOMBRE DE COLLEGES	1	
NOMBRE D'ETUDIANTS		0

6.7 IF

Fonction :

Définition d'une condition déterminant l'exécution d'une séquence d'instructions.

Format :

```
|-------------------------------------------------|
| [<étiquette>.] IF <expression-conditionnelle>   |
|                 THEN [<séquence-1>]             |
|                [ELSE [<séquence-2>]]            |
|                END [<étiquette>]                |
|-------------------------------------------------|
```

Description de l'instruction :

Lorsque IF est exécutée, l'expression conditionnelle est évaluée. Si la condition est vérifiée, la séquence d'instructions spécifiée par THEN <séquence-1> est exécutée et le traitement continue avec la première instruction qui suit END.

Si la condition n'est pas vérifiée et que ELSE <séquence-2> est spécifié, <séquence-2> est exécutée et le traitement continue avec la première instruction qui suit END. Si la condition n'est pas vérifiée et que ELSE <séquence-2> n'est pas spécifié, un branchement est effectué sur la première instruction qui suit END.

L'instruction IF peut être préfixée d'une étiquette, auquel cas cette dernière doit également figurer dans l'instruction END correspondante. Cette étiquette ne peut être spécifiée dans une instruction EXIT ou REPEAT. <séquence-1> et <séquence-2> peuvent comporter d'autres instructions IF.

En ce qui concerne l'utilisation de l'instruction IF pour tester la valeur de la variable @STATUS, se reporter au chapitre 3 du présent manuel et au guide du programmeur IQS-V4 (79UR).

Description des paramètres :

étiquette	Nom permettant d'identifier l'instruction IF à laquelle se rapporte une instruction END ultérieure. Cette étiquette ne doit pas être spécifiée dans une instruction EXIT ou REPEAT. A noter que l'étiquette doit être suivie d'un point dans l'instruction IF, mais pas dans l'instruction END.
expression-conditionnelle	Ce type d'expression est traité au chapitre 2. Après évaluation, la condition exprimée est testée. Si elle est vérifiée, la séquence spécifiée par THEN (séquence-1) est exécutée et la séquence spécifiée par ELSE (séquence-2) est sautée. Si elle n'est pas vérifiée, séquence-1 est sautée et séquence-2 est exécutée.
THEN séquence-1	séquence-1 est la séquence à exécuter si la condition est vérifiée. Elle peut contenir d'autres instructions IF. Cette séquence est facultative.
ELSE séquence-2	séquence-2 est la séquence à exécuter si la condition n'est pas vérifiée. Elle peut contenir d'autres instructions IF. Lorsqu'elle est spécifiée, cette séquence doit être précédée de ELSE. Lorsqu'elle est omise, un branchement est effectué sur la première instruction qui suit END.

EXEMPLE 1 :

```
LET $NOMBRE-MAITRISES = 0
LET $NOMBRE-LICENCES = 0
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
       WHERE U-NOM = "PARIS-1"
   IF GRADE = "LICENCE"
   THEN LET $NOMBRE-LICENCES = $NOMBRE-LICENCES + 1
   ELSE LET $NOMBRE-MAITRISES = $NOMBRE-MAITRISES + 1
   END
END
DISPLAY $NOMBRE-LICENCES, $NOMBRE-MAITRISES
```

Résultat :

```
NOMBRE-LICENCES : 3
NOMBRE-MAITRISES : 4
```

❑

EXEMPLE 2 :

```
E1.IF $TEST-1 GE 0
THEN IF $TEST-2 GE 0
     THEN LET $C=0
     ELSE LET $C=1
     END
ELSE IF $TEST-2 GE 0
     THEN LET $C=2
     ELSE LET $C=3
     END
END E1
```

❑

L'étiquette E1 de la dernière instruction END correspond à l'étiquette E1 de la première instruction IF. Le résultat de cette séquence d'instructions est illustré par le tableau suivant :

Valeur de $TEST-1	Valeur de $TEST-2	Valeur finale de $C
positive ou égale à zéro	positive ou égale à zéro	0
positive ou égale à zéro	négative	1
négative	positive ou égale à zéro	2
négative	négative	3

EXEMPLE 3 : Test de @STATUS

```
ACCEPT @NO-SEC
RETRIEVE ETUDIANT WHERE SEC-SOCIALE = @NO-SEC
   PRINT SEC-SOCIALE, GRADE, E-NOM
END
IF @STATUS = KEYUNKN
THEN PRINT "AUCUN ETUDIANT POUR CETTE CLE"
END
```

❑

Résultat (les données introduites par l'utilisateur sont soulignées) :

```
NO-SEC : 123 45 6789
AUCUN ETUDIANT POUR CETTE CLE
```

REMARQUE :

Si cet exemple était utilisé avec un schéma UFAS séquentiel, l'instruction PRINT "AUCUN ETUDIANT POUR CETTE CLE" ne serait jamais exécutée.

6.8 INSERT

Fonction :

Insertion d'une nouvelle occurrence d'article dans une base de données UFAS séquentiel indexé, UFAS relatif ou IDS/II.

Format 1 :

```
|-----------------------------------------------------|
| INSERT <nom-article-réel-1> [WITHIN <nom-aire>]     |
|                                                     |
|   [IN <nom-article-réel-2> [VIA <nom-ensemble>]     |
|                                                     |
|     [,<nom-article-réel-2> [VIA <nom-ensemble>]]...]|
|-----------------------------------------------------|
```

Format 2 (UFAS relatif uniquement) :

```
 -------------------------------------------------------
|  INSERT <nom-article-réel-1>  <expression-numérique>  |
 -------------------------------------------------------
```

Description de l'instruction :

Cette instruction permet d'insérer une nouvelle occurrence d'article réel dans une base de données et dans le cas d'IDS/II, de l'inclure dans tous les ensembles où l'article est défini comme article détail (MEMBER) avec la clause INSERTION IS AUTOMATIC. L'article doit avoir été spécifié dans une instruction CREATE précédente. L'article désigné par nom-article-réel-2 doit avoir été spécifié dans une instruction RETRIEVE précédente.

Il est possible d'effectuer une série d'insertions, en spécifiant plusieurs instructions INSERT dans un même bloc CREATE. En cas d'insertions en série dans une base de données IDS/II, la procédure est la suivante :

- Au début du bloc CREATE, l'article maître est considéré comme l'article courant de type d'ensemble. Par conséquent, pour la première insertion, une clause INSERTION NEXT dans le DDL est traitée comme s'il s'agissait d'une clause INSERTION FIRST et une clause INSERTION PRIOR comme s'il s'agissait d'une clause INSERTION LAST. Les autres modes d'insertion sont traités normalement.

- Les insertions suivantes à l'intérieur du même bloc CREATE sont effectuées normalement, le dernier article inséré étant considéré comme l'article courant de type d'ensemble.
- Le bloc CREATE conserve son propre point courant qui ne peut en aucun cas être modifié par une instruction RETRIEVE spécifiée à l'intérieur du bloc.

INSERT peut être utilisée pour insérer un article dans un fichier UFAS séquentiel indexé, mais non dans un fichier UFAS séquentiel. Il peut s'agir d'un article d'un schéma ou bien d'un article d'une vue à condition qu'il corresponde à un article réel décrit dans le schéma. Dans les deux cas, les droits d'accès de l'article doivent permettre l'insertion.

INSERT peut être utilisée pour insérer un article dans un fichier UFAS relatif (cf. le format 2). L'aire associée à cet article doit être déclarée comme RELATIVE dans le schéma DDL.

En cas d'insertion d'un article de longueur variable, la valeur appropriée doit être spécifiée dans la clause "DEPENDING ON".

IQS utilise dans la base de données la longueur maximum de l’enregistrement à insérer. Cela permet de changer la valeur du champ de contrôle par la commande MODIFY ou pendant la sous-commande MODIFY de REVIEW.

Description des paramètres :

nom-article-réel-1	Nom ou synonyme de l'article à insérer. Cet article doit avoir été spécifié dans une instruction CREATE précédente. L'aire à laquelle il est associé doit avoir été ouverte en mode mise à jour (UPDATE). Lorsque cet article est article détail dans le schéma, l'instruction INSERT doit être exécutée dans le contexte d'une ou plusieurs instructions RETRIEVE spécifiant tous les articles maîtres des ensembles dont il est détail. Dans le cas d'IDS/II, ce sont seulement les ensembles où il a été défini avec la clause INSERTION AUTOMATIC.
nom-aire	Nom de l'aire où l'article est à insérer. Lorsque cet article appartient à plusieurs aires IDS/II et que sa clause LOCATION MODE spécifie CALC, la clause WITHIN est obligatoire.
nom-article-réel-2	Nom ou synonyme de l'article réel défini comme article maître de l'ensemble. Il doit avoir été spécifié dans une instruction RETRIEVE précédente et l'aire dont il fait partie doit avoir été ouverte en mode mise à jour (UPDATE). Lorsque l'article à insérer est défini comme article détail (MEMBER), avec la clause INSERTION AUTOMATIC dans plusieurs ensembles, tous les articles maîtres de ces ensembles **doivent** être spécifiés. Ce paramètre ne peut être utilisé que dans le cas d'une base de données IDS/II.
nom-ensemble	Nom de l'ensemble dans lequel l'article à insérer est défini comme article détail (MEMBER) avec la clause INSERTION AUTOMATIC. Lorsque l'article à insérer et son article maître sont reliés par plusieurs ensembles, la clause VIA est obligatoire. VIA ne peut être utilisée que dans le cas d'une base de données IDS/II. Lorsque la clause LOCATION MODE de l'article à insérer spécifie VIA nom-ensemble et qu'il est défini comme article détail de cet ensemble avec la clause INSERTION MANUAL, l'article maître de l'ensemble doit avoir été spécifié par une instruction RETRIEVE précédente. Dans ce cas, l'instruction INSERT n'inclut pas l'article dans l'ensemble ; pour ce faire, c'est l'instruction CONNECT qui doit être utilisée.
Expression-numérique	Valeur de la clé $RECORD-NUMBER, sous la forme d'un entier positif.

Exemple 1 : Insertion dans un fichier UFAS séquentiel indexé

La requête suivante permet d'examiner successivement les différents collèges de chaque université afin d'y insérer de nouveaux étudiants :

```
RETRIEVE UNIVERSITE,COLLEGE
  DISPLAY U-NOM,C-NOM
  CREATE ETUDIANT
     ACCEPT
     IF E-NOM ABSENT
     THEN EXIT
     END
     INSERT ETUDIANT
     REPEAT
  END
END
```

Le système indique les noms de l'université et du collège sur lesquels il est positionné et demande d'introduire des valeurs pour les différentes zones de l'occurrence d'article ETUDIANT créée :

```
U-NOM : PARIS-1
C-NOM : MEDECINE
ETUDIANT
     TYP : E
     E-U-NOM : PARIS-1
     E-C-NOM : MEDECINE
     E-NOM : DUVAL
     SEC-SOCIALE : 142 12 6321
```

Exemple 2 : Insertion simple dans une base de données IDS/II

La requête suivante fonctionne de la même manière que la précédente mais permet d'insérer de nouveaux collèges :

```
RETRIEVE UNIVERSITE
  DISPLAY U-NOM
  CREATE COLLEGE
     ACCEPT IF C-NOM ABSENT
     THEN EXIT
     END
     INSERT COLLEGE
     REPEAT
  END
END
```

Exemple 3 : Insertion dans une base IDS/II avec spécification d'un article maître

Requête destinée à insérer un nouveau cours dans la base de données ; ce cours doit dépendre d'une certaine occurrence de l'article ENSEIGNANT qui est elle-même détail d'un certain collège et d'une certaine université :

```
RETRIEVE UNIVERSITE, COLLEGE, ENSEIGNANT
     WHERE U-NOM = "PARIS-1"
        AND C-NOM = "MEDECINE"
        AND PROFESSEUR = "DUPONT"
  CREATE COURS
     ACCEPT
     INSERT COURS
  END
END
```

Lorsque cette requête est exécutée, les guidages suivants sont visualisés et l'utilisateur peut introduire les données concernant le nouveau cours :

```
COURS
     NOM-COURS : ANATOMIE
     NO-COURS : 733
     DATE-COURS : 071086
```

Exemple 4 : Insertion dans une base IDS/II avec spécification de deux articles maîtres

Dans le schéma (cf. annexe A), l'article ETUDIANT est défini comme ayant deux articles maîtres : COLLEGE et ENSEIGNANT. Dans les deux cas, sa clause INSERTION spécifie AUTOMATIC. La requête suivante permet d'insérer une nouvelle occurrence de l'article ETUDIANT dans la base en l'incluant automatiquement dans les deux ensembles :

```
RETRIEVE UNIVERSITE, COLLEGE, ENSEIGNANT
     WHERE U-NOM = "PARIS-1"
        AND C-NOM = "MEDECINE"
        AND PROFESSEUR = "DUPONT"
  CREATE ETUDIANT
     ACCEPT
     INSERT ETUDIANT IN COLLEGE VIA C-E,
                        ENSEIGNANT VIA EN-E
  END
END
```

Exemple 5 : Insertion dans une base IDS/II avec spécification de deux articles maîtres et inclusion automatique et manuelle

Supposons que dans l'exemple 3 le cours ANATOMIE ait été créé et que dans l'exemple 4 l'étudiant BARON ait été créé. Il est maintenant possible de créer une occurrence de l'article HORAIRE qui soit détail de ces deux occurrences. Elle est incluse automatiquement dans l'ensemble C-HR mais, pour l'inclure dans l'ensemble E-HR, il faut utiliser l'instruction CONNECT (cf. annexe A) :

```
RETRIEVE COURS WHERE NOM-COURS = "ANATOMIE"
  RETRIEVE ETUDIANT WHERE E-NOM = "BARON"
     CREATE HORAIRE
        ACCEPT
        INSERT HORAIRE IN COURS
        CONNECT HORAIRE TO ETUDIANT
     END
  END
END
```

6.9 Instructions de contrôle

Fonction :

Définition d'une séquence d'instructions exécutable au début ou à la fin de chaque boucle de l'instruction READ ou RETRIEVE à laquelle elle est associée, lorsque la condition spécifiée est remplie.

Format 1 :

```
.-------------------------------------------------------------.
|                           { <nom-article>             }     |
|                { AFTER  } { <nom-zone>                } CHANGE |
| [<étiquette>.] { BEFORE } { <nom-variable-temporaire> }     |
|                           { <expression-sous-chaîne>  }     |
|                                                             |
|                     {<nom-article>             }            |
|            { AND } {<nom-zone>                 }            |
|          [ { OR  } {<nom-variable-temporaire> }  CHANGE ] ...  |
|                     {<expression-sous-chaîne>  }            |
|                                                             |
|     <séquence-d'instructions>                               |
|                                                             |
| END [<étiquette>]                                           |
 -------------------------------------------------------------
```

Format 2 :

```
|-----------------------------------------------|
|  [<étiquette>.]  ON ABSENT                    |
|                  <séquence-d'instructions>    |
|                  END [<étiquette>]            |
|-----------------------------------------------|
```

Description de l'instruction :

L'instruction de contrôle est suivie d'une séquence d'instructions terminée par END. Cette séquence est exécutée chaque fois que la condition spécifiée est remplie. La condition exprimée par CHANGE ou ON ABSENT porte sur le résultat de la boucle READ ou RETRIEVE courante ; ces deux formats sont analysés dans les pages suivantes.

Avec le format CHANGE, une comparaison est effectuée à chaque boucle de l'instruction READ ou RETRIEVE pour déterminer si l'occurrence d'article ou la zone spécifiée a changé. Si elle est différente, la condition est vérifiée et la séquence d'instructions est exécutée.
Plusieurs conditions peuvent être combinées au moyen des opérateurs logiques AND et OR ; dans ce cas, l'expression est évaluée comme une expression conditionnelle composée et la séquence n'est exécutée que si la condition est vérifiée. La condition CHANGE est assortie d'un mot-clé BEFORE ou AFTER.

Lorsque BEFORE est spécifié, la séquence est exécutée avec les valeurs de l'ancien article (c'est-à-dire de l'article fourni par la boucle précédente de READ ou RETRIEVE) ; à noter qu'elle est exécutée de manière systématique après la dernière boucle READ ou RETRIEVE, bien que dans ce cas, la condition CHANGE ne soit pas vérifiée.

Lorsque AFTER est spécifié, la séquence est exécutée avec les valeurs de l'article courant (c'est-à-dire de l'article fourni par la boucle courante de READ ou RETRIEVE) ; à noter qu'elle est exécutée systématiquement avant la première boucle READ ou RETRIEVE bien que dans ce cas la condition CHANGE ne soit pas vérifiée.

Avec le format ON ABSENT, la séquence d'instructions n'est exécutée que si la boucle READ ou RETRIEVE ne fournit aucun article, à savoir lorsque le fichier est vide, lorsque l'article spécifié n'a pas d'occurrences ou lorsqu'aucune occurrence ne répond aux critères de sélection définis dans l'instruction READ ou RETRIEVE. Lorsque la condition ON ABSENT est vérifiée, aucune condition AFTER CHANGE ou BEFORE CHANGE spécifiée pour la même boucle READ ou RETRIEVE n'est remplie.

Un bloc READ ou RETRIEVE peut contenir plusieurs instructions CHANGE mais ne peut comporter qu'une seule instruction ON ABSENT. La totalité d'une instruction de contrôle doit se trouver à l'intérieur du bloc auquel elle est associée. Les emboîtements d'instructions de contrôle sont interdits.

Il est possible de spécifier une étiquette pour une instruction de contrôle.

Pour plus de détails sur le traitement des instructions de contrôle, se reporter au guide du programmeur IQS-V4 (79UR).

Description des paramètres :

étiquette	Nom permettant d'identifier l'instruction ; elle doit également être spécifiée dans l'instruction END correspondante.
AFTER/BEFORE	Ces mots-clés déterminent avec quelles valeurs doit être exécutée la séquence d'instructions (cf. ci-dessus).
nom-article	Nom d'un article spécifié dans l'instruction READ ou RETRIEVE associée. Si la boucle courante de l'instruction READ ou RETRIEVE a fourni une nouvelle occurrence de cet article, la condition AFTER/BEFORE CHANGE est vérifiée. Si la boucle n'a fourni aucune occurrence de cet article (parce qu'aucune d'entre elles ne répondait aux critères de sélection par exemple), la condition ON ABSENT est vérifiée. Lorsque l'instruction READ ou RETRIEVE fournit une ou plusieurs occurrences de cet article : 1. la condition AFTER CHANGE est vérifiée pour la première occurrence et 2. la condition BEFORE CHANGE est vérifiée pour la dernière occurrence.
nom-zone	Nom d'une zone dont la valeur avant exécution de READ ou RETRIEVE est comparée à sa valeur après exécution. Si elle a changé, la condition BEFORE/AFTER CHANGE est vérifiée. Cette zone doit faire partie d'un article spécifié par RETRIEVE ou du fichier spécifié par READ. La condition AFTER CHANGE est vérifiée pour la première occurrence et la condition BEFORE CHANGE l'est pour la dernière.
nom-variable-temporaire	Nom d'une zone de l'article lu par l'instruction READ. La valeur de cette zone avant exécution de READ est comparée à sa valeur après exécution. Si elle a changé, la condition BEFORE/AFTER CHANGE est vérifiée. La condition AFTER CHANGE est vérifiée pour la première occurrence lue et la condition BEFORE CHANGE l'est pour la dernière.

expression-sous-chaîne	Expression permettant de spécifier une portion de zone ou de variable temporaire. Cette zone doit faire partie d'un article spécifié par l'instruction READ ou RETRIEVE associée.
CHANGE	CHANGE indique que la séquence d'instructions est à exécuter si l'occurrence d'article ou la zone a changé (cf. description de l'instruction). Les modalités d'exécution dépendent du mot-clé (BEFORE ou AFTER) spécifié.
AND / OR	Opérateurs logiques utilisables pour former les expressions conditionnelles composées (cf. chapitre 2).
séquence-d'instructions	Séquence d'instructions à exécuter si la condition spécifiée est remplie. Elle peut contenir n'importe quelle instruction à l'exception de LET avec l'option BY et d'une autre instruction de contrôle.
END	Fin de la séquence d'instructions. Si une étiquette a été spécifiée dans l'instruction de contrôle, elle doit également figurer après END.
ON ABSENT	ON ABSENT indique que la séquence d'instructions est à exécuter si aucun article n'est délivré (cf. description de l'instruction).

EXEMPLE :

```
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
   WHERE U-NOM = "PARIS-1"
   AFTER C-NOM CHANGE
      DISPLAY C-NOM
   END
   LET $NUM = COUNT E-NOM BY C-NOM
   BEFORE C-NOM CHANGE
      PRINT "NOMBRE D'ETUDIANTS :" COL 7, $NUM JUST LEFT
   END
END
```

❑

Résultat :

```
C-NOM : MEDECINE
      NOMBRE D'ETUDIANTS : 3
C-NOM : SCIENCE
      NOMBRE D'ETUDIANTS : 4
```

6.10 LET

Fonction :

Affectation d'une valeur à une zone.

Format 1 :

```
 ------------------------------------------------------
|       { <nom-zone-1>             }                   |
|   LET {                          } =                 |
|       { <expression-sous-chaîne> }                   |
|                                                      |
|       { <expression> }                          }    |
|       { RATIO (<nom-zone-2>, <nom-zone-3>)      }    |
|       { PERCENTAGE (<nom-zone-2>, <nom-zone-3>) }    |
|                                                      |
|                    { "<chaîne-caractères>" }         |
|       [PIC[TURE]   {                       } ]       |
|                    {  <chaîne-caractères>  }         |
|                                                      |
|                     { LEFT   }                       |
|       [JUST[IFIED] { CENTER }   ]                    |
|                     { RIGHT  }                       |
 ------------------------------------------------------
```

Format 2 :

```
 ----------------------------------------
|      { <nom-zone-1>             }      |
|  LET {                          } =    |
|      { <expression-sous-chaîne> }      |
|                                        |
|      {COUNT  }                         |
|      {SUM    }  { <nom-zone-4> }       |
|      {MAX    }  {              }       |
|      {MIN    }  {(<nom-zone-4>)}       |
|      {AVERAGE}                         |
|                                        |
|      [BY <zone-données>]               |
 ----------------------------------------
```

Description de l'instruction :

L'instruction LET permet d'affecter une valeur à la zone spécifiée à gauche du signe égal (=). Cette valeur remplace l'ancienne valeur de la zone.

LET ne modifie que les valeurs se trouvant en mémoire ; les valeurs se trouvant dans une base de données ou dans un fichier ne sont modifiées que si LET est suivie d'une instruction MODIFY, INSERT ou WRITE qui concerne ces valeurs.

Les valeurs des zones spécifiées à droite du signe égal ne sont pas modifiées par l'exécution de LET.

Lorsque LET spécifie une valeur incorrecte pour une zone, il se produit une erreur à l'exécution. Les règles concernant la conversion lors de l'affectation de valeurs sont données au chapitre 2.

Description des paramètres :

nom-zone-1	Nom de la zone à laquelle une nouvelle valeur est affectée. Ce peut être un nom de variable temporaire, de variable permanente, de variable système modifiable ou le nom d'une zone d'article. Dans ce dernier cas, LET doit être exécutée à l'intérieur d'un bloc READ ou RETRIEVE.
expression-sous-chaîne	Le résultat est le même qu'avec un nom de zone, mais seule la portion de zone spécifiée est modifiée (cf. chapitre 2).
expression	Se reporter au chapitre 2.
RATIO	La valeur affectée est le résultat de la division de nom-zone-2 par nom-zone-3.
nom-zone-2 nom-zone-3	Ces deux noms de zones sont utilisés pour les fonctions RATIO et PERCENTAGE. Seuls les noms de zones numériques sont admis.
PERCENTAGE	La valeur affectée est égale au résultat de la division de nom-zone-2 par nom-zone-3 multiplié par 100.
PICTURE PIC	Lorsque la clause PICTURE est spécifiée, nom-zone-1 doit désigner une zone de type alphanumérique. Lorsqu'elle est omise, c'est le modèle d'édition implicite qui est utilisé (cf. chapitre 2).

JUSTIFIED JUST	Lorsque la clause JUSTIFIED est spécifiée, nom-zone-1 doit désigner une zone de type alphanumérique. Lorsqu'elle est omise, c'est le cadrage implicite qui est utilisé (cf. chapitre 2).
COUNT	Lorsque nom-zone-4 est omis, la valeur de nom-zone-1 augmente de 1 à chaque exécution de LET. Lorsque nom-zone-4 est spécifié, la valeur de nom-zone-1 augmente de 1 chaque fois que LET est exécutée et que nom-zone-4 a une valeur définie (c'est-à-dire différente d'espaces). Lorsque l'option BY n'est pas utilisée, la valeur de nom-zone-1 n'est pas remise à zéro automatiquement avant la première exécution de LET ; c'est donc l'utilisateur qui doit le faire.
SUM	A chaque exécution de LET, la valeur de nom-zone-4 est ajoutée à la valeur de nom-zone-1. Les deux zones doivent être de type numérique. Lorsque l'option BY n'est pas utilisée, la valeur de nom-zone-1 n'est pas remise à zéro automatiquement avant la première exécution de LET ; c'est donc l'utilisateur qui doit le faire.
MAX	A chaque exécution de LET, la valeur de nom-zone-1 est comparée à la valeur de nom-zone-4. Si elle lui est inférieure, nom-zone-1 prend la valeur de nom-zone-4. Sinon, nom-zone-1 conserve son ancienne valeur. Lorsque l'option BY n'est pas utilisée, la valeur de nom-zone-1 n'est pas initialisée automatiquement ; c'est donc l'utilisateur qui doit le faire (à la valeur la plus basse).
MIN	Chaque fois que LET est exécutée, la valeur de nom-zone-1 est comparée à la valeur de nom-zone-4. Si elle lui est supérieure, nom-zone-1 prend la valeur de nom-zone-4. Sinon, elle conserve son ancienne valeur. Lorsque l'option BY n'est pas utilisée, la valeur de nom-zone-1 n'est pas initialisée automatiquement ; c'est donc l'utilisateur qui doit le faire (à la valeur la plus haute).

AVERAGE	Chaque fois que LET est exécutée, la valeur de nom-zone-4 est ajoutée à un accumulateur interne, un compteur interne est incrémenté de 1 et la valeur de nom-zone-1 est forcée au résultat de la division de l'accumulateur par le compteur. L'accumulateur a le même type et le même nombre de positions décimales que nom-zone-4. Sa longueur est la longueur maximum autorisée pour ce type. Le compteur est de type BINARY 31. Chaque instruction LET comportant l'option AVERAGE dispose d'un accumulateur et d'un compteur qui sont remis à zéro au début de la requête.
BY	L'option BY ne peut être utilisée qu'à l'intérieur d'un bloc READ ou RETRIEVE spécifiant un article dont zone-données fait partie. Elle n'est pas autorisée à l'intérieur d'une instruction de contrôle. Lorsque zone-données est spécifié, il doit être précédé de BY. Au début de la boucle READ ou RETRIEVE ou lorsque la valeur de zone-données a changé depuis la dernière exécution de LET, la valeur de nom-zone-1 est automatiquement initialisée : - à zéro pour SUM et COUNT - à sa valeur minimum pour MAX - à sa valeur maximum pour MIN. Pour AVERAGE, les valeurs de l'accumulateur et du compteur sont également remises à zéro. Lorsqu'aucun article n'est délivré, la valeur de nom-zone-1 reste la même. L'option BY évite d'avoir à remettre nom-zone-1 à la valeur appropriée (zéro, minimum ou maximum) dans une instruction AFTER...CHANGE (cf. exemple 7 plus loin). L'option BY est équivalente à une instruction AFTER et provoque une rupture dans le traitement linéaire : l'instruction LET concernée n'est pas exécutée de façon linéaire avec les autres instructions.

EXEMPLE 1 (SUM) :

```
LET $VENTES = SUM($REG) BY DEPARTEMENT
❑
```

La valeur de $REG est ajoutée à celle de $VENTES à chaque exécution de LET. La valeur de $VENTES est initialisée à zéro au début de la boucle et remise à zéro chaque fois que la valeur de DEPARTEMENT change. La zone DEPARTEMENT doit faire partie d'un article en cours de traitement, c'est-à-dire que l'instruction LET doit se trouver à l'intérieur d'une boucle READ ou RETRIEVE.

EXEMPLE 2 (COUNT) :

```
LET $NBR-VENTES = COUNT($REG)
❑
```

Cette instruction compte le nombre d'occurrences valides de la zone $REG, la valeur de $NBR-VENTES augmentant de 1 à chaque occurrence. A noter que l'instruction LET est toujours exécutée lorsque nom-zone-4 est de type numérique ; s'il s'agit d'une zone de type CHARACTER, LET est exécutée uniquement si la zone a une valeur définie (différente d'espaces).

EXEMPLE 3 (MAX) :

```
LET $V-MAXI = MAX($REG)
❑
```

Cette instruction détermine la valeur maximum de la zone $REG et la place dans $V-MAXI.

EXEMPLE 4 (MIN) :

```
LET $V-MINI = MIN($REG)
```

❑

Cette instruction détermine la valeur minimum de la zone $REG et la place dans $V-MINI.

EXEMPLE 5 (EMPLOI DES PARENTHESES) :

```
LET $CALC = ($UN + $DEUX) / ($TROIS + $QUATRE) - $CINQ * $SIX
❑
```

Cette instruction produit le résultat suivant :

1. la somme de $UN et $DEUX est divisée par la somme de $TROIS et $QUATRE.
2. le résultat de la multiplication de $CINQ par $SIX est soustrait du résultat obtenu en (1) et le résultat final est placé dans $CALC.

EXEMPLE 6 (BY ET BEFORE...CHANGE) :

```
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
      WHERE U-NOM = "PARIS-1"
   LET $AGE = $YEAR -1900 -N-ANNEE
   LET $TOTAL = SUM($AGE) BY C-NOM
   LET $SOMME = COUNT(E-NOM) BY C-NOM
   BEFORE C-NOM CHANGE
      LET $MOYENNE = $TOTAL / $SOMME
      PRINT "C-NOM:", C-NOM
      PRINT "MOYENNE DES AGES:" COL 7, $MOYENNE
   END
END
❑
```

Résultat (la valeur de la variable système $YEAR étant 1982) :

```
C-NOM: MEDECINE
      MOYENNE DES AGES: 23
C-NOM: SCIENCE
      MOYENNE DES AGES: 22
```

Si la zone E-NOM est toujours présente, cette requête peut également être écrite de la manière suivante :

```
DEFINE $AGE DEC 2
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
      WHERE U-NOM = "PARIS-1"
   LET $AGE = $YEAR -1900 -N-ANNEE
   LET $MOYENNE = AVERAGE($AGE) BY C-NOM
   BEFORE C-NOM CHANGE
      PRINT "C-NOM:", C-NOM
      PRINT "MOYENNE DES AGES:" COL 7, $MOYENNE
   END
END
```

EXEMPLE 7 (BY ET AFTER...CHANGE) :

Illustration de l'équivalence entre l'option BY et l'instruction AFTER...CHANGE

```
RETRIEVE A
   LET $ZONE-1 = SUM(ZONE-2) BY ZONE-3
END
```

❑

équivaut à :

```
RETRIEVE A
   AFTER ZONE-3 CHANGE
      LET $ZONE-1 = 0
   END
   LET $ZONE-1 = SUM(ZONE-2)
END
```

REMARQUE :

Si la fonction utilisée était MAX (ou MIN) et non pas SUM, la valeur de $ZONE-1 dans la deuxième requête devrait être initialisée à la valeur la plus basse (ou la plus haute) et non à zéro.

6.11 MODIFY

Fonction :

Modification de zones ou d'articles de la base de données, à partir de données se trouvant en mémoire.

Format 1 :

```
.--------------------------.
|         { <nom-article>}  |
|  MODIFY {              }  |
|         { <nom-zone>   }  |
 --------------------------
```

Format 2 :

```
.-------------------------------------------.
|  MODIFY <nom-zone> = <expression>          |
|                                            |
|                { "<chaîne-caractères>" }    |
|    [PIC[TURE]  {                      } ]  |
|                { <chaîne-caractères>   }    |
|                                            |
|                 { LEFT   }                 |
|    [JUST[IFIED] { CENTER } ]               |
|                 { RIGHT  }                 |
 -------------------------------------------
```

Description de l'instruction :

Le premier format de l'instruction permet d'enregistrer dans la base de données les modifications effectuées en mémoire par des instructions LET et ACCEPT sur le contenu d'un article spécifié par RETRIEVE. MODIFY spécifie un nom de zone ou un nom d'article ; dans le premier cas, seule la zone est modifiée, dans le second cas la totalité de l'article est modifiée.

Le deuxième format permet de modifier directement les données sans avoir à utiliser LET.

L'instruction MODIFY, quel que soit son format, ne peut être utilisée qu'à la suite d'une instruction RETRIEVE. Pour que MODIFY soit exécutée, il faut que l'aire dans laquelle se trouve l'article à modifier ait été ouverte en mode mise à jour (UPDATE).

MODIFY peut être utilisée :

- avec une vue, pour les fichiers UFAS dont les droits d'accès permettent la mise à jour à condition que l'article à modifier fasse partie de l'un des schémas sous-jacents à la vue.
- avec un schéma, pour les fichiers UFAS et les bases de données IDS/II.

Lorsque la nouvelle valeur d'une zone ne satisfait pas au contrôle spécifié dans la définition du schéma (clause CHECK), l'instruction MODIFY n'est pas exécutée. En conséquence, si l'instruction MODIFY spécifie un article dont l'une des zones est dans ce cas, elle n'est pas exécutée.

Les clés primaires et les zones indiquant le type de l'article ne peuvent pas être modifiées. A noter que lorsqu'une clé secondaire est modifiée, son index est mis à jour.

REMARQUE :

Le contenu d'un article n'est garanti au cours d'une requête que dans le bloc RETRIEVE qui a permis de l'obtenir. Il est donc nécessaire d'effectuer la mise à jour avant de passer à l'occurrence d'article suivante.

Description des paramètres :

nom-article	Nom ou synonyme de l'article à modifier. Il doit avoir été spécifié par une instruction RETRIEVE précédente et l'aire dont il fait partie doit avoir été ouverte en mode mise à jour (UPDATE).
nom-zone	Nom d'une zone d'un article spécifié par une instruction RETRIEVE précédente et appartenant à une aire ouverte en mode mise à jour (UPDATE). Le contenu de la zone (en mémoire) est écrit dans la base de données. Les autres zones de l'article ne sont pas affectées par l'opération.
expression	Nouvelle valeur de la zone fournie directement ou obtenue en évaluant l'expression. Les expressions sont traitées au chapitre 2.
PICTURE/JUSTIFIED	Ces clauses ne peuvent être utilisées que si la zone à modifier est de type alphanumérique. Lorsqu'elles sont omises, le modèle d'édition et le cadrage implicites sont utilisés (cf. chapitre 2).

EXEMPLE 1 :

Instructions simples

```
MODIFY ETUDIANT
```

❑

Cette instruction permet d'enregistrer un article ETUDIANT dans la base de données. Elle doit se trouver à l'intérieur du bloc RETRIEVE spécifiant cet article. Si l'une des zones de l'article ETUDIANT ne satisfait pas au contrôle spécifié dans la définition du schéma (clause CHECK), l'instruction n'est pas exécutée.

```
MODIFY GRADE
```

Cette instruction permet d'enregistrer une zone GRADE dans la base de données. Elle doit se trouver à l'intérieur du bloc RETRIEVE spécifiant l'article dont fait partie la zone GRADE. Si la zone GRADE ne satisfait pas au contrôle spécifié par la clause CHECK, MODIFY n'est pas exécutée.

```
MODIFY GRADE = "MAITRISE"
```

Cette instruction donne à la zone GRADE la valeur MAITRISE et enregistre cette nouvelle valeur dans la base de données.
Elle doit se trouver à l'intérieur du bloc RETRIEVE spécifiant l'article dont fait partie la zone GRADE. Si la zone GRADE ne satisfait pas au contrôle spécifié par la clause CHECK, l'instruction MODIFY n'est pas exécutée.

EXEMPLE 2 :

```
RETRIEVE UNIVERSITE,COLLEGE,ETUDIANT
      WHERE U-NOM = "PARIS-1"
   IF GRADE = "LICENCE" AND N-ANNEE < 63
   THEN MODIFY GRADE = "MAITRISE"
      PRINT WITH TITLE E-NOM
   END
END
```

Résultat :

```
E-NOM

ARNAUD
SARRE
```

❑

6.12 PARAMETER (PARAM)

Fonction :

Définition du ou des paramètres d'une sous-requête à fournir lors de l'appel de cette dernière par une instruction EXECUTE (spécifiée dans la requête principale).

Format :

```
|-------------------------------------------------------------|
| PARAM[ETER] [<numéro-niveau>] <zone> [(<occ>)] [<attribut>]  |
|                                                             |
|  [, [<numéro-niveau>] <zone> [(<occ>)] [<attribut>]]  ...    |
|-------------------------------------------------------------|
```

Le format de <attribut> est le suivant :

```
.-------------------------------------------------------------.
|                                                             |
| 1) IDEM <nom-zone>                                          |
|                                                             |
| 2) Lorsque la zone est de type alphanumérique :             |
|                                                             |
|                    {(<longueur>) }                          |
|       CHAR[ACTER] {             }                           |
|                    {<longueur>   }                          |
|                                                             |
| 3) Lorsque la zone est de type numérique :                  |
|                                                             |
|{               {(<longueur>)}                             }|
|{      BIN[ARY] {            }                             }|
|{               { <longueur> }                             }|
|{                                                          }|
|{ {UNPACKED}  {SIGNED  }           {(<longueur>[,<décimales>])}}|
|{[{         }][{        }] DEC[IMAL]{                      }}|
|{ { PACKED }  {UNSIGNED}           { <longueur>[,<décimales>] }}|
|                                                             |
 -------------------------------------------------------------
```

Description de l'instruction :

Cette instruction définit les paramètres à fournir à la sous-requête lorsqu'elle est appelée par une instruction EXECUTE ou par une commande EXEC. Il s'agit d'une instruction non-exécutable qui doit précéder toute instruction exécutable de la requête. Lorsque plusieurs paramètres sont définis, ils doivent obligatoirement être séparés par des virgules.

Description des paramètres :

numéro-niveau	Constante entière non signée différente de zéro; les zéros à gauche sont ignorés. Les paramètres de type scalaire ou vecteur peuvent être définis avec un numéro de niveau égal à 1 ou sans numéro de niveau, auquel cas il prend la valeur implicite 1. Les noms de structure et de groupe doivent être précédés du numéro de niveau 1 ; les noms des éléments dont ils se composent doivent être précédés d'un numéro de niveau indiquant leur position dans la hiérarchie. A noter que les structures et les groupes ne doivent pas avoir d'attribut (IDEM, CHAR, BIN, DEC).
zone	Noms des paramètres.
occ	Ce paramètre est utilisé pour définir des vecteurs et des groupes. Il s'agit d'une constante entière qui indique le nombre d'occurrences d'un paramètre de type vecteur ou groupe.
nom-zone	Nom de la zone dont la description est à utiliser pour le paramètre spécifié par <zone>. Sa description doit déjà exister (dans une structure, un article de la vue courante ou une instruction DEFINE ou PARAMETER précédente). Il peut s'agir du nom d'un article, d'une structure, d'un vecteur, d'un groupe ou d'un scalaire. Lorsque le nom d'un vecteur ou d'un groupe est utilisé, le nombre d'occurrences (occ) peut être modifié. nom-zone ne peut pas être qualifié ou indicé.
CHARACTER	Ce paramètre indique que la zone est de type alphanumérique. <longueur> indique le nombre maximum de caractères que peut contenir la zone ; sa valeur maximum est 32767.

BINARY	Ce paramètre indique que la zone est de type binaire. Les deux longueurs autorisées sont 15 et 31 bits. Les valeurs possibles sont les suivantes :
	`BINARY 15  de -32768 à 32767`
	`BINARY 31  de -2**31 à (2**31) -1`

SIGNED/UNSIGNED	Ces deux paramètres s'appliquent aux zones de type DECIMAL uniquement. Ils indiquent si une position doit être prévue pour le signe lorsque la valeur est imprimée. La valeur implicite est SIGNED.
PACKED/UNPACKED	Ces deux paramètres s'appliquent aux zones de type DECIMAL uniquement. PACKED indique que la valeur doit être rangée sous forme condensée (2 chiffres décimaux par octet) et UNPACKED qu'elle doit l'être sous forme éclatée (1 chiffre décimal par octet). La valeur implicite est UNPACKED.
DECIMAL	Ce paramètre indique que la zone est de type décimal. <longueur> spécifie le nombre maximum de chiffres décimaux que peut contenir la zone ; sa valeur ne doit pas dépasser 31 et ne comprend pas le signe.
longueur	Constante numérique indiquant la longueur maximum de la valeur que peut contenir la zone. L'unité dans laquelle la longueur est exprimée dépend du type de la zone (voir CHARACTER, BINARY et DECIMAL ci-dessus).
décimales	Ce paramètre ne s'applique qu'aux zones de type DECIMAL. Il s'agit d'une constante numérique qui indique le nombre de chiffres devant figurer à droite de la marque décimale. Sa valeur implicite est zéro. Les paramètres longueur et décimales doivent être séparés par au moins un espace ou une virgule. Le nombre de chiffres après la marque décimale doit être inclus dans la longueur.

Exemple :

Les deux requêtes ci-dessous ont déjà été utilisées pour illustrer l'instruction EXECUTE ; elles contiennent également l'instruction PARAMETER. REQUETE-1 définit deux arguments qui sont transmis à REQUETE-2 en tant que paramètres. Dans cet exemple, les valeurs des paramètres sont introduites au clavier, puis fournies à REQUETE-1 après l'instruction RETURN.

REQUETE-1 :

```
LET $I = 0
DEFINE $NOMBRE UNSIGNED DECIMAL 4
DEFINE @GRADE CHARACTER 20
   EXECUTE REQUETE-2 ($NOMBRE,@GRADE)
RETRIEVE ONLY $NOMBRE ETUDIANT
         WHERE GRADE=@GRADE
   PRINT E-NOM
END
```

REQUETE-2 (sous-requête) :

```
PARAMETER $NOMBRE UNSIGNED DECIMAL 4,
          @GRADE CHARACTER 20
ACCEPT $NOMBRE PROMPT "EXAMEN DE COMBIEN D'ARTICLES?"
ACCEPT @GRADE PROMPT "QUEL GRADE?"
RETURN
```

6.13 PRINT

Fonction :

Impression d'une ligne d'état selon un format de présentation défini par l'utilisateur.

Format :

```
+-------------------------------------------------------------+
| [<étiquette>.] PRINT [FREE] [WITH TITLE]                    |
|                                                             |
|       [<descripteur-élément> [,<descripteur-élément>] ...]  |
|                                                             |
|           { PRINTER             }                           |
|       [TO {                     } ]                         |
|           { <fichier-impression> }                          |
+-------------------------------------------------------------+
```

- Le format de "descripteur-élément" est le suivant :

```
1)   { <nom-article> }   || SHORT              ||
    [{               }] [||{NCOL <nn>        } ||]
     { <nom-fichier> }   ||{COL[UMN] [+] <n> } ||

2)                 || {NCOL <nn>         } ||
                   || {COL[UMN] [+]  <n> } ||
    <expression> [ ||SHORT                 ||]
                   ||<clauses-édition>     ||
                   ||<clause-titre>        ||
```

- Le format de <clauses-édition> est le suivant :

```
||          {"<chaîne-caractères>"}||
||PIC[TURE] {                     }||
||          { <chaîne-caractères> }||
||                                  ||
||            {LEFT  }              ||
||JUST[IFIED] {CENTER}              ||
||            {RIGHT }              ||
```

- Le format de <clause-titre> est le suivant :

```
       {LEFT  }
TITLE [{CENTER}] [{<exp-alpha-1>                              }]
       {RIGHT }  {(<exp-alpha-1> [,<exp-alpha-2>]
                                      [,<exp-alpha-3>])}
```

Description de l'instruction :

L'instruction PRINT crée une ligne d'état. L'état peut être rangé dans un fichier spécifié explicitement (dont le format de sortie est SSF) ou dans le fichier de sortie implicite affecté à PRINTER.

L'option FREE permet de générer des caractères graphiques dans les zones à imprimer, ces caractères étant fonction du type de terminal utilisé. L'option FREE n'est utilisable que sous IOF et TDS.

L'option WITH TITLE permet de générer des titres de colonnes constitués implicitement des noms de zones correspondants (sans préfixe @, # ou $ s'il s'agit de variables temporaires ou système), ou explicitement d'expressions définies dans des clauses TITLE. Il n'y a pas de titres implicites pour les constantes et les expressions composées. IMPORTANT : les clauses de description d'état (voir chapitre 8) ne sont prises en compte que si l'option WITH TITLE est spécifiée.

Lorsque l'instruction PRINT spécifie plusieurs zones, celles-ci sont imprimées de gauche à droite dans l'ordre où elles sont spécifiées.

Lorsque PRINT spécifie un nom d'article ou de fichier, toutes les zones de l'article sont imprimées, de gauche à droite, dans l'ordre d'apparition.
Pour un article base de données, l'ordre est défini dans le schéma. Pour un article de fichier de travail, deux cas sont à considérer : si PRINT spécifie un nom d'article, l'ordre est celui dans lequel les zones sont citées dans l'instruction WRITE qui a défini l'article ; si PRINT spécifie un nom de fichier, l'ordre est celui dans lequel les zones sont citées dans la première instruction WRITE qui a défini le fichier.

L'instruction PRINT peut ne spécifier aucun élément. Dans ce cas, toutes les zones fournies par l'exécution du bloc READ, RETRIEVE ou CREATE auquel est associée l'instruction PRINT sont imprimées.

Lorsque PRINT est utilisée à l'intérieur d'un bloc READ <nom-aire> et que l'aire spécifiée est de type multi-article, tous les articles lus, quel que soit leur type, sont imprimés selon la structure définie dans l'instruction PRINT. Lorsque PRINT est spécifiée sans une liste de zones, les résultats sont imprévisibles.

Les clauses d'édition permettent de mettre en forme les données avant de les imprimer (cf. chapitre 2).

Lorsque la clause TITLE est utilisée, la place réservée à une zone dans l'état correspond à la plus grande des deux valeurs suivantes :

- place nécessaire à l'impression du titre correspondant, ou
- place nécessaire à l'impression de la valeur maximum de la zone (après édition selon a clause PICTURE).

La longueur de ligne est spécifiée par la variable système $PAGE-WIDTH. Dans le cas de zones numériques, lorsque la troncation provoque la perte de chiffres significatifs (à gauche de la marque décimale), le caractère * est imprimé dans la première position de la zone.

Description des paramètres :

nom-article	Nom ou synonyme de l'article fourni par l'exécution du bloc RETRIEVE ou CREATE auquel est associée l'instruction PRINT. Toutes les zones de l'article sont imprimées, dans l'ordre physique.
nom-fichier	Nom du fichier lu par la boucle READ à laquelle est associée l'instruction PRINT. Toutes les zones définies par le premier article écrit dans le fichier sont imprimées dans l'ordre d'apparition.
SHORT	Ce paramètre indique que les espaces à gauche et à droite sont à supprimer dans la zone (ou dans toutes les zones de l'article). Si la zone suivante comporte un numéro de colonne relatif n, les valeurs significatives des deux zones sont séparées par n espaces. Lorsqu'aucun numéro de colonne (absolu ou relatif) n'est spécifié, c'est la valeur implicite définie par $COLUMN-SPACING qui est utilisée.
NCOL nn	nn est le numéro relatif de la colonne définie dans la description courante du titre (dans une instruction HEADING et/ou FOOTING par exemple).
COLUMN	Ce paramètre indique que la position à laquelle doit commencer l'impression de la zone ou du littéral est déterminée par la valeur de n ci-dessous.

n / +n	n est un entier, éventuellement précédé du signe +, dont la valeur ne doit pas excéder la largeur de page du fichier. Valeur maximum 255. +n signifie que l'impression de la zone ou du littéral doit commencer n positions après l'élément précédent de la ligne. +0 signifie donc que les deux éléments doivent être accolés, +1 qu'ils doivent être séparés par un espace, etc. n sans signe + signifie que l'impression de la zone ou du littéral doit commencer n positions après le début de la ligne, qui a la valeur 1. Lorsque la valeur de n est inférieure au numéro de la prochaine position disponible de la ligne, tout ou partie des éléments précédents est recouvert. Lorsque le paramètre COLUMN n'est pas spécifié, l'espacement implicite entre colonnes est défini par la variable système $COLUMN-SPACING.
étiquette.	Nom permettant d'identifier l'instruction PRINT dans les instructions HEADING et/ou FOOTING correspondantes.(cf. HEADING/FOOTING pour plus de détails).
expression	Expression à imprimer. Ce peut être une expression numérique ou alphanumérique (cf. chapitre 2). Si une expression numérique contient des données numériques incorrectes, la valeur imprimée est ???.
TO fichier-impression	Nom du fichier séquentiel récepteur des sorties de l'instruction PRINT. La valeur implicite est PRINTER.
WITH TITLE	Cette option demande l'impression de lignes de titre (3 maximum de 30 caractères chacune) pour les colonnes de l'état. Les titres imprimés sont les titres implicites. Ces derniers peuvent être remplacés par des titres explicites au moyen de la clause TITLE. Le titre implicite pour une zone est le nom de cette zone sans préfixe @, $ ou #. Le titre implicite pour une constante ou une expression composée est entièrement à espaces.

Le titre est précédé et suivi d'une ligne créée à partir des valeurs des variables système @LINE-BORDER et @COLUMN-BORDER. A noter que les séparateurs de lignes (@LINE-BORDER) et de colonnes (@COLUMN-BORDER) ne sont imprimés que si l'option WITH TITLE est spécifiée (ou si une instruction PRINT WITH TITLE a été utilisée antérieurement pour imprimer les mêmes éléments).

Lorsque des clauses TITLE sont spécifiées pour des zones dans l'instruction, le titre implicite de ces zones est remplacé par l'expression alphanumérique figurant dans ces clauses.

Le titre apparaît en haut des colonnes sur chaque page. Sur les terminaux, le saut à la page suivante est remplacé par un effacement d'écran et le titre est visualisé chaque fois que le nombre de lignes de la page, spécifié par la variable système $PAGE-HEIGHT, est atteint. Lorsqu'aucune valeur n'a été donnée à cette variable, le titre n'est visualisé qu'une seule fois (au début).
L'instruction PRINT WITH TITLE <descripteur-élément> équivaut aux instructions :

```
HEADING E1
E1. PRINT <descripteur-élément>
```

A noter que les titres implicites ne comportent qu'une seule ligne.

FREE

Option utilisable uniquement sous IOF et TDS.
L'option FREE permet de générer des caractères graphiques dans les zones à imprimer. Ces caractères employés, par exemple, pour modifier la couleur ou définir le curseur, sont fonction du terminal.

TITLE	Cette clause permet de remplacer le titre implicite d'une zone par un titre explicite de une à trois lignes de 30 caractères maximum. En sortie, chaque expression alphanumérique est placée sur une ligne de titre différente. Dans chaque colonne, les lignes de titre sont placées dans l'espace réservé selon le cadrage spécifié. Le cadrage implicite se fait à gauche (LEFT) pour les zones alphanumériques et à droite (RIGHT) pour les zones numériques. Cette clause ne peut être utilisée que si des titres de colonnes en haut ou en bas de page ont été demandés soit par l'option WITH TITLE dans l'instruction PRINT, soit par des instructions HEADING ou FOOTING.
exp-alpha	Une à trois expressions alphanumériques à utiliser comme titre (voir ci-dessus). Lorsque plusieurs expressions sont spécifiées, elles doivent figurer entre parenthèses.
JUSTIFIED	Cette clause spécifie le cadrage à utiliser à la place du cadrage implicite (cf. chapitre 2).
PICTURE	Cette clause spécifie le modèle d'édition à utiliser à la place du modèle d'édition implicite (cf. chapitre 2).

EXEMPLE 1 :

```
PRINT E-NOM, FRAIS-SCOL, $AGE
❑
```

Résultat :

```
ARNAUD   2000 19
SERREAU  2500 21
WAGNER   1900 23
```

Les options WITH TITLE et COLUMN n'étant pas spécifiées, les colonnes ne portent pas de titre et l'espacement entre colonnes est celui défini par la variable système $COLUMN-SPACING (en l'occurrence 1). Les valeurs numériques sont cadrées à droite et les valeurs alphanumériques à gauche.

```
PRINT WITH TITLE GRADE COL 5, $MONTANT COL +3 PIC ZZ9.9,
       E-NOM JUST RIGHT
```

Résultat :

```
     GRADE          MONTANT              E-NOM

    LICENCE           1.3                ARNAUD
    MAITRISE        201.6                SERREAU
    MAITRISE         20.8                WAGNER
```

Le titre est répété en haut de chaque page de l'état. L'impression commence en colonne 5, comme le spécifie l'instruction, et les colonnes GRADE et MONTANT sont séparées par 3 espaces. L'espace réservé pour la zone E-NOM est déterminé par sa valeur maximum (20 caractères). L'espace réservé pour MONTANT est de 7 colonnes puisque la longueur du titre excède les 5 colonnes spécifiées par la clause PICTURE. Les valeurs numériques sont cadrées à droite implicitement et les valeurs de E-NOM sont cadrées à droite sur demande explicite.

EXEMPLE 2 :

```
DEFINE $AGE DECIMAL 2
PRINT "UNIVERSITE DE PARIS-1" COL 11
PRINT "-----" COL 19
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
      WHERE U-NOM = "PARIS-1"
      AND E-NOM BEGINS "S"
   LET $AGE = 80 - N-ANNEE
   PRINT E-NOM JUST RIGHT, $AGE, GRADE COL +2
END
SPACE
PRINT "-----" COL 20
❑
```

Résultat :

```
UNIVERSITE DE PARIS-1
               -----
          SERREAU 21  MAITRISE
            SARRE 18  LICENCE
               -----
```

EXEMPLE 3 :

```
RETRIEVE ETUDIANT
   PRINT E-NOM SHORT, GRADE
END
❑
```

Résultat :

```
ARNAUD LICENCE
SERREAU MAITRISE
WAGNER MAITRISE
MARTIN LICENCE
MARTY MAITRISE
SARRE LICENCE
WIART MAITRISE
```

7. Instructions du langage de requêtes IQS (Q-Z)

7.1 Introduction

Le présent chapitre fournit une description des instructions du langage de requêtes de la lettre Q à la lettre Z incluse. Pour les autres instructions se reporter aux chapitres précédents.
Une requête comprend un ensemble d'instructions et des commentaires facultatifs. Se reporter à la présentation générale du langage de requêtes fournie au début du chapitre 5.

En ce qui concerne la création de requêtes utilisables sous TDS, se reporter au guide utilisateur IQS-V4/TDS.

Pour tous renseignements complémentaires en matière de programmation, consulter le manuel d'initiation au langage de requêtes et le guide du programmeur.

7.2 READ

Fonction :

Lecture d'un fichier de travail séquentiel ou d'une base de données UFAS.

Format :

```
.-------------------------------------------------------------.
|                        { ONLY <expression-numérique> [TIMES] }  |
| [<étiquette>.] READ [ {                                     }] |
|                        { NEXT                               }  |
|                                                               |
|        { * <expression-alphanum> AS <nom-zone>   }            |
|        { <nom-fichier-travail> [AS <nom-zone> ]  }            |
|        { <nom-aire> [AS <nom-zone>]              }            |
|                                                               |
|       [WHERE <condition-sélection>]                           |
|                                                               |
|       [<séquence-d'instructions>] ...                         |
|                                                               |
|   END [<étiquette>]                                           |
 ---------------------------------------------------------------
```

Description de l'instruction :

Lorsque l'instruction READ est utilisée pour lire un fichier de travail créé au cours de la même requête et que plusieurs instructions WRITE portent sur ce fichier, toutes les descriptions d'article spécifiées par ces instructions sont accessibles à l'intérieur du bloc READ (c'est implicitement la première de ces descriptions qui est utilisée).

La longueur d'article du fichier est égale à celle de l'article le plus long écrit dans le fichier. Lorsque le fichier est spécifié par une expression alphanumérique ou qu'il n'a pas été créé au cours de la même session et qu'il n'a pas fait l'objet d'une commande USE STRUCTURE, il est nécessaire d'utiliser l'option AS spécifiant une variable temporaire préalablement définie pour servir de description de fichier.

Lorsque READ est utilisée pour lire un fichier correspondant à une aire définie dans un schéma (READ nom-aire), tous les articles de l'aire, quel que soit leur type, et toutes leurs descriptions sont accessibles à l'intérieur du bloc READ. L'accès au fichier est donc un accès de type COBOL qui peut être utilisé pour améliorer les performances, mais est déconseillé dans le cas des aires multiarticles.

Une instruction READ nom-aire ne peut être utilisée qu'avec des fichiers UFAS ou des unités de bibliothèque. Lorsqu'elle est utilisée avec un fichier séquentiel indexé, les articles sont lus dans l'ordre de la clé primaire. Lorsqu'elle est utilisée avec un fichier relatif, les articles sont lus dans l'ordre de leur position dans le fichier, la variable $RECORD-NUMBER utilisée dans une option WHERE permettant de sélectionner les articles à traiter (voir l'instruction RETRIEVE).

Un bloc READ nom-aire ne peut pas être emboîté dans un bloc RETRIEVE portant sur la même aire.

La séquence d'instructions est exécutée une fois pour chaque article lu.

Un branchement est effectué sur la première instruction suivant END si une instruction EXIT est exécutée, s'il n'y a plus d'articles dans le fichier (c'est-à-dire lorsque la fin de fichier est atteinte), si l'option NEXT a été spécifiée et que END est exécutée ou si le nombre d'articles spécifié par <expression-numérique> a été lu.

Une itération est effectuée (retour à l'instruction READ pour lecture d'un nouvel article) si une instruction REPEAT est exécutée ou si l'option NEXT n'a pas été spécifiée et que END est exécutée.

Les zones de l'article lu ne sont accessibles qu'à l'intérieur du bloc READ. Un bloc READ peut être emboîté dans un bloc RETRIEVE, CREATE ou READ, auquel cas les zones de l'article spécifié par ce dernier restent accessibles à l'intérieur du bloc READ emboîté.

Une instruction READ peut être préfixée d'une étiquette. Dans ce cas, l'instruction END correspondante doit comporter la même étiquette.

Description des paramètres :

étiquette	Nom permettant d'identifier l'instruction READ dans une instruction END, REPEAT ou EXIT ultérieure.
expression-numérique	Expression entière qui définit le nombre maximum d'exécutions de l'instruction READ.
NEXT	Lorsque NEXT est utilisé, l'instruction READ n'est pas itérative. Le positionnement en début de fichier ou d'aire est fourni par une instruction REWIND exécutée avant l'instruction READ. Lorsque l'instruction END est atteinte, le traitement continue avec la première instruction qui la suit, au lieu de reprendre au début du bloc READ.
*expression-alphanum	Cette option permet de spécifier un nom de fichier de travail par l'intermédiaire d'une expression alphanumérique, au lieu de le fournir directement. Dans ce cas, l'option AS nom-zone est obligatoire. Par contre, la variable système $RECORD-LENGTH n'est pas utilisable.
nom-fichier-travail	Nom du fichier à lire. Il peut avoir été créé au cours de la même session par une instruction WRITE ou SORT ou par une commande EXTRACT, WRITE, SORT, COPY ou MERGE ou au cours d'une autre session, auquel cas la commande USE STRUCTURE ou l'option AS est obligatoire.
AS nom-zone	Nom d'une variable temporaire préalablement définie pour servir de description de fichier. Cette variable doit être un nom de structure existante. Lorsque le fichier a été spécifié au moyen d'une expression alphanumérique ou lorsqu'il n'a pas été affecté pendant la session en cours ou par l'intermédiaire de la requête, l'option AS est obligatoire.
nom-aire	Nom de l'aire à lire. Le schéma UFAS correspondant doit avoir été spécifié par une commande SELECT et l'aire doit avoir été ouverte par une commande OPEN.

Condition-sélection	Ce paramètre spécifie les conditions que doit remplir un article pour être traité par les instructions du bloc READ. La valeur implicite est l'absence de conditions, c'est-à-dire, que tous les articles sont à traiter. condition-sélection est une expression conditionnelle (cf. chapitre 2). Toutes les zones spécifiées dans l'expression doivent faire partie de l'article lu.
Séquence-d'instructions	Cette séquence doit être exécutée pour chaque article lu. Elle ne doit pas comporter d'instructions WRITE ou SORT portant sur le fichier en cours de lecture.

EXEMPLE 1 (WRITE/READ) :

```
WRITE E-NOM, FRAIS-SCOL, @ADR TO FICH-NOUV
  .
  .
  .
READ FICH-NOUV WHERE FRAIS-SCOL>100
  .
  .
  .
END
```
❑

La séquence d'instructions spécifiée dans le bloc READ est exécutée pour chaque article du fichier dans lequel la valeur de FRAIS-SCOL est supérieure à 100. A l'intérieur du bloc READ, @ADR spécifie la zone de l'article lu et non la variable temporaire @ADR qui a été définie avant READ.

EXEMPLE 2 (WRITE/SORT/READ) :

```
RETRIEVE ETUDIANT
  WRITE E-NOM, GRADE, FRAIS-SCOL TO FICH-ETUD
END
SORT FICH-ETUD ON GRADE, E-NOM
READ FICH-ETUD WHERE FRAIS-SCOL>0
  AFTER GRADE CHANGE
    PRINT GRADE
  END
  PRINT E-NOM COL 7
END
```
❑

Résultat :

```
LICENCE
      ARNAUD
      MARTIN
      SARRE
MAITRISE
      MARTY
      SERREAU
      WAGNER
      WIART
```

EXEMPLE 3 (READ NOM-AIRE) :

```
READ IND-A
  IF TYP OF UNIVERSITE="U"
  THEN
    PRINT U-NOM
  END
  IF TYP OF COLLEGE="C"
  THEN
    PRINT C-NOM COL 7
  END
END
```

❑

Résultat :

```
PARIS-1
      MEDECINE
      PHARMACIE
      SCIENCE
TOULON-3
TOULOUSE-2
      ECONOMIE
```

EXEMPLE 4 (READ ONLY) :

```
RETRIEVE ETUDIANT
  WRITE ETUDIANT TO FICH-ETUD
END
READ ONLY 2 TIMES FICH-ETUD
   PRINT E-NOM
END
```

❑

Résultat :

```
ARNAUD
SEREAU
```

EXEMPLE 5 (READ AS) :

Supposons qu'au cours d'une session précédente le nom de tous les étudiants ait été écrit dans le fichier de travail permanent TRAV1 par la requête suivante :

```
RETRIEVE ETUDIANT
   WRITE E-NOM, GRADE TO TRAV1
END
```

❑

Le fichier peut maintenant être lu en définissant préalablement une variable temporaire qui servira de description de fichier :

```
DEFINE 01 DESCR,
       02 NOM CHARACTER 20,
       02 GRADE CHARACTER 9
READ TRAV1 AS DESCR
   DISPLAY DESCR
END
```

Il est également possible d'utiliser une structure existante (ici STR-ETUD) au moyen de l'option IDEM :

```
DEFINE DESCR IDEM STR-ETUD
READ TRAV1 AS DESCR
   DISPLAY DESCR
END
```

EXEMPLE 6 (UTILISATION DE $RECORD-LENGTH) :

```
DEFINE DESCR IDEM STR-ETUD
RETRIEVE ETUDIANT
   WRITE E-NOM, GRADE TO TRAV1
END

READ ONLY 1 AS DESCR
   PRINT $RECORD-LENGTH
END
```

❑

Résultat : 29

```
LET @F="TRAV1"
READ ONLY 1 TIMES *@F AS DESCR
   PRINT $RECORD-LENGTH
END
```

Résultat : Emission d'un message d'erreur à la compilation, car $RECORD-LENGTH n'est pas défini.

7.3 RECONNECT

Fonction :

Changement d'occurrence d'article maître pour une occurrence d'article détail IDS/II.

Format :

```
|--------------------------------------|
| RECONNECT <article-réel-1>           |
|                                      |
|       TO <article-réel-2>            |
|                                      |
|       [VIA <nom-ensemble>]           |
|--------------------------------------|
```

Description de l'instruction :

L'instruction RECONNECT supprime la liaison établie entre un article et l'un de ses articles maîtres et en crée une nouvelle avec un autre article maître par l'intermédiaire d'un ensemble.
Dans cet ensemble, article-réel-1 doit être défini comme article détail (MEMBER) et article-réel-2 comme article maître (OWNER) avec la clause INSERTION MANUAL ou AUTOMATIC.

RECONNECT ne peut être utilisée avec une vue, ni s'appliquer à un article défini avec la clause RETENTION IS FIXED.

Elle doit être exécutée dans le contexte d'une instruction RETRIEVE spécifiant article-réel-1 et article-réel-2.

Les règles d'insertion de l'article dans le nouvel ensemble sont indiquées dans l'instruction INSERT.

Description des paramètres :

Article-réel-1	Nom ou synonyme de l'article dont l'occurrence considérée doit changer de maître. Il doit avoir été défini comme article détail (MEMBER) dans l'ensemble considéré et avoir été spécifié par une précédente instruction RETRIEVE. La ou les aires dont font partie cet article et son article maître doivent avoir été ouvertes en mode mise à jour (UPDATE).
Article-réel-2	Nom ou synonyme de l'article dont l'occurrence considérée doit devenir maîtresse de celle d'article-réel-1. Il doit avoir été défini comme article maître (OWNER) de l'ensemble considéré et avoir été spécifié par une précédente instruction RETRIEVE. L'aire dont il fait partie doit avoir été ouverte en mode mise à jour (UPDATE).
nom-ensemble	Nom de l'ensemble reliant article-réel-1 et article-réel-2. <article-réel-1> peut avoir été défini dans cet ensemble avec la clause INSERTION AUTOMATIC ou MANUAL. Lorsque article-réel-1 et article-réel-2 sont reliés par plusieurs ensembles, la clause VIA nom-ensemble est obligatoire. Lorsque l'accès à article-réel-1 se fait au moyen de la clause VIA nom-ensemble, il n'est pas possible de modifier l'article courant (par exemple par DELETE ou MODIFY).

EXEMPLE :

Le professeur DUPONT (ENSEIGNANT) enseigne l'ANATOMIE (COURS) et l'un de ses étudiants est WAGNER (ETUDIANT). Si le cours d'anatomie doit être confié au professeur MALVIN et non plus au professeur DUPONT, les deux occurrences ANATOMIE et WAGNER doivent être rattachées à MALVIN :

```
RETRIEVE COURS WHERE NOM-COURS = "ANATOMIE"
   RETRIEVE ETUDIANT WHERE E-NOM = "WAGNER"
      RETRIEVE ENSEIGNANT FROM SYSTEM
         WHERE PROFESSEUR = "MALVIN"
         RECONNECT COURS TO ENSEIGNANT VIA EN-C
         RECONNECT ETUDIANT TO ENSEIGNANT VIA EN-E
      END
   END
END
```

❑

REMARQUE :

Les articles COURS et ETUDIANT sont recherchés avant l'article ENSEIGNANT. En effet, si l'occurrence MALVIN était recherchée d'abord, seules ses occurrences détail seraient disponibles ; le système ne trouverait donc pas les occurrences ANATOMIE et WAGNER.

Une fois les occurrences ANATOMIE et WAGNER obtenues, l'occurrence MALVIN doit être recherchée directement (FROM SYSTEM) puisqu'elle n'a encore aucune relation avec ANATOMIE et WAGNER.

7.4 REPEAT

Fonction :

Retour à la première instruction d'un bloc CREATE, DO, READ ou RETRIEVE.

Format :

```
|-------------------------|
|  REPEAT [<étiquette>]   |
|-------------------------|
```

Description de l'instruction :

L'instruction REPEAT détermine le nombre d'itérations du bloc. Elle renvoie à l'instruction CREATE, DO, READ ou RETRIEVE qui a lancé la boucle. REPEAT peut être utilisée à plusieurs endroits différents dans le bloc.

L'étiquette spécifiée dans REPEAT doit être identique à celle de l'instruction qui a lancé la boucle. Une instruction REPEAT sans étiquette est associée à l'instruction CREATE, DO, READ ou RETRIEVE de niveau immédiatement supérieur. L'instruction REPEAT doit se trouver à l'intérieur du bloc auquel elle s'applique.

REMARQUE :

Une instruction REPEAT ne doit pas comporter la même étiquette qu'une instruction de contrôle.

Description des paramètres :

étiquette	Lorsqu'elle est spécifiée, elle doit être identique à l'étiquette d'une instruction CREATE, DO, READ ou RETRIEVE précédente. Lorsqu'elle est omise, l'instruction REPEAT est associée à l'instruction CREATE, DO, READ ou RETRIEVE de niveau immédiatement supérieur.

EXEMPLE 1 :

```
RETRIEVE ART-A
   LET $INDEX = 0
   LET $SOMME = 0
   DO
     LET $INDEX = $INDEX +1
     LET $SOMME = $SOMME + VAL($INDEX)
     IF $INDEX = NBRE-ART
     THEN PRINT NOM-COMPTE,RAYON,$SOMME
          EXIT
     ELSE REPEAT
     END
   END
END
```

❑

L'article ART-A contient les zones NOM-COMPTE, RAYON, VAL et NBRE-ART. Cette dernière contient le nombre d'occurrences de la zone répétitive VAL.

Pour chaque article extrait, les valeurs de VAL sont ajoutées à la variable temporaire $SOMME.

L'itération du bloc est déterminée par la valeur de la variable $INDEX et l'instruction REPEAT. Celle-ci renvoie à la première instruction de la boucle DO, c'est-à-dire à LET $INDEX=$INDEX + 1.

Une fois toutes les occurrences de VAL traitées, les zones NOM-COMPTE, RAYON et $SOMME sont imprimées. L'instruction EXIT permet de sortir de la boucle DO. L'article ART-A suivant est extrait et le processus se répète.

EXEMPLE 2 :

```
RETRIEVE ETUDIANT
   LET $AGE = 80 - N-ANNEE
   IF $AGE LT 20
   THEN REPEAT
   END
   PRINT WITH TITLE $AGE, E-NOM
END
```

❑

Résultat :

```
AGE E-NOM
 21 SERREAU
 23 WAGNER
 22 MARTY
 24 WIART
```

7.5 REPORT

Fonction :

Edition d'un fichier de travail séquentiel monoarticle selon un format de présentation et/ou une description d'état déterminés.

Format :

```
 ----------------------------------------
|  REPORT <nom-fichier-travail>          |
|                                        |
|          { PRINTER                }    |
|    [  TO {                        } ]  |
|          { <nom-fichier-impression> }  |
 ----------------------------------------
```

Description de l'instruction :

L'instruction REPORT permet d'éditer le contenu d'un fichier de travail. Le fichier est édité conformément au format de présentation sélectionné par l'instruction (ou la commande IQS) USE FORMAT ON nom-fichier-travail et/ou par la description d'état sélectionnée par l'instruction (ou la commande) USE REPORT ON nom-fichier-impression.

Lorsque les valeurs données à une même clause (COL SPACING par exemple) sont différentes dans le format de présentation et dans la description d'état, c'est la valeur spécifiée dans le format qui est utilisée.

Description des paramètres :

nom-fichier-travail	Nom du fichier à éditer.
nom-fichier-impression	Nom du fichier récepteur des sorties de l'instruction REPORT. La valeur implicite est PRINTER.

7.6 RESTART

Fonction :

Annulation de toutes les modifications effectuées dans une ou plusieurs bases de données et relance de la requête au dernier point de reprise. Cette instruction n'est utilisable que sous TDS et en traitement par lots.

Format :

```
|-----------|
|  RESTART  |
|-----------|
```

Description de l'instruction :

L'instruction RESTART permet d'effectuer un restauration non actualisée de la ou les bases de données modifiées et de relancer la requête à partir du dernier point de reprise constitué.

Lorsque cette instruction est utilisée sous IOF, la variable système @STATUS est forcée à la valeur "FUNCNAV". Dans ce cas, la requête s'arrête prématurément, à moins que l'utilisateur n'ait prévu un test sur la valeur de @STATUS.

Pour plus de détails sur le concept de relance, se reporter au guide utilisateur IQS-V4/TDS (81UR).

Description des paramètres :

Néant.

7.7 RETRIEVE

Fonction :

Recherche d'articles dans des bases de données UFAS ou IDS/II pour lesquelles un schéma ou une vue ont été définis.

Format :

```
.-------------------------------------------------------------------------.
|[<étiquette>.]RETRIEVE [ONLY <expression-numérique> [TIMES]]             |
|                                                                         |
|                                        {WITHIN <nom-aire>    }          |
|                                        {                     }          |
|[<synonyme>=]<nom-article> [FROM SYSTEM][{    {<nom-ensemble>} }]        |
|                                        {VIA {              } }          |
|                                        {    {<nom-clé>     } }          |
|                                                                         |
|                                        {WITHIN <nom-aire>    }          |
|                                        {                     }          |
|[,<synonyme>=]<nom-article>[FROM SYSTEM][{    {<nom-ensemble>} }]]...    |
|                                        {VIA {              } }          |
|                                        {    {<nom-clé>     } }          |
|                                                                         |
|      [WHERE <condition-sélection>]                                      |
|                                                                         |
|      [<séquence-d'instructions>] ...                                    |
|                                                                         |
|  END [<étiquette>]                                                      |
 -------------------------------------------------------------------------
```

Description de l'instruction :

L'instruction RETRIEVE permet d'accéder aux articles d'une base de données pour constituer des articles utilisateur ; un article utilisateur se compose de l'ensemble des occurrences d'article demandées dans l'instruction.
Le bloc RETRIEVE est exécuté itérativement. A chaque itération, une nouvelle occurrence de l'article utilisateur est obtenue et la séquence d'instructions éventuellement spécifiée est exécutée sur cette occurrence ; une fois la dernière occurrence obtenue, le contrôle passe à l'instruction qui suit END. Entre RETRIEVE et END, toutes les zones des articles spécifiés sont accessibles.
L'instruction RETRIEVE peut comporter une étiquette, auquel cas l'instruction END correspondante doit avoir la même. Les instructions EXIT et REPEAT peuvent être utilisées à l'intérieur d'une instruction RETRIEVE.

Description des paramètres :

étiquette	Nom permettant d'identifier l'instruction RETRIEVE à laquelle se rapporte une instruction END, EXIT ou REPEAT ultérieure. A noter que l'étiquette doit être suivie d'un point dans l'instruction RETRIEVE et pas dans l'instruction END.
expression-numérique	Expression numérique entière (cf. chapitre 2) qui indique le nombre maximum d'exécutions de l'instruction RETRIEVE.
synonyme =	Chaîne de 30 caractères maximum qui permet de distinguer les différentes occurrences d'un même article lorsqu'il apparaît plusieurs fois dans un article utilisateur ou dans un ensemble d'instructions RETRIEVE emboîtées.
nom-article	Nom d'un article de la base de données dont des occurrences doivent être recherchées.
FROM SYSTEM	Cette option est utilisée pour rechercher un article sans tenir compte des autres articles spécifiés avant lui. Lorsqu'elle n'est pas utilisée, la recherche s'effectue à partir de tous les articles spécifiés avant lui dans la même instruction ou dans les autres instructions RETRIEVE dans lesquelles celle-ci est emboîtée.
nom-aire	Nom de l'aire de la base de données dans laquelle se trouve l'article à rechercher.
nom-ensemble	Nom d'un ensemble de la base de données dont l'article à rechercher est maître ou détail. Lorsqu'il existe plusieurs chemins d'accès à un article, la clause VIA nom-ensemble est obligatoire afin de définir un chemin unique. La clause VIA nom-ensemble ne peut être utilisée pour le premier article spécifié (article d'entrée) que si l'instruction RETRIEVE est emboîtée dans d'autres instructions RETRIEVE.
nom-clé	Nom d'une clé de la base de données. Il est utilisé avec une base de données UFAS séquentielle indexée ou une base de données IDSII -clé secondaire - pour indiquer que l'ordre des articles à rechercher est fourni par cette clé (sinon, l'ordre est défini par la clé UFAS primaire, ou la clé CALC IDSII). Remarque : une condition peut générer un accès direct par clé, équivalent à "via nom-clé" implicite.

condition-sélection	Cette option permet de définir la condition que doit remplir l'article utilisateur à rechercher. Les articles ne la remplissant pas ne sont pas traités. Cette condition peut être exprimée sous forme d'une expression conditionnelle composée (cf. chapitre 2). Toutes les zones spécifiées dans la condition doivent faire partie de l'article utilisateur à rechercher.
séquence-d'instructions	Cette séquence est exécutée pour chaque article fourni par RETRIEVE. Elle peut contenir n'importe quelle instruction du langage de requêtes, y compris d'autres instructions RETRIEVE.

7.7.1 Accès aux bases de données par l'intermédiaire d'une vue

Une vue est une description monoarticle ou multiarticle arborescente (avec certaines relations entre les différents types d'articles). Elle est définie à partir de fichiers UFAS (séquentiel ou séquentiel indexé) et de bases de données IDS/II.

L'accès aux bases de données contenant un seul type d'article s'effectue de la manière suivante :

```
RETRIEVE <nom-article-logique>
```

une occurrence de l'article spécifié étant fournie à chaque itération.

Pour une base de données arborescente, il est possible d'accéder à plusieurs articles selon leurs relations définies dans la vue. Le premier article spécifié dans l'instruction doit toujours être l'article racine de la vue.

```
RETRIEVE <nom-article-logique-1>, <nom-article-logique-2>
```

signifie que l'article logique 2 dépend de l'article logique 1. Pour chaque occurrence de l'article logique 1, le système accède aux occurrences de l'article logique 2 correspondantes ; à chaque itération, il fournit un agrégat composé d'une occurrence de l'article 1 et d'une occurrence de l'article 2 et ce, jusqu'à ce qu'il ait atteint la fin de la base de données.

Les mêmes règles s'appliquent lorsque plus de deux articles logiques sont spécifiés, chacun d'entre eux devant dépendre du précédent.

Les articles ne remplissant pas la condition spécifiée ne sont pas retenus. Une occurrence d'article utilisateur n'est fournie que si elle est complète, c'est-à-dire composée d'une occurrence de chacun des articles spécifiés.
Pour obtenir les occurrences de l'article logique 1 même si aucune occurrence de l'article logique 2 ne leur correspond, l'article utilisateur doit être divisé en deux parties recherchées par des instructions RETRIEVE emboîtées :

```
RETRIEVE <nom-article-logique-1>
   RETRIEVE <nom-article-logique-2>
      .
      .
      .
   END
END
```

Pour chaque occurrence de l'article logique 1, une boucle de recherche est effectuée sur l'article logique 2. Les occurrences de l'article 1 sont fournies par la boucle extérieure même si la boucle intérieure ne fournit aucune occurrence de l'article 2.

Les aires contenant les articles spécifiés dans RETRIEVE doivent avoir été ouvertes par une commande OPEN pour que l'instruction puisse être exécutée.

7.7.2 Accès à une base de données séquentielle par l'intermédiaire d'un schéma

Les bases de données séquentielles peuvent avoir deux types de structure :

- non hiérarchique (bases mono ou multiarticles sans relations entre articles).
- arborescente (bases multiarticles avec relations entre articles).

L'accès à un seul type d'article, que la base ait une structure non hiérarchique ou arborescente, se fait de la manière suivante :

```
RETRIEVE <article-réel>
```

A chaque itération, le système fournit une occurrence de l'article spécifié en lisant la base de données dans l'ordre physique des articles (cf. exemple 2 plus loin).

L'accès à plusieurs types d'articles dans une base arborescente s'effectue en fonction de leur ordre dans la base et des relations définies dans le schéma (cf. Exemple 1).

Dans l'instruction :

```
RETRIEVE <article-réel-1>, <article-réel-2>
```

article-réel-2 peut ou non dépendre de article-réel-1. S'il en dépend, pour chaque occurrence de article-réel-1, le système accède à toutes les occurrences de l'article-réel-2 situées physiquement entre cette occurrence et l'occurrence suivante de article-réel-1. Il fournit ainsi des agrégats composés d'une occurrence de article-réel-1 et d'une occurrence de article-réel-2 jusqu'à ce qu'il ait atteint la fin de la base de données (cf. exemple 3).

Lorsque, pour une occurrence de article-réel-1, il n'existe aucune occurrence de article-réel-2, le système ne fournit rien.

Si article-réel-2 ne dépend pas de article-réel-1, le système accède, pour chaque occurrence de article-réel-1, à toutes les occurrences de article-réel-2 du début à la fin du fichier (produit cartésien). Il en est de même lorsque article-réel-2 est spécifié dans une sous-requête. Exemple :

```
RETRIEVE <article-réel-1>
   PRINT <article-réel-1>
   EXEC REQ-2
END
```

la sous-requête REQ-2 étant la suivante :

```
RETRIEVE <article-réel-2>
   PRINT <article-réel-2>
END
```

L'occurrence courante de l'article sélectionné dans la requête appelante n'étant pas transmise à la sous-requête (REQ-2), le système imprime, pour chaque occurrence de article-réel-1, **toutes** les occurrences de article-réel-2.

A noter qu'un article réel peut dépendre de n'importe quel article spécifié avant lui. Il peut également ne dépendre d'aucun article, auquel cas il est implicitement "FROM SYSTEM".

L'accès est toujours séquentiel, qu'une condition soit spécifiée ou non. Les articles ne remplissant pas une condition spécifiée ne sont pas retenus. Une occurrence d'article utilisateur n'est fournie que si elle est complète ; pour reprendre l'exemple précédent, elle n'est fournie que s'il existe au moins une occurrence de article-réel-2 pour une occurrence de article-réel-1.

Pour obtenir les occurrences de article-réel-1 même si aucune occurrence de article-réel-2 ne leur correspond, l'article utilisateur doit être divisé en deux parties recherchées par des instructions RETRIEVE emboîtées :

```
RETRIEVE <article-réel-1>
   RETRIEVE <article-réel-2>
        .
        .
        .
   END
END
```

Pour chaque occurrence de article-réel-1, une boucle de recherche est effectuée sur article-réel-2. Les occurrences de article-réel-1 sont fournies par la boucle extérieure même si la boucle intérieure ne fournit aucune occurrence de article-réel-2 (cf. exemple 4).

En général, l'article spécifié dans l'instruction RETRIEVE intérieure dépend du dernier article réel spécifié par l'instruction RETRIEVE extérieure. Les occurrences fournies par la boucle intérieure sont donc celles situées entre deux occurrences consécutives de l'article spécifié par l'instruction extérieure. Cependant, lorsque l'article intérieur ne dépend pas d'un article extérieur, pour chaque occurrence de l'article extérieur, le système accède à toutes les occurrences de l'article intérieur du début à la fin du fichier. Il convient donc de s'assurer que les articles spécifiés dans des instructions RETRIEVE emboîtées sont en relation.

De manière plus générale, un article utilisateur intérieur peut dépendre de n'importe quel article utilisateur extérieur.

REMARQUE :
Dans un fichier séquentiel, il n'est pas possible d'obtenir une occurrence d'article maître à partir d'une occurrence d'article détail.

Par exemple, avec l'instruction :

```
RETRIEVE <article-réel-2>, <article-réel-1>
```

le système fournit, pour chaque occurrence de article-réel-2, toutes les occurrences de article-réel-1, c'est-à-dire le produit cartésien de article-réel-2 par article-réel-1. De même, lorsque l'option FROM SYSTEM est utilisée pour un article qui est détail d'un article précédent, le système fournit le produit cartésien des deux articles réels.

En principe, lorsque des articles sont recherchés dans un fichier séquentiel, la totalité du fichier est lue même si certains articles ne remplissent pas la condition spécifiée. Cependant, il est possible d'optimiser le traitement lorsque le premier article réel de l'article utilisateur est déclaré trié et que sa clé (ou la première partie de sa clé) est utilisée dans la condition avec l'un des opérateurs =, <, <=, >, >=, BETWEEN ou BEGINS. L'itération peut ainsi être arrêtée au premier article ne remplissant pas la condition, en fonction des facteurs suivants :

- ordre de tri (ASC ou DSC)
- opérateur utilisé
- emplacement de la clé dans la condition (à gauche ou à droite de l'opérateur).

Les différents cas possibles sont indiqués dans le tableau ci-dessous, O signifiant que l'itération s'arrête et N qu'elle continue.

Tableau 7-1. Arrêt de l'itération (optimisation)

ORDRE DE TRI	EMPLACEMENT	>	>=	BETWEEN BEGINS	=	<	<=
DSC	GAUCHE	O	O	N	O	N	N
	DROITE	N	N	N	O	O	O
ASC	GAUCHE	N	N	O	O	O	O
	DROITE	O	O	N	O	N	N

Cette optimisation (arrêt de l'itération) évite une lecture du fichier jusqu'à la fin lorsqu'aucun article ne peut plus être fourni.

A noter que, dans le cas d'une instruction :

```
RETRIEVE A, B WHERE <condition-a> ...
```

lorsque la <condition-a> ne dépend pas de l'article B, elle est évaluée juste après la lecture de l'article A. La lecture boucle sur les occurrences de A tant que la condition n'est pas remplie.

Compte tenu de l'optimisation décrite plus haut, les deux requêtes ci-dessous sont équivalentes. Les articles du fichier séquentiel sont classés en ordre croissant (ASC) sur la zone REFERENCE :

```
(1)     ACCEPT $LIMITE
        RETRIEVE PIECE WHERE REFERENCE < $LIMITE
           PRINT
        END

(2)     ACCEPT $LIMITE
        RETRIEVE PIECE
           IF REFERENCE < $LIMITE
           THEN PRINT
           ELSE EXIT
           END
        END
```

7.7.3 Accès à une base de données séquentielle indexée par l'intermédiaire d'un schéma

Les bases de données séquentielles indexées peuvent avoir les deux mêmes types de structure que les bases séquentielles :

- non hiérarchique
- arborescente

Tout ce qui a été dit pour les bases séquentielles vaut également pour les bases séquentielles indexées. La seule différence est que la condition de sélection (clause WHERE) permet l'accès direct au premier article réel de l'article utilisateur sous réserve que les règles suivantes soient respectées :

- Opérateurs utilisables : =,<=,<,>=,>,BETWEEN,BEGINS

  L'ordre de recherche est le suivant :

  – clé complète avec =

  – clé partielle avec un opérateur parmi : =,>=,>,BETWEEN,BEGINS
  L'arrêt de l'itération est possible avec =,<=,<,BETWEEN et/ou BEGINS.

- Les zones constituant la clé d'un article réel indépendant d'un article spécifié avant lui, ou utilisé avec l'option FROM SYSTEM, figurent dans la partie gauche des conditions simples.
- Toute expression de même type (alphanumérique ou numérique par exemple) est spécifiée dans la partie droite des conditions qui définissent la clé.

  – si les règles ci-dessus sont respectées mais que le contenu des parties gauches et des parties droites a été interverti, le système tente d'inverser la condition pour trouver une clé sauf si l'opérateur est BEGINS.

- Seuls les opérateurs AND sont traités. Ainsi, dans une condition composée telle que C1 AND (C2 OR (C3 OR C4)), le système ne prendra en compte que C1.
- Avec les clés constituées d'une seule zone, les seuls opérateurs relationnels pris en compte sont >,>=, = et BEGINS.
- Avec les clés constituées de plusieurs zones, les seuls opérateurs pris en compte sont :

  – l'opérateur = utilisé sur toutes les zones de la clé, ou

  – les opérateurs =, >, >=, BETWEEN et BEGINS pour la première zone de la clé (pour les autres zones, n'importe quel opérateur peut être utilisé).

Avec la clause VIA, l'accès direct au premier article réel n'est obtenu que si la clé figurant dans la condition de sélection de la clause WHERE correspond à l'index (nom-de-clé) spécifié par VIA. Dans le cas contraire, l'accès est séquentiel sur l'index indiqué par VIA.

A noter que dans la mesure où les règles énoncées ci-dessus sont respectées, il est également possible d'utiliser des zones autres que des zones-clés dans les conditions de sélection.

L'optimisation par arrêt de l'itération dès qu'un article ne remplit pas la condition de sélection n'est possible que pour une zone avec l'opérateur =, <, < =, BETWEEN ou BEGINS.

Si l'une de ces règles n'est pas respectée, l'accès est séquentiel (cf. exemple 6).

Dans le cas où article-réel-2 dépend d'article-réel-1, l'instruction :

```
RETRIEVE <article-réel-2>, <article-réel-1>
```

donne l'accès direct à article-réel-1 en fonction de la valeur de clé de l'occurrence d'article-réel-2. Il n'y a alors qu'une et une seule occurrence d'article-réel-1 pour chaque occurrence d'article-réel-2 (cf. exemple 6).

7.7.4 Accès à une base de données relative par l'intermédiaire d'un schéma

Les bases de données relatives UFAS peuvent uniquement avoir une structure hiérarchique (bases mono ou multiarticles sans relations entre articles).

Tout ce qui a été dit sur le produit cartésien pour les bases de données séquentielles vaut également pour les bases de données relatives dans la mesure où cela concerne des requêtes principales et appelantes.

La variable système $RECORD-NUMBER permet l'accès direct à un article réel lorsque la condition de sélection (clause WHERE) comporte l'une des options suivantes :

```
=  >  >=  BETWEEN
```

L'arrêt de l'itération est possible avec :

```
=  <  <=  BETWEEN
```

L'optimisation par arrêt de l'itération est la même que pour les bases de données indexées. Les bases de données relatives ont une clé relative unique rangée dans une zone numérique non signée.

La variable système $RECORD-NUMBER peut uniquement recevoir un entier positif. Si une valeur négative ou nulle est fournie, IQS accède à la première occurrence de l'article.

7.7.5 Accès à une base de données intégrée (IDS/II) par l'intermédiaire d'un schéma

Les informations qui suivent impliquent une connaissance préalable des bases de données de type CODASYL et des particularités des fichiers IDS/II.

Une base de données intégrée présente une structure en réseau dans laquelle les liaisons définies entre les types d'articles déterminent des **ensembles**. On a vu précédemment que les structures hiérarchiques étaient caractérisées par des relations article maître-article détail. La différence entre le modèle hiérarchique et le modèle réseau réside dans le fait que dans le premier un type d'article ne peut avoir qu'un seul maître alors que dans le second, il peut en avoir plusieurs. Le modèle réseau est un surensemble du modèle hiérarchique. En d'autres termes, tout ce qui a été dit pour la base de données séquentielle indexée est également valable pour la base de données intégrée :

- un article utilisateur peut être constitué d'un seul article réel.
- un article utilisateur peut être constitué de plusieurs articles réels liés ou non entre eux par des relations maître-détail.
- l'accès direct au premier article réel de l'article utilisateur est possible si les critères de sélection contiennent une clé CALC (calculée) satisfaisant aux mêmes règles que celles énoncées plus haut pour les bases de données séquentielles indexées ; la seule différence est que l'accès direct n'est pas obtenu si les opérateurs relationnels sont >, >=, BETWEEN ou BEGINS, du fait que les clés ne sont pas triées.
- l'optimisation du traitement par arrêt de l'itération dès qu'un article ne remplit pas la condition de sélection est possible si l'ensemble est trié.

En outre, les possibilités offertes par la structure en réseau permettent de demander des articles utilisateur IDS/II plus complexes dans RETRIEVE.

Pour constituer un article utilisateur à partir d'articles réels d'une structure en réseau, le processus est le suivant : partant d'un article réel, appelé article d'entrée, un chemin doit être défini à travers les ensembles de la base de données, en procédant d'article maître à article détail ou inversement d'article détail à article maître. L'article utilisateur est un agrégat d'occurrences de chaque article réel du chemin ainsi défini et d'occurrences d'articles réels obtenues en procédant de détail à maître pour chaque article réel de l'article utilisateur.

Cette description du chemin à suivre ne met pas en évidence le mécanisme de la clause VIA qui va maintenant être analysée avec ses règles d'emploi.

Avant de poursuivre, il convient toutefois de définir la terminologie utilisée :

- article d'entrée — il s'agit du premier article réel d'un article utilisateur ; il permet d'entrer dans le réseau.

- sous-article utilisateur — Il se compose des articles réels du chemin, obtenus en passant de détail à maître, et des articles réels ne faisant pas partie du chemin, également obtenus en passant de détail à maître. Par conséquent, dès que le chemin va de maître à détail ou que l'article réel obtenu est indépendant de tous les précédents (que ce soit au moyen de la clause FROM SYSTEM ou autrement), un nouveau sous-article utilisateur commence.

- article suivant — Il s'agit du prochain article réel à prendre en compte. S'il est maître de l'un des articles réels précédents, il est incorporé dans le sous-article utilisateur.
S'il en est détail, il devient le premier article réel d'un nouveau sous-article utilisateur.

Les règles de création d'un article utilisateur, développées ci-après, se classent en cinq catégories :

1. Article d'entrée : traitement du premier article réel du premier sous-article utilisateur.
2. Extension d'un sous-article-utilisateur : adjonction d'articles réels à un sous article utilisateur.
3. Début d'un nouveau sous-article-utilisateur : passage à un nouveau sous-article utilisateur.
4. Fin de chemin : fin de traitement du dernier sous-article utilisateur.
5. Synonymes : traitement des noms d'articles réels apparaissant plusieurs fois dans un même article utilisateur ou dans deux articles utilisateur d'un emboîtement.

1. Article d'entrée

1.1 L'article d'entrée devient le premier article réel du premier sous-article utilisateur.

1.2 Lorsque l'article d'entrée figure dans plusieurs aires, la clause WITHIN permet d'en sélectionner une. Si WITHIN n'est pas utilisée, l'article d'entrée est prélevé dans toutes les aires où il figure.

1.3 Lorsqu'il existe plusieurs liaisons (détail à maître ou maître à détail) entre l'article d'entrée et l'un des articles réels de l'instruction RETRIEVE de niveau immédiatement supérieur, la clause VIA est obligatoire pour indiquer l'ensemble par lequel doit s'effectuer l'accès à l'article d'entrée, sauf si la clause FROM SYSTEM est utilisée pour spécifier qu'aucune de ces liaisons n'est à prendre en compte.

A noter que lorsqu'il n'existe qu'une seule liaison, la clause VIA n'est pas nécessaire et la clause FROM SYSTEM ne l'est que si cette liaison ne doit pas être utilisée.

2. Extension d'un sous-article utilisateur

2.1 L'extension d'un sous-article utilisateur se fait en choisissant comme article suivant l'article maître de l'un des articles réels déjà spécifiés dans le sous -article utilisateur.

2.2 Lorsqu'il existe plusieurs liaisons (détail à maître ou maître à détail) entre l'article suivant et l'un des articles réels déjà spécifiés dans le sous-article utilisateur, la clause VIA est obligatoire pour indiquer l'ensemble par lequel doit s'effectuer l'accès à l'article suivant. (A noter que, conformément à la règle 2.1, l'article suivant doit être maître de l'ensemble et que l'un des articles réels du sous-article utilisateur doit en être détail).

2.3 L'article suivant est incorporé dans le sous-article utilisateur.

2.4 Cette opération peut se répéter autant de fois que nécessaire.

3. Début d'un nouveau sous-article utilisateur

3.1 La création d'un nouveau sous-article utilisateur se fait en choisissant comme article suivant soit un article détail de l'un des articles réels de l'article utilisateur, soit un article réel n'ayant aucune liaison avec ceux de l'article utilisateur, soit un article réel auquel l'accès se fait indépendamment de ses liaisons au moyen de la clause FROM SYSTEM.

3.2 S'il existe plusieurs liaisons (détail à maître ou maître à détail) entre l'article suivant et l'un des articles réels de l'article utilisateur, la clause VIA peut être utilisée pour indiquer l'ensemble par lequel doit s'effectuer l'accès à l'article suivant. Si la clause VIA n'est pas utilisée, l'ensemble pris en compte est celui reliant l'article suivant à l'article réel le plus proche. (A noter que, conformément à la règle 3.1, dans l'ensemble considéré, l'article suivant doit être détail et l'un des articles réels de l'article utilisateur doit être maître).

3.3 Le sous-article utilisateur précédent est incorporé à l'article utilisateur.

3.4 L'article suivant devient le premier article réel d'un nouveau sous-article utilisateur.

3.5 L'extension de ce nouveau sous-article utilisateur peut s'effectuer conformément à la règle 2.

4. Fin de chemin

Lorsqu'il n'y a plus besoin d'ajouter de nouveaux articles réels au chemin, le dernier sous-article utilisateur est incorporé à l'article utilisateur.

5. Synonymes

Lorsqu'un article réel figure plusieurs fois dans le même article utilisateur ou qu'il figure dans un article utilisateur et dans un autre article utilisateur de niveau supérieur (instructions RETRIEVE emboîtées), des synonymes doivent être attribués à chacune des occurrences, sauf une, de cet article afin que leurs zones puissent être identifiées. Chaque fois que le nom d'une zone de cet article réel est employé, il doit être qualifié (par le synonyme ou le nom d'article réel). A noter qu'il est possible d'attribuer des synonymes à toutes les occurrences de l'article si nécessaire.

L'article utilisateur est spécifié dans l'instruction RETRIEVE en donnant les noms des articles réels dans l'ordre où ils doivent figurer conformément aux règles énoncées ci-dessus.

Les clauses VIA et WITHIN doivent être spécifiées immédiatement après le nom de l'article réel auquel elles s'appliquent.

EXEMPLE (D'APRES LA BASE DE DONNEES IDS/II DECRITE A L'ANNEXE A) :

L'article utilisateur à créer est le suivant :

```
ENSEIGNANT, DEPARTEMENT, COLLEGE VIA C-EN, ETUDIANT VIA C-E,
         HORAIRE, COURS, CENS=ENSEIGNANT.
```

❑

La méthode de création de l'article utilisateur peut se résumer comme suit :

sous-article utilisateur 1

```
ENSEIGNANT              Règle 1.1
DEPARTEMENT             Règles 2.1, 2.3
COLLEGE VIA C-EN        Règles 2.4, 2.1, 2.2, 2.3
```

sous-article utilisateur 2

```
ETUDIANT VIA C-E        Règles 3.1, 3.2, 3.3, 3.4
```

sous-article utilisateur 3

```
HORAIRE                 Règles 3.1, 3.3, 3.4
COURS                   Règles 3.5, 2.1, 2.3
CENS=ENSEIGNANT         Règles 2.4, 2.1, 2.3, 4, 5
```

La description qui suit analyse le traitement de chaque article réel :

ENSEIGNANT : il s'agit de l'article d'entrée ; c'est par cet article que commence le premier sous-article utilisateur (règle 1.1). Ne sont nécessaires ni la clause WITHIN (règle 1.2), ni la clause VIA (règle 1.3).

DEPARTEMENT : cet article réel est sélectionné (règle 2.1) et incorporé dans le sous-article utilisateur (règle 2.3). La clause VIA n'est pas nécessaire (règle 2.2).

COLLEGE VIA C-EN : cet article réel est sélectionné (règles 2.4 et 2.1). La clause VIA est obligatoire (règle 2.2) pour indiquer que l'accès à COLLEGE se fait par l'intermédiaire de l'ensemble C-EN et non de l'ensemble C-D. Il est incorporé dans le sous-article utilisateur (règle 2.3).

ETUDIANT VIA C-E : cet article réel est sélectionné (règle 3.1). La clause VIA est nécessaire (règle 3.2) pour indiquer que l'accès à ETUDIANT se fait par l'intermédiaire de l'ensemble C-E et non de l'ensemble EN-E. Le premier sous-article utilisateur (ENSEIGNANT, DEPARTEMENT, COLLEGE) est incorporé dans l'article utilisateur (règle 3.3). ETUDIANT est le premier article réel d'un nouveau sous-article utilisateur (règle 3.4).

HORAIRE : cet article réel est sélectionné (règle 3.1). Le deuxième sous-article utilisateur (composé uniquement de ETUDIANT) est incorporé dans l'article utilisateur (règle 3.3) et un nouveau sous-article utilisateur commence (règle 3.4).

COURS : cet article réel est sélectionné (règles 3.5 et 2.1) et incorporé dans le sous-article utilisateur (règle 2.3).

CENS=ENSEIGNANT : cet article réel est sélectionné (règles 2.4 et 2.1) et incorporé dans le sous-article utilisateur (règle 2.3). Le chemin ne comportant plus d'articles réels, le dernier sous-article utilisateur est incorporé dans l'article utilisateur (règle 4). L'article ENSEIGNANT étant spécifié deux fois dans l'article utilisateur, il doit avoir un synonyme au moins l'une des deux fois (règle 5).

Lorsque l'instruction RETRIEVE est exécutée, le système commence par rechercher la première occurrence de ENSEIGNANT ; de là il passe aux articles maîtres DEPARTEMENT et COLLEGE, à la première occurrence de ETUDIANT (pour COLLEGE), à la première occurrence de HORAIRE (pour ETUDIANT) puis à son article maître COURS et enfin à l'article maître ENSEIGNANT. Tout ceci constitue la première occurrence de l'article utilisateur. A noter que les deux occurrences de ENSEIGNANT sont probablement différentes.

Ensuite, il cherche l'occurrence suivante de HORAIRE avec son article maître COURS et l'article maître ENSEIGNANT et répète ce processus jusqu'à ce qu'il n'y ait plus d'occurrences de HORAIRE pour la première occurrence de ETUDIANT. Il passe alors à l'occurrence suivante de ETUDIANT et cherche les occurrences de HORAIRE, COURS et ENSEIGNANT correspondantes.

Lorsqu'il n'y a plus d'étudiants, il recherche l'occurrence suivante de l'article d'entrée ENSEIGNANT avec les articles maîtres DEPARTEMENT et COLLEGE ; le processus continue, toutes les occurrences de HORAIRE étant fournies pour chaque occurrence de ETUDIANT, et toutes les occurrences de ETUDIANT pour chaque occurrence de COLLEGE, jusqu'à ce qu'il n'y ait plus d'occurrences de l'article d'entrée ENSEIGNANT.

7.7.6 Instructions RETRIEVE emboîtées

Lorsque des instructions RETRIEVE sont emboîtées, deux cas se présentent :

- L'un des articles réels de l'article utilisateur intérieur est relié par un ensemble à l'un des articles réels d'un article utilisateur de niveau supérieur : les occurrences de l'article utilisateur intérieur dépendent des occurrences de l'article utilisateur de niveau supérieur.
- L'article d'entrée de l'article utilisateur intérieur n'est pas relié par un ensemble à l'un des articles réels d'un article utilisateur de niveau supérieur ou bien l'option FROM SYSTEM a été utilisée dans l'article utilisateur intérieur : les occurrences de l'article utilisateur intérieur sont recherchées indépendamment de l'article utilisateur extérieur.

7.7.7 Exemples d'instructions RETRIEVE

EXEMPLE 1 :

Les différents articles utilisateur pouvant être obtenus par l'intermédiaire des liaisons hiérarchiques définies dans le schéma séquentiel de l'annexe A sont les suivants :

```
UNIVERSITE
COLLEGE
DEPARTEMENT
ETUDIANT
UNIVERSITE, COLLEGE
UNIVERSITE, COLLEGE, DEPARTEMENT
UNIVERSITE, COLLEGE, ETUDIANT
COLLEGE, DEPARTEMENT
COLLEGE, ETUDIANT
```

❑

Par ailleurs, il est également possible d'obtenir d'autres articles utilisateur par le produit cartésien.

EXEMPLE 2 :

Accès séquentiel à un type d'article dans une base séquentielle ou séquentielle indexée

```
RETRIEVE UNIVERSITE
   PRINT U-NOM
END
PRINT "FIN DE L'EXECUTION"
```

❑

Résultat :

```
PARIS-1
TOULON-3
TOULOUSE-2
FIN DE L'EXECUTION
```

Chaque occurrence de l'article UNIVERSITE est lue dans l'ordre physique de la base et l'instruction PRINT U-NOM est exécutée pour chacune d'entre elles. Lorsque toutes les occurrences ont été fournies, le contrôle passe à la première instruction suivant END, c'est-à-dire à PRINT "FIN DE L'EXECUTION".

EXEMPLE 3 :

Accès séquentiel à deux types d'articles dans une base de données séquentielle ou séquentielle indexée

```
RETRIEVE UNIVERSITE, COLLEGE
   PRINT U-NOM, C-NOM
END
```

❑

Résultat :

```
PARIS-1              MEDECINE
PARIS-1              PHARMACIE
PARIS-1              SCIENCE
TOULOUSE-2           ECONOMIE
```

L'ordre des articles dans la base de données (cf. annexe A, schéma séquentiel) est le suivant :

```
U PARIS-1
C MEDECINE
C PHARMACIE
C SCIENCE
U TOULON-3
U TOULOUSE-2
C ECONOMIE
```

U et C sont les valeurs des zones TYP qui indiquent le type de l'article, à savoir UNIVERSITE et COLLEGE. Dans le schéma, l'article COLLEGE est relié à l'article UNIVERSITE.

Le système fournit une première occurrence de l'article utilisateur constituée de la première occurrence de UNIVERSITE et de la première occurrence de COLLEGE : PARIS-1/MEDECINE, puis il passe aux deuxième et troisième occurrences de COLLEGE et fournit PARIS-1/PHARMACIE et PARIS-1/SCIENCE. Comme il n'y a plus d'occurrences de COLLEGE pour la première occurrence de UNIVERSITE, il passe à l'occurrence suivante de UNIVERSITE, TOULON-3. Ne trouvant pas d'occurrences de COLLEGE pour celle-ci, il lit l'occurrence suivante de UNIVERSITE et la première occurrence de COLLEGE correspondante et fournit TOULOUSE-2/ECONOMIE.

EXEMPLE 4 :

Instructions RETRIEVE emboîtées

```
RETRIEVE UNIVERSITE, COLLEGE
   PRINT C-NOM
   RETRIEVE ETUDIANT
      PRINT E-NOM COL 4
   END
END
```

❑

Dans le schéma, l'article ETUDIANT dépend de COLLEGE. Le système accède à tous les étudiants de chaque collège de chaque université ; si un collège n'a pas d'étudiants, seule la première instruction PRINT est exécutée.

Résultat :

```
MEDECINE
   ARNAUD
   SERREAU
   WAGNER
PHARMACIE
SCIENCE
   MARTIN
   MARTY
   SARRE
   WIART
ECONOMIE
```

Si la requête suivante avait été utilisée (une seule instruction RETRIEVE) :

```
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
   PRINT C-NOM, E-NOM COL 11
END
```

le résultat aurait été différent :

```
MEDECINE          ARNAUD
MEDECINE          SERREAU
MEDECINE          WAGNER
SCIENCE           MARTIN
SCIENCE           MARTY
SCIENCE           SARRE
SCIENCE           WIART
```

Aucune instruction ne peut être insérée entre COLLEGE et ETUDIANT. Lorsqu'un collège n'a pas d'étudiants, l'instruction PRINT n'est pas exécutée ; c'est le cas pour les collèges PHARMACIE et ECONOMIE dont le nom est imprimé par la première requête mais pas par la seconde.

EXEMPLE 5 :

Principaux articles utilisateur pouvant être créés à partir du schéma séquentiel indexé de l'annexe A

```
UNIVERSITE
COLLEGE
ETUDIANT
UNIVERSITE, COLLEGE
UNIVERSITE, COLLEGE, ETUDIANT
COLLEGE, ETUDIANT
ETUDIANT, COLLEGE
COLLEGE, UNIVERSITE
ETUDIANT, COLLEGE, UNIVERSITE
```

❑

EXEMPLE 6 :

Accès direct à une base de données séquentielle indexée (cf. annexe A)

```
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT WHERE
      U-NOM="PARIS-1" AND E-NOM="MARTIN"
  PRINT SEC-SOCIALE
END
```

❑

Résultat :

```
140 26 1344
```

U-NOM ayant été défini dans le schéma comme clé de l'article UNIVERSITE, le système peut accéder directement à l'article correspondant au nom spécifié (PARIS-1) ; ensuite, il lit séquentiellement les articles COLLEGE et ETUDIANT en recherchant ceux qui remplissent la deuxième condition.

Avec la requête suivante, l'accès est direct puisque SEC-SOCIALE est une clé secondaire :

```
RETRIEVE ETUDIANT, COLLEGE, UNIVERSITE
      WHERE SEC-SOCIALE="140 26 1344"
  PRINT E-NOM, U-NOM
END
```

Résultat :

```
MARTIN         PARIS-1
```

Lorsque des valeurs identiques sont admises pour une clé secondaire, il y a itération et le système fournit toutes les occurrences remplissant la condition spécifiée.

EXEMPLE 7 :

Création de divers articles utilisateur à partir du schéma IDS/II de l'annexe A

```
COLLEGE, ETUDIANT
❑
```

Ces deux articles sont reliés par l'ensemble C-E de maître à détail et l'accès se fait par l'intermédiaire de cet ensemble.

```
COLLEGE, UNIVERSITE, ETUDIANT, HORAIRE
```

Le chemin COLLEGE, ETUDIANT, HORAIRE est choisi ; l'article UNIVERSITE vient en complément de COLLEGE. Pour chaque occurrence de COLLEGE, le système accède à l'occurrence de UNIVERSITE qui en est maître par l'intermédiaire de l'ensemble U-C.

Pour chaque occurrence de COLLEGE, le système lit toutes les occurrences de ETUDIANT correspondantes par l'intermédiaire de l'ensemble C-E puis, pour chacune d'entre elles, toutes les occurrences de HORAIRE correspondantes par l'intermédiaire de l'ensemble E-HR.

```
UNIVERSITE, COLLEGE, ENSEIGNANT, COURS
```

Pour aller de UNIVERSITE à COURS, le chemin COLLEGE, ENSEIGNANT est choisi. Tous les articles doivent être spécifiés pour constituer l'article utilisateur.

```
COURS, HORAIRE, ETUDIANT, COLLEGE, DEPARTEMENT
```

Le chemin va de COURS à DEPARTEMENT en passant par HORAIRE, ETUDIANT et COLLEGE. Tous les articles doivent être spécifiés.

```
ENSEIGNANT, COLLEGE, ETUDIANT VIA EN-E
```

La clause VIA est obligatoire puisque l'accès à ETUDIANT peut se faire par l'intermédiaire de deux ensembles : C-E ou EN-E. Dans ce cas, l'accès à ETUDIANT se fait par l'intermédiaire de EN-E, COLLEGE est donc un article qui vient en complément de ENSEIGNANT. Le chemin aurait également pu être direct (de ENSEIGNANT à ETUDIANT en passant par COLLEGE) auquel cas l'accès à ETUDIANT aurait dû se faire par l'intermédiaire de C-E.

EXEMPLE 8 :

Accès séquentiel à la base de données IDS/II de l'annexe A

```
RETRIEVE COLLEGE, ETUDIANT
   PRINT C-NOM, E-NOM COL 12
END
```
❑

Résultat :

```
MEDECINE          ARNAUD
MEDECINE          SERREAU
MEDECINE          WAGNER
SCIENCE           MARTIN
SCIENCE           MARTY
SCIENCE           SARRE
SCIENCE           WIART
```

COLLEGE et ETUDIANT sont reliés dans le sens maître-détail par l'ensemble C-E. Pour chaque occurrence de COLLEGE, le système fournit toutes les occurrences de ETUDIANT qui en sont détail d'après l'ensemble C-E ; s'il n'y en a pas (ce qui est le cas pour PHARMACIE par exemple), il passe à l'occurrence suivante de COLLEGE.

EXEMPLE 9 :

Accès direct et séquentiel à la base de données IDS/II de l'annexe A

```
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
      WHERE U-NOM = "PARIS-1"
      AND E-NOM = "SARRE"
   PRINT SEC-SOCIALE
END
```
❑

U-NOM est une clé de type CALC. L'accès à UNIVERSITE est direct ; l'accès à COLLEGE et ETUDIANT se fait par l'intermédiaire des ensembles U-C et C-E. La condition E-NOM = "SARRE" permet d'éliminer toutes les occurrences de ETUDIANT dans lesquelles elle n'est pas remplie.

```
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
      WHERE C-NOM = "SCIENCE"
      AND E-NOM = "SARRE"
   PRINT SEC-SOCIALE
END
```

L'accès est séquentiel puisqu'aucune condition ne porte sur UNIVERSITE. L'article ETUDIANT dépend de COLLEGE qui dépend lui-même de UNIVERSITE.

```
RETRIEVE UNIVERSITE, ETUDIANT
      WHERE U-NOM = "PARIS-1"
      AND SEC-SOCIALE = "140 26 1344"
   PRINT E-NOM
END
```

L'accès à UNIVERSITE est direct (au moyen de la clé U-NOM) et l'accès à ETUDIANT est également direct par la clé CALC SEC-SOCIALE puisque ETUDIANT est indépendant de UNIVERSITE.

EXEMPLE 10 :

Instructions RETRIEVE emboîtées portant sur la base IDS/II de l'annexe A

```
RETRIEVE UNIVERSITE, COLLEGE, ENSEIGNANT
   RETRIEVE COURS
   END
END
```

❑

L'article COURS dépend de ENSEIGNANT ; l'accès à COURS se fait donc par l'intermédiaire de l'ensemble EN-C. Si l'instruction avait été RETRIEVE COURS FROM SYSTEM, l'article COURS aurait été indépendant et le système aurait lu tous les cours de tous les enseignants de tous les collèges de toutes les universités.

EXEMPLE 11 :

Utilisation de synonymes avec le schéma séquentiel indexé de l'annexe A

Ce schéma définit la zone SEC-SOCIALE comme étant une clé secondaire de l'article ETUDIANT.

Supposons qu'il faille accéder à un article ETUDIANT au moyen de cette clé secondaire, puis obtenir une liste de tous les étudiants faisant partie du même collège.

L'instruction :

```
RETRIEVE ETUDIANT, COLLEGE WHERE SEC-SOCIALE=...
```

❑

permet d'obtenir l'étudiant dont le numéro de sécurité sociale est spécifié et le collège correspondant. Cependant, pour créer l'article utilisateur complet (ETUDIANT, COLLEGE, ETUDIANT), il faut spécifier deux fois ETUDIANT.

Le système sait qu'il doit rechercher la première occurrence de ETUDIANT au moyen de la clé secondaire et la seconde par l'intermédiaire de l'ensemble C-E, mais lorsque des zones de ETUDIANT sont citées dans les instructions qui suivent, il lui est impossible de savoir à laquelle des deux occurrences de ETUDIANT elles se réfèrent.

L'utilisation de synonymes supprime cette ambiguïté en permettant de qualifier les noms de zone. Dans l'instruction :

```
RETRIEVE ETUDIANT, COLLEGE, ETUD=ETUDIANT
  WHERE SEC-SOCIALE OF ETUDIANT = ...
```

l'ambiguïté du nom N-ANNEE (par exemple) est supprimée en le qualifiant : N-ANNEE OF ETUDIANT fait référence à la zone de l'article obtenu au moyen de la clé secondaire SEC-SOCIALE et N-ANNEE OF ETUD à la zone de l'article obtenu par l'intermédiaire de l'ensemble C-E.

Le synonyme peut également être attribué à la première occurrence :

```
RETRIEVE PREM-ET=ETUDIANT, COLLEGE, ETUDIANT
  WHERE SEC-SOCIALE OF PREM-ET = ...
```

ou même aux deux occurrences comme dans la requête suivante qui permet d'obtenir la liste des étudiants du même collège moins âgés que celui dont le numéro de sécurité sociale est demandé :

```
ACCEPT @NUMERO
SPACE
RETRIEVE PREM-ET=ETUDIANT, COLLEGE, ETUD=ETUDIANT
  WHERE SEC-SOCIALE OF PREM-ET=@NUMERO
   IF N-ANNEE OF ETUD > N-ANNEE OF PREM-ET
   THEN PRINT E-NOM OF ETUD
   END
END
```

Résultat (les données introduites par l'utilisateur sont soulignées) :

```
NUMERO  : 243 25 1313

MARTIN
SARRE
```

Cette requête recherche l'étudiant dont le numéro de sécurité sociale est fourni et l'article COLLEGE correspondant. Elle recherche ensuite, dans ce collège, toutes les occurrences de l'article ETUDIANT et imprime le nom des étudiants plus jeunes que celui dont le numéro a été donné.

Si les synonymes n'avaient pas été utilisés, les noms de zones SEC-SOCIALE, N-ANNEE et E-NOM auraient été ambigus puisque le système n'aurait pas pu savoir à quelle occurrence de l'article ETUDIANT ils faisaient référence.

La requête suivante, dans laquelle un seul synonyme est utilisé, produit le même résultat que la requête précédente :

```
ACCEPT @NUMERO
SPACE
RETRIEVE PREM-ET=ETUDIANT, COLLEGE, ETUDIANT
      WHERE SEC-SOCIALE OF PREM-ET=@NUMERO
   IF N-ANNEE OF ETUDIANT > N-ANNEE OF PREM-ET
   THEN PRINT E-NOM OF ETUDIANT
   END
END
```

A noter que le synonyme (=) renomme l'article auquel il est attribué. A l'intérieur de l'instruction RETRIEVE, cet article ne sera connu que sous ce nouveau nom. Cependant, les noms des zones de l'article ne sont pas modifiés ; par conséquent, ils restent ambigus s'ils ne sont pas qualifiés (OF).

EXEMPLE 12 :

Utilisation de RETRIEVE ONLY ... TIMES

La requête suivante permet de rechercher uniquement 3 articles :

```
RETRIEVE ONLY 3 TIMES ETUDIANT
   PRINT E-NOM
END
```

❑

Résultat :

```
ARNAUD
SERREAU
WAGNER
```

7.8 RETURN

Fonction :

Sortie d'une requête.

Format :

```
RETURN
```

Description de l'instruction :

Si la requête a été appelée par une autre requête au moyen d'une instruction EXECUTE, RETURN provoque le retour à la requête appelante. Sinon, RETURN renvoie au niveau commande.

A noter qu'une instruction RETURN implicite est exécutée à la fin de chaque requête.

RETURN ne peut pas être spécifiée dans une instruction BEFORE...CHANGE ou AFTER...CHANGE.

7.9 REWIND

Fonction :

Positionnement sur le premier article d'un fichier de travail séquentiel ou d'une aire d'une base de données UFAS pour lecture par une instruction READ NEXT.

Format :

```
 ------------------------------------------
|          { *<expression-alphanumérique> }  |
|  REWIND  { <nom-fichier>                }  |
|          { <nom-aire>                   }  |
 ------------------------------------------
```

Description de l'instruction :

L'instruction REWIND permet à la prochaine instruction READ NEXT portant sur le même fichier ou sur la même aire de lire le premier article de ce fichier ou de cette aire.

Une instruction REWIND nom-aire ne peut être exécutée que pour des fichiers UFAS ou des unités de bibliothèque.

Cette instruction ne doit pas être emboîtée dans un bloc READ spécifiant le même fichier ou la même aire.

Description des paramètres :

*expression-alphanumérique	Cette option permet de spécifier un nom de fichier de travail par l'intermédiaire d'une expression alphanumérique, au lieu de le fournir directement.
nom-fichier	Nom du fichier de travail dont le premier article doit être rendu accessible. Ce fichier peut avoir été créé par une instruction WRITE ou SORT ou par une commande EXTRACT, WRITE, SORT, COPY ou MERGE dans la même session ou décrit par une commande USE STRUCTURE.
nom-aire	Nom de l'aire dont le premier article doit être rendu accessible. Le schéma UFAS correspondant doit avoir été sélectionné par une commande SELECT et l'aire ouverte par une commande OPEN.

7.10 ROLLBACK

Fonction :

Annulation de toutes les mises à jour effectuées depuis la dernière consolidation (voir instruction COMMIT).
Utilisable uniquement sous IOF.

Format :

```
ROLLBACK
```

Description de l'instruction :

L'instruction ROLLBACK annule toutes les mises à jour effectuées sur une base de données depuis la précédente instruction COMMIT.

Elle peut être utilisée pour les fichiers UFAS et IDS/II gérés par GAC (Gestion générale des accès).

Elle est interdite à l'intérieur des blocs READ, RETRIEVE et CREATE.

7.11 SORT

Fonction :

Tri d'un fichier de travail séquentiel ou d'une base de données séquentielle ou séquentielle indexée et rangement du contenu trié dans un fichier UFAS séquentiel.

Format :

```
 ------------------------------------------------------------------------
|    {                                                          }  |
|    {                                          { <fichier-sortie> } }  |
|    { <fichier-entrée> [AS <nom-var-temp>]  [TO {                  }]}  |
|    {                                          {*<exp-alphanum>   } }  |
|    {                                                          }  |
|    {                                          { <fichier-sortie> } }  |
|SORT{*<exp-alphanum-1> AS <nom-var-temp>    TO  {                  } }  |
|    {                                          {*<exp-alphanum-2> } }  |
|    {                                                          }  |
|    {                                          { <fichier-sortie> } }  |
|    { <nom-aire>                                TO {                  } }  |
|    {                                          {*<exp-alphanum>   } }  |
|    {                                                          }  |
|                    [{ASC}]                {ASC}                    |
|       ON <clé-tri> [{   }] [ ,<clé-tri> [{   }] ]...               |
|                    [{DSC}]                [{DSC}                    |
|                                                                     |
 ---------------------------------------------------------------------
```

Description de l'instruction :

L'instruction SORT permet de trier une aire ou un fichier UFAS séquentiel sur la ou les clés spécifiées en ordre croissant ou décroissant. Lorsque le fichier d'entrée est un fichier de travail et qu'il est spécifié par l'intermédiaire d'une expression alphanumérique ou qu'aucune structure ne lui est associée (fichier créé au cours d'une session précédente et n'ayant pas fait l'objet d'une commande USE STRUCTURE), l'option AS est obligatoire.

Description des paramètres :

fichier-entrée	Nom du fichier de travail à trier (fichier d'entrée). Il peut avoir été créé au cours de la même session par une instruction WRITE ou SORT ou par une commande EXTRACT, COPY, MERGE ou WRITE ou au cours d'une autre session, auquel cas la commande USE STRUCTURE ou l'option AS (cf. ci-dessous) est obligatoire.
AS nom-var-temp	Nom d'une variable temporaire préalablement définie pour servir de description de fichier. Cette variable peut être un nom de structure. L'option AS est **obligatoire** lorsque le fichier a été spécifié au moyen d'une expression alphanumérique ou qu'aucune structure ne lui est associée (fichier créé au cours d'une session précédente et n'ayant pas fait l'objet d'une commande USE STRUCTURE).
fichier-sortie	Nom du fichier cible (fichier de sortie). Lorsqu'il est omis, la valeur implicite est <fichier-entrée>. Il est obligatoire si le nom du fichier d'entrée n'a pas été fourni directement.
*exp-alphanum	Cette option permet de spécifier un nom de fichier par l'intermédiaire d'une expression alphanumérique, au lieu de le fournir directement.
nom-aire	Nom de l'aire à trier. Le schéma correspondant doit avoir été sélectionné par une commande SELECT, et l'aire doit avoir été ouverte par une commande OPEN.
clé-tri	Noms des zones à utiliser comme clés de tri. Ces zones doivent faire partie de l'article de fichier-entrée ou de aire. Pour un fichier comportant des articles de longueur variable, les zones utilisées comme clés de tri doivent faire partie de l'article le plus court afin d'éviter les risques d'arrêt prématuré. Il est possible de spécifier jusqu'à 64 clés dans l'ordre de priorité voulu (de gauche à droite).
ASC	Tri en ordre croissant (de la valeur la plus basse à la valeur la plus haute de la clé). C'est l'ordre implicite.
DSC	Tri en ordre décroissant (de la valeur la plus haute à la valeur la plus basse de la clé).

EXEMPLE :

L'instruction SORT trie le fichier FICH-ETUD. La clé majeure est N-ANNEE et la clé mineure E-NOM. Le tri s'effectue en ordre décroissant sur la clé majeure et en ordre croissant sur la clé mineure.

```
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
      WHERE U-NOM = "PARIS-1"
   WRITE N-ANNEE, E-NOM TO FICH-ETUD
END

SORT FICH-ETUD ON N-ANNEE DSC, E-NOM

READ FICH-ETUD
   PRINT WITH TITLE
   BEFORE N-ANNEE CHANGE
      SPACE
   END
END
```

❑

Cette requête produit le résultat suivant (en supposant que la base de données contienne d'autres articles que ceux indiqués à l'annexe A) :

```
N-ANNEE E-NOM

     63 MAILLET
     63 MARTIN
     63 MASSON

     62 SARRE

     61 ARNAUD

     59 DABAN
     59 DOUCET
     59 SERREAU
     59 WEIL

     58 FEVRE
     58 MARTY

     57 HENRY
     57 ROBIN
     57 WAGNER

     56 DANTEC
     56 WIART
```

7.12 SPACE

Fonction :

Commande de l'interlignage dans une page d'état ou saut en haut d'une nouvelle page.

Format :

```
 ---------------------------------------
|           { 1           }             |
|  SPACE  [ { <nnn>       } ]           |
|           { TOP [SHORT] }             |
|                                       |
|         [ TO <fichier-impression> ]   |
 ---------------------------------------
```

Description de l'instruction :

L'instruction SPACE permet de commander l'interlignage dans une page, le nombre de lignes à sauter étant spécifié par nnn (valeur implicite 1).
Elle permet également de changer de page, auquel cas c'est l'option TOP qui doit être utilisée.

Avec l'option SHORT, lorsqu'une page n'est pas remplie, les titres en bas des colonnes ou la dernière ligne de séparation horizontale ne sont pas imprimés en bas de page mais juste après la dernière ligne détail. Si l'état comporte un bloc de bas de page (clause PAGE FOOTING), son emplacement n'est pas modifié.

Les variables système $LINE et $PAGE sont automatiquement mises à jour lorsque SPACE est exécutée.

Description des paramètres :

nnn	nnn est un entier positif qui indique le nombre de lignes à sauter. Sa valeur maximum est 255 et sa valeur implicite 1 (une ligne blanche). La valeur de la variable système $LINE progresse de nnn, celle de $PAGE ne change pas. Lorsque le nombre de lignes de la page est dépassé, le résultat est indéterminé.
TOP	Ce mot-clé provoque un saut en haut de la page suivante avant impression de la ligne détail suivante. La valeur de la variable système $LINE est remise à zéro et celle de $PAGE progresse de 1.
SHORT	Avant de changer de page, les titres en bas des colonnes (FOOTING) ou la dernière ligne de séparation horizontale sont imprimés juste après la dernière ligne détail et non tout en bas de la page. Ainsi, lorsque la page n'est pas remplie, l'espace resté vierge figure à la suite de la dernière ligne imprimée. Si l'état comporte un bloc de bas de page (clause PAGE FOOTING), son emplacement n'est pas modifié.
fichier-impression	Nom du fichier de sortie. La valeur implicite est PRINTER (cf. instruction PRINT).

EXEMPLES :

```
SPACE
SPACE 1
❑
```

Ces deux instructions sont équivalentes et provoquent un saut d'une ligne (sortie d'une ligne blanche).

7.13 USE FORMAT

Fonction :

Association d'un format de présentation à un fichier de travail.

Format :

```
|-----------------------------------------------------|
| USE FORMAT <nom-format> ON <nom-fichier-travail>    |
|                                                     |
|      [, <nom-format> ON <nom-fichier-travail> ] ... |
|-----------------------------------------------------|
```

Description de l'instruction :

L'instruction USE FORMAT permet d'associer un format de présentation préalablement défini à un fichier de travail. Elle peut être utilisée plusieurs fois dans une même requête.

Description des paramètres :

nom-format	Nom du format à utiliser lors de l'édition du fichier de travail au moyen d'une instruction REPORT. Il doit s'agir d'un format existant dans la bibliothèque BINLIB.
nom-fichier-travail	Nom du fichier à éditer selon le format spécifié.

7.14 USE REPORT (USE)

Fonction :
Association d'une description d'état à un fichier de sortie.

Format :

```
 ------------------------------------------------------
|  USE [REPORT] <nom-état>                             |
|                                                      |
|          { PRINTER                 }                 |
|      [ON {                         }]                |
|          { <nom-fichier-impression> }                |
|                                                      |
|                     { PRINTER                 }      |
|   [,<nom-état> [ON {                          }] ]...|
|                     { <nom-fichier-impression> }     |
 ------------------------------------------------------
```

Description de l'instruction :

Cette instruction permet d'associer des descriptions d'état à des fichiers de sortie. Elle s'applique chaque fois que la requête est exécutée.

Elle peut être utilisée plusieurs fois dans une même requête.

Lorsqu'une description d'état est associée à un fichier dans une sous-requête (par USE REPORT), elle peut en être dissociée au moyen d'une instruction CANCEL REPORT spécifiée dans une autre sous-requête ou dans la requête appelante.

REMARQUE :
En cas d'utilisation d'un format créé sous le nouveau processeur de formats IQS-V4 et comportant des clauses de description d'état, l'emploi de USE REPORT est inutile. En effet, ces clauses sont automatiquement appliquées lorsque le format correspondant est associé au fichier.

Description des paramètres :

nom-état	Nom d'une description d'état compilée et rangée dans la bibliothèque binaire (BINLIB).
nom-fichier-impression	Nom du fichier séquentiel récepteur des sorties de l'instruction PRINT. Chaque fichier est rempli suivant la description d'état qui lui est associée.

7.15 WRITE

Fonction :

Ecriture d'articles dans un fichier de travail UFAS séquentiel.

Format :

```
.-----------------------------------------.
| WRITE [APPEND] [<nom-article>  AS ]     |
|                                         |
|       [<nom-zone> [,<nom-zone>]...]     |
|                                         |
|          {*<expression-alphanumérique>} |
|       TO {                            } |
|          {<nom-fichier-travail>       } |
|                                         |
|       [SHORT]                           |
|                                         |
 -----------------------------------------
```

Description de l'instruction :

Cette instruction permet d'écrire une ou plusieurs occurrences d'article dans un fichier de travail séquentiel.

Le fichier créé peut ensuite être lu (instruction READ) ou trié (instruction SORT) dans la même requête. Pour que le fichier de sortie soit permanent (à savoir traitable par un programme écrit dans un autre langage par exemple), il doit avoir fait l'objet d'une commande ASSIGN avant exécution de la requête qui le crée, ou d'une instruction ASSIGN.

Lorsqu'un schéma est défini pour le fichier (avant ou après sa création), ce dernier peut être traité dans d'autres requêtes par des instructions READ ou RETRIEVE. Une structure ayant fait l'objet d'une commande SAVE STRUCTURE peut également être utilisée pour le fichier.

Les zones sont écrites dans l'ordre où elles sont spécifiées dans l'instruction WRITE. Il peut s'agir de n'importe quelle zone déjà définie (cf. instruction DEFINE) à l'exception des variables système. Sont admises dans WRITE, les variables temporaires, les zones d'articles de la base de données (fournies par l'instruction CREATE ou RETRIEVE correspondante) et les zones d'un autre fichier séquentiel (fournies par l'instruction READ correspondante). Il est également possible de spécifier un nom d'article ou de fichier ce qui revient à spécifier toutes les zones de l'article ; dans ce cas, elles sont écrites dans l'ordre où elles apparaissent physiquement dans l'article. Les zones indicées ne peuvent être spécifiées explicitement dans WRITE (il faut dans ce cas passer par l'intermédiaire de variables temporaires).

L'instruction WRITE peut être utilisée sans spécifier de liste des zones à écrire. Dans ce cas, toutes les zones fournies par la précédente instruction READ ou RETRIEVE ou créées par la précédente instruction CREATE sont écrites dans le fichier dans l'ordre dans lequel elles sont fournies.

Lorsque plusieurs instructions WRITE portent sur le même fichier, il est préférable qu'elles spécifient la même liste de zones ou une liste de zones présentant les mêmes définitions afin d'éviter toute ambiguïté dans la description de l'article.

Cependant, si les listes de zones (et par conséquent les descriptions d'article) sont différentes, toutes les descriptions d'article sont accessibles aux instructions READ et SORT ultérieures. La longueur d'article maximum est celle de l'article le plus long écrit dans le fichier.

La longueur d'un article doit être inférieure ou égale à la longueur d'article déclarée pour un fichier permanent.

Lorsque l'option SHORT est spécifiée, la longueur de l'article est évaluée avant suppression des espaces à droite de la dernière zone.

Des exemples de programmation comportant des instructions WRITE sont fournies dans le guide du programmeur (79UR).

Description des paramètres :

APPEND	Cette option permet d'écrire à la fin d'un fichier déjà créé.
nom-article AS	Nom logique donné à l'article défini par l'instruction WRITE. Il peut ensuite être utilisé dans la séquence d'instructions spécifiée dans un bloc READ portant sur ce fichier. Si aucune zone n'est spécifiée, nom-article est le nom logique de toutes les zones d'article rendues disponibles par la précédente instruction CREATE, READ ou RETRIEVE.
nom-zone	Nom d'une zone à écrire dans le fichier. Ce peut être un nom de variable temporaire, le nom d'une zone d'un article de la base de données (fourni par l'instruction RETRIEVE ou CREATE précédente) ou le nom d'une zone d'un autre fichier séquentiel (fourni par l'instruction READ correspondante). Ce peut également être un nom d'article ou de fichier, ce qui équivaut à spécifier toutes les zones de l'article. Les zones indicées ne peuvent être directement écrites dans le fichier. Il faut passer par l'intermédiaire d'une variable temporaire non indicée.
*expression-alphanumérique	Cette option permet de spécifier un nom de fichier par l'intermédiaire d'une expression alphanumérique, au lieu de le fournir directement.
nom-fichier-travail	Nom du fichier séquentiel à créer. Pour pouvoir lire ensuite ce fichier, il faut spécifier ce nom dans une instruction READ.
SHORT	Cette option permet de supprimer les espaces à droite dans la dernière zone de la liste lors de l'écriture d'articles de longueur variable. Les espaces à droite des autres zones sont conservés.

EXEMPLE 1 :

```
WRITE A, B, C, @X, $Y TO NOUV-FICH-1
```

❑

Les zones A, B et C (qui font partie d'articles spécifiés par une instruction READ ou RETRIEVE précédente) et les variables temporaires @X et $Y sont écrites dans le fichier séquentiel NOUV-FICH-1. Pour lire ensuite ce fichier dans la même requête, il faut utiliser l'instruction :

```
READ NOUV-FICH-1.
```

EXEMPLE 2 :

```
WRITE TO NOUV-FICH-2
❑
```

Toutes les zones de l'article spécifié par la précédente instruction READ, CREATE ou RETRIEVE sont écrites dans le fichier NOUV-FICH-2.

EXEMPLE 3 :

```
WRITE ART-A TO NOUV-FICH-3
❑
```

Toutes les zones de l'article ART-A (spécifié par une précédente instruction READ ou RETRIEVE) sont écrites dans le fichier NOUV-FICH-3.

EXEMPLE 4 :

```
RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
      WHERE U-NOM = "PARIS-1"
   WRITE TO FICH-ETUD
END
❑
```

```
SORT FICH-ETUD ON E-NOM
READ FICH-ETUD
   IF GRADE = "LICENCE"
   THEN
      WRITE E-NOM, SEC-SOCIALE, C-NOM TO FICH-LICENCE
   ELSE
      WRITE E-NOM, SEC-SOCIALE, C-NOM TO FICH-MAITRISE
   END
END
```

8. Générateur d'états IQS

8.1 Généralités

Le générateur d'états IQS permet la création et l'impression d'états simples ou complexes. Ces états peuvent être dirigés sur l'imprimante ligne, un terminal ou dans un fichier séquentiel.

Les états simples peuvent être créés directement à partir d'une requête en affectant des valeurs aux variables système appropriées (voir plus loin). Pour les états plus complexes, une **description d'état** constituée de clauses spécifiques doit être créée au cours d'une session IQS sous IOF ou en traitement par lots.
Le processus est similaire à celui utilisé pour les requêtes avec un espace de travail pour le stockage temporaire, une bibliothèque pour le stockage permanent et les commandes de l'éditeur de textes. Avant d'être exploitable, une description d'état doit avoir été compilée et sauvegardée.

A noter qu'une même requête peut produire plusieurs états au moyen d'instructions PRINT ... TO nom-fichier, mais il ne peut être créé à la fois qu'un seul état par fichier.

Les descriptions d'état peuvent également être créées, compilées et sauvegardées en mode menu ou en mode ligne/guidage sous le nouveau processeur d'états IQS-V4. Ce processeur offre des fonctions pratiquement identiques à celles du générateur d'états, mais peut être appelé directement à partir du nouveau processeur de formats IQS-V4 : l'utilisateur peut alors définir des clauses de description d'état à associer au format et les sauvegarder en même temps que ce dernier. Les processeurs d'état et de formats IQS-V4 sont décrits dans le volume 1.

Le présent chapitre traite uniquement du générateur d'états procédural.

REMARQUE :

Le nombre de lignes par page et le nombre de caractères par ligne sont limités à 255.

8.2 Etats simples

8.2.1 Variables système commandant la présentation des états

Les états simples peuvent être créés directement par une requête, sans qu'il soit nécessaire de définir une description d'état. Les instructions utilisables sont PRINT, HEADING, FOOTING, et éventuellement AFTER et BEFORE (voir chapitres 5 à 7), la présentation de l'état étant alors déterminée par l'affectation de valeurs aux variables système énumérées au tableau 8-1 ci-après (voir également chapitre 3).

Tableau 8-1. Variables système commandant la présentation des états

Variable système	Description	Clause de description d'état équivalente
@COLUMN-BORDER	Caractère séparateur de colonnes. Valeur implicite : espace.	COLUMN BORDER
$COLUMN-SPACING	Espacement entre colonnes, sous la forme d'un entier. Valeur implicite : 1.	COLUMN SPACING
@LINE-BORDER	Caractère séparateur de lignes (séparation des en-têtes de colonnes et des lignes détail). Valeur implicite : espace	LINE BORDER
$MAX-PAGES	Nombre maximum de pages, sous la forme d'un entier. Valeur implicite : nombre illimité.	NUMBER OF PAGES
$PAGE	Numéro de la page courante .	Pas d'équivalent direct. STARTING PAGE NUMBER fixe une valeur initiale.
$PAGE-HEIGHT	Nombre maximum de lignes par page, sous la forme d'un entier. Valeur implicite : 56.	LINES de PAGE LIMIT
$PAGE-WIDTH	Nombre de positions dans une ligne, sous la forme d'un entier. Valeur implicite déterminée à l'exécution en fonction de l'appareil utilisé.	PAGE WIDTH

Pour plus de détails sur ces fonctions, se reporter aux clauses de description d'état équivalentes, analysées en fin de chapitre.

Dans le cas où la clause "TO nom-fichier" figure dans plusieurs instructions PRINT (moyen de produire plusieurs états au cours d'une même requête), la variable système doit être qualifiée par "OF nom-fichier" pour indiquer l'état auquel elle se rapporte. Si elle n'est pas qualifiée, elle est considérée comme se rapportant à l'état destiné à PRINTER (voir instruction PRINT).

$COLUMN-SPACING OF MONFIC, par exemple, se rapporte à l'état destiné à MONFIC alors que $COLUMN-SPACING se rapporte à l'état destiné à PRINTER.

8.2.2 Exemple de création d'un état au moyen de variables système

Soit la requête suivante :

```
10    :  LET @COLUMN-BORDER = "I"
20    :  LET $COLUMN-SPACING = 3
30    :  LET @LINE-BORDER = "X"
40    :  RETRIEVE ETUDIANT
50    :   PRINT WITH TITLE E-NOM, GRADE, DATE-NAISSANCE
60    :  END
```

L'état produit est le suivant :

```
IXXXXXXXXXXXXXXXXXXXXXXIXXXXXXXXXXXIXXXXXXXXXIXXXXXXXXIXXXXXXXXI   (30)
I E-NOM                 I GRADE      I N-ANNEE  I N-MOIS  I N-JOUR  I   (50)
IXXXXXXXXXXXXXXXXXXXXXXIXXXXXXXXXXXIXXXXXXXXXIXXXXXXXXIXXXXXXXXI   (30)
I ARNAUD                I LICENCE    I       61 I      10 I       9 I
I SERREAU               I MAITRISE   I       59 I      11 I      23 I
I WAGNER                I MAITRISE   I       57 I       8 I      15 I
I MARTIN                I LICENCE    I       63 I       1 I      22 I   (50)
I MARTY                 I MAITRISE   I       58 I       4 I      19 I
I SARRE                 I LICENCE    I       62 I       7 I      16 I
I WIART                 I MAITRISE   I       56 I       3 I      27 I
IXXXXXXXXXXXXXXXXXXXXXXIXXXXXXXXXXXIXXXXXXXXXIXXXXXXXXIXXXXXXXXI   (30)
(10)                    (10)         (10)       (10)      (10)      (10)
```

Chaque nombre entouré d'un cercle représente le numéro de ligne (dans la requête) de l'instruction qui commande l'élément considéré.
A noter que les en-têtes de colonnes et les lignes détail sont déterminés par l'instruction PRINT de la ligne 50.

Le nombre d'espaces entre les colonnes est déterminé par la ligne 20.

8.3 Etats complexes

8.3.1 Généralités

Pour créer des états plus élaborés, il est nécessaire de définir une description d'état.

Un état se compose des éléments suivants :

- en-tête d'état
- fin d'état
- en-tête de page
- bas de page
- lignes détail

L'en-tête d'état (clause REPORT HEADING) contient des informations à n'imprimer qu'une seule fois, en début d'état. Il s'agit d'informations telles que titres, droits de reproduction, restrictions de diffusion, liste de diffusion, etc.

La fin d'état (clause REPORT FOOTING) contient des informations à n'imprimer qu'une seule fois, en fin d'état. Elles peuvent être identiques à celles de l'en-tête d'état.

L'en-tête de page (clause PAGE HEADING) contient des informations à imprimer au début de chaque page de l'état (à l'exclusion de l'en-tête et de la fin d'état). Il s'agit d'informations telles que titres (devant apparaître sur chaque page), nom de l'entreprise, etc.

Le bas de page (clause PAGE FOOTING) contient des informations à imprimer à la fin de chaque page de l'état. Elles sont identiques à celles de l'en-tête de page, mais peuvent également inclure des sous-totaux (par exemple).

Les lignes détail contiennent les données proprement dites de l'état. Ces informations se présentent généralement comme une suite de zones imprimées sur la largeur de la page, dans les colonnes correspondantes.

Une description d'état (voir plus loin) permet à l'utilisateur de définir le contenu et la présentation des en-têtes et fins d'états et des en-têtes et bas de pages. Le contenu des lignes détail est défini à l'aide de l'instruction PRINT, leur présentation pouvant être affectée par la description d'état.

8.3.2 Structure de page

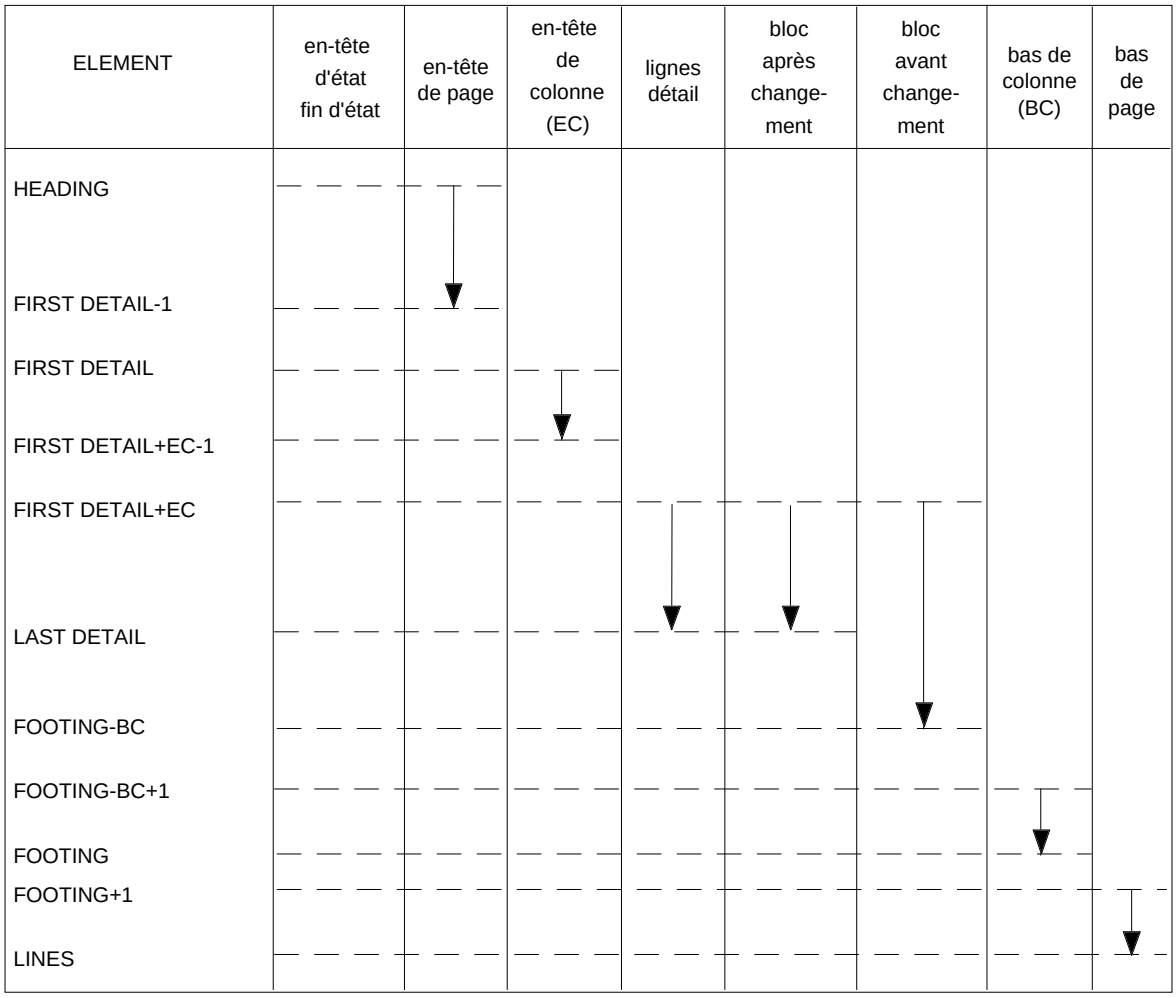
La structure d'une page est schématisée à la figure 8-1 qui montre le positionnement vertical de chaque élément de l'état.
A part EC et BC (définis plus loin), tous ces symboles sont des paramètres de la clause de description d'état PAGE LIMIT. Si l'utilisateur n'affecte pas de valeurs à ces paramètres, ce sont leurs valeurs implicites qui sont appliquées. Les flèches indiquent les lignes disponibles pour l'impression des éléments concernés.

Les différents éléments d'un état ont été décrits précédemment, à l'exception des quatre suivants :

- l'en-tête de colonnes, à imprimer en haut de chaque colonne.
  Il est défini au moyen de l'instruction HEADING et des instructions PRINT associées ou au moyen d'instructions PRINT WITH TITLE. Il peut comporter 1 à 3 lignes de texte (voir la clause TITLE de l'instruction PRINT au chapitre 6) plus les lignes de séparation horizontale.
- le bas de colonnes, à imprimer en bas de chaque colonne.
  Il est défini au moyen de l'instruction FOOTING et des instructions PRINT associées. Il peut comporter 1 à 3 lignes de texte plus les lignes de séparation horizontale.
- le bloc après changement, qui identifie le groupe de lignes imprimées par une instruction PRINT figurant dans une séquence AFTER ... CHANGE (voir "Instructions de contrôle" au chapitre 6).
- le bloc avant changement, qui identifie le groupe de lignes imprimées par une instruction PRINT figurant dans une séquence BEFORE ... CHANGE (voir "Instructions de contrôle" au chapitre 6).

Pour plus de détails sur ces éléments, se reporter aux instructions du langage de requêtes appropriées.

Hormis dans la figure 8-1, l'en-tête de colonnes, le bas de colonnes, le bloc après changement et le bloc avant changement sont considérés dans ce chapitre comme des lignes détail.



Remarque: En l'absence d'en-têtes de colonnes, EC=0
En l'absence de bas de colonnes, BC=0

Figure 8-1. Structure de page d'état complexe

8.3.3 Description d'état

Une description d'état définit le format de présentation d'un état, à l'exclusion du contenu des lignes détail qui est défini par l'instruction PRINT.

Une description d'état se compose de clauses dont la liste complète est fournie à la figure 8-2. Ces clauses sont décrites en détail à la fin de ce chapitre. La première est obligatoirement REPORT et la dernière END.
Les autres sont facultatives et utilisables suivant les besoins. Certaines de ces clauses peuvent être remplacées par une variable système affectée de la valeur appropriée (voir tableau 8-1). Lorsqu'une clause et la variable système équivalente coexistent, c'est la valeur affectée à la variable système dans la requête qui prime sur celle définie dans la clause.

Avant d'être utilisée, une description d'état doit être compilée (au moyen de la commande COMPILE REPORT), puis sauvegardée (au moyen de la commande SAVE REPORT). **Une description d'état ne peut être exécutée seule, elle doit être associée à une requête.** Cette association peut être effectuée dynamiquement à l'exécution, par l'instruction ou la commande suivante :

```
USE REPORT <nom-état> [ON <fichier-impression>]
```

La commande USE REPORT peut être ensuite annulée par la commande CANCEL REPORT (pour le même nom-fichier). Une requête qui produit plusieurs états (par l'intermédiaire de l'option TO nom-fichier de PRINT) peut utiliser la même description d'état ou des descriptions différentes pour chaque état. De même, une description d'état peut être utilisée par plusieurs requêtes.

REMARQUE :

Si les clauses de description d'état sont définies à partir du nouveau processeur de formats IQS-V4 (voir le volume 1), elles sont automatiquement intégrées au format correspondant. Lorsque ce format est associé au fichier à traiter, les clauses de description d'état associées sont automatiquement prises en compte pour l'impression. Il n'est donc pas nécessaire d'utiliser une commande USE REPORT. A noter que, dans ce cas, la description d'état n'est pas sauvegardée séparément.

```
REPORT

                {[LINE] <liste-descripteur-éléments> }
[REPORT HEADING {SPACE <n>                           }
                {SPACE TOP                           }

                {[LINE] <liste-descripteur-éléments> }
               [{SPACE <n>                           }]...]
                {SPACE TOP                           }

                {[LINE] <liste-descripteur-éléments> }
[REPORT FOOTING {SPACE <n>                           }
                {SPACE TOP                           }

                {[LINE] <liste-descripteur-éléments> }
               [{SPACE <n>                           }]...]
                {SPACE TOP                           }

[PAGE LIMIT [<n> LINES] [,HEADING <n>] [,FOOTING <n>]
                [,FIRST [DETAIL] <n>] [,LAST [DETAIL] <n>]]

[PAGE WIDTH <n>]

              {[LINE] <liste-descripteur-éléments> }
[PAGE HEADING {                                    }
              {SPACE <n>                           }

              {[LINE] <liste-descripteur-éléments> }
             [{                                    }]...]
              {SPACE <N<                           }

              {[LINE] <liste-descripteur-éléments> }
[PAGE FOOTING {                                    }
              {SPACE <n>                           }

              {[LINE] <liste-descripteur-éléments> }
             [{                                    }]...]
              {SPACE <n>                           }

[STARTING PAGE NUMBER <n>]

[STARTING PAGE DETAIL <n>]

[STARTING COLUMN NUMBER <n>]

[NUMBER OF PAGES <n>]

[COL[UMN] SPACING <n>]

[COL[UMN] BORDER "<c>"]

[LINE SPACING <n>]

[LINE BORDER "<c>"]

END
```

Figure 8-2. Format général d'une description d'état

8.3.4 Gestion des descriptions d'état

Pour gérer les descriptions d'état, l'utilisateur dispose de commandes IQS et d'instructions du langage de requêtes.

Les instructions (tableau 8-3) sont traitées dans les chapitres 5 à 7.

Les commandes, récapitulées au tableau 8-2, sont décrites en détail dans le volume 1 de ce manuel.

Tableau 8-2. Commandes de gestion des descriptions d'état

Commande	Description
COMPILE REPORT	Compilation d'une ou plusieurs descriptions d'état.
DEFINE REPORT	Appel du processeur d'états interactif pour définir un nouvel état.
DROP REPORT	Suppression d'une ou plusieurs descriptions d'état enregistrées.
LIST REPORT	Listage d'informations sur une ou plusieurs descriptions d'état.
UPDATE REPORT	Appel du processeur d'états interactif pour mettre à jour une description d'état.
AUTO	Appel de l'éditeur de texte IQS pour introduction d'une nouvelle description d'état dans l'espace de travail origine.
LOAD	Chargement d'une description d'état origine dans l'espace de travail.
SAVE REPORT	Sauvegarde d'une description d'état (version origine et éventuellement version résultante).
REPLACE REPORT	Remplacement d'une description d'état existante par une version modifiée ou une nouvelle description portant le même nom.
CANCEL REPORT	Suppression du lien existant entre une description d'état et un fichier d'impression.
RECOMPILE REPORT	Recompilation d'une ou plusieurs descriptions d'état.
USE REPORT	Association d'une description d'état à un fichier d'impression.

Tableau 8-3. Instructions de gestion des descriptions d'état

Instruction	Description
CANCEL REPORT	Suppression du lien existant entre une description d'état et un fichier d'impression.
USE REPORT	Association d'une description d'état à un fichier d'impression.
REPORT	Edition d'un fichier de travail selon la description d'état (ou le format de présentation) spécifié et envoi du résultat dans un fichier d'impression.

8.3.5 Exemple de création d'état au moyen d'une description d'état

Soit la description d'état suivante :

```
10    :  REPORT
20    :  COLUMN BORDER "I"
30    :  COLUMN SPACING 3
40    :  LINE BORDER "X"
60    :  END
```

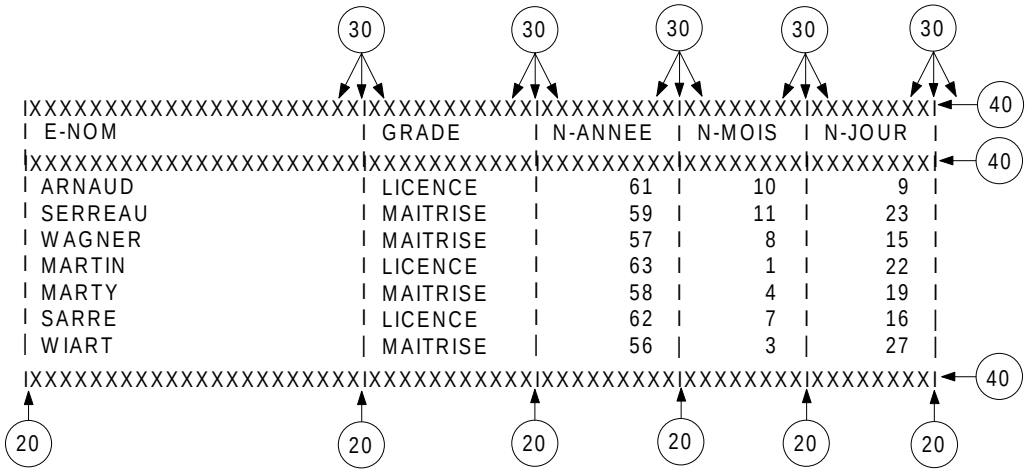
Cette description d'état est compilée, puis sauvegardée à l'aide des deux commandes suivantes :

```
COMPILE REPORT
SAVE REPORT ETAT-SIMPLE
```

Cette description d'état peut alors être associée (statiquement) à la requête suivante (instruction USE REPORT à la ligne 10) :

```
10    :  USE REPORT ETAT-SIMPLE
20    :  RETRIEVE ETUDIANT
30    :  PRINT WITH TITLE E-NOM, GRADE, DATE-NAISSANCE
40    :  END
```

L'exécution de la requête produit l'état suivant :



Chaque nombre entouré d'un cercle indique le numéro de ligne (dans la description d'état) de la clause qui commande l'élément considéré. Les en-têtes de colonnes et les lignes détail sont déterminés par l'instruction PRINT (ligne 30 de la requête). Cet état est identique à celui généré précédemment à l'aide de variables système.

8.3.6 Exemple avec une description d'état complète

Soit la description d'état suivante :

```
 10 : REPORT
 20 : COLUMN BORDER "*"
 30 : COLUMN SPACING 3
 40 : LINE BORDER "#"
 50 : NUMBER OF PAGES 20
 60 : PAGE WIDTH 60
 70 : PAGE LIMIT 18 LINES HEADING 2 FOOTING 15
 80 :      FIRST DETAIL 6 LAST DETAIL 13
 90 : REPORT HEADING SPACE 5
100 :      LINE "UNIVERSITE DE PARIS-1" COL 3
110 :      SPACE
120 :      LINE "RAPPORT FINANCIER CONFIDENTIEL" COL 3
130 :      SPACE
140 :      LINE "PREPARE LE" COL 3, $DAY, @MONTH SHORT, $YEAR
150 : PAGE HEADING LINE "UNIVERSITE DE PARIS-1" COL 3
160 :      SPACE
170 :      LINE "RAPPORT FINANCIER TRIMESTRIEL" COL 3
180 : PAGE FOOTING SPACE
190 :      LINE "DATE" COL 5,$DAY,@MONTH SHORT,$YEAR,
           "PAGE" COL 41,$PAGE JUST LEFT
200 : REPORT FOOTING
210 :      LINE "RAPPORT FINANCIER TRIMESTRIEL" COL 4
220 :      SPACE
230 :      LINE "PREPARE PAR" COL 4
240 :           @USER, "LE", $DAY, @MONTH SHORT, $YEAR
250 : END
```

Cette description d'état est compilée et sauvegardée à l'aide des commandes suivantes :

```
COMPILE REPORT
SAVE REPORT ETAT-COMPLET
```

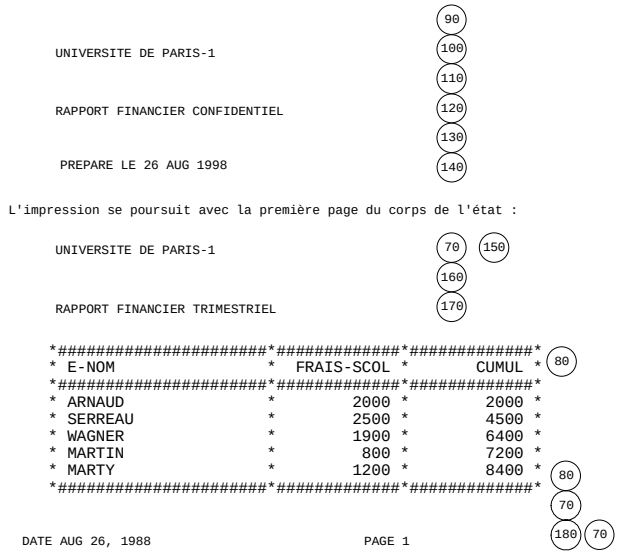
Elle peut ensuite être associée à la requête suivante (instruction USE REPORT de la ligne 10) :

```
10 : USE REPORT ETAT-COMPLET
20 : RETRIEVE ETUDIANT
30 :   LET $CUMUL = SUM (FRAIS-SCOL)
40 :   PRINT WITH TITLE E-NOM, FRAIS-SCOL, $CUMUL
50 : END
60 : SPACE TOP
```

L'état obtenu est présenté et commenté ci-après.
Chaque nombre entouré d'un cercle indique le numéro de ligne (dans la description d'état) de la clause qui détermine l'élément considéré.

Exemple d'en-tête d'état

Il commence sur une nouvelle page et se termine par un saut (automatique) à la page suivante :



Sauf indication contraire, dans les explications qui suivent, les numéros de ligne cités correspondent à ceux de la page de l'état considérée. A noter que la première ligne de la page est blanche.

Le paramètre HEADING ayant pour valeur 2 (ligne 70 de la description d'état), l'en-tête de page commence à la ligne 2 de la page. Le paramètre LAST DETAIL ayant pour valeur 13 (ligne 80 de la description d'état), la ligne 14 de la page est blanche. De même, le paramètre FIRST DETAIL ayant pour valeur 6 (ligne 80 de la description d'état), la ligne 5 qui suit l'en-tête de page est également blanche.

Les lignes détail sont déterminées par l'instruction PRINT (ligne 40 de la requête).

La page suivante de l'état se présente comme suit :

```
    UNIVERSITE DE PARIS-1                                          (150)
                                                                   (160)

    RAPPORT FINANCIER TRIMESTRIEL                                  (170)

*######################*##############*##############*
* E-NOM                *  FRAIS-SCOL  *       CUMUL  *
*######################*##############*##############*
* SARRE                *         1100 *         9500 *
* WIART                *         1500 *        11000 *
*                      *              *              *
*                      *              *              *
*                      *              *              *
*                      *              *              *
*######################*##############*##############*
                                                                   (180)

     DATE AUG 26, 1988                    PAGE 2
```

La dernière page est constituée par la fin d'état :

```
RAPPORT FINANCIER TRIMESTRIEL                                  (200)
                                                               (210)

PREPARE PAR SECRET      LE 26 AUG 1988                  (220)  (230)
```

A noter que l'instruction SPACE TOP à la ligne 60 de la requête est nécessaire pour que la fin d'état apparaisse sur une nouvelle page. Sinon, le bloc de fin d'état serait imprimé à la page 2 et dans ce cas, les lignes détail vierges et la fin de page de cette dernière ne seraient pas imprimées.

8.4 Clauses de description d'état

La suite de ce chapitre est consacrée à une analyse détaillée des clauses de description d'état. Celles-ci sont présentées dans l'ordre alphabétique pour faciliter la consultation.

Ces clauses sont les suivantes :

```
COLUMN BORDER
COLUMN SPACING
liste-descripteurs-éléments (COLUMN, clauses cadrage et édition)
END
LINE BORDER
LINE SPACING
NUMBER OF PAGES
PAGE FOOTING
PAGE HEADING
PAGE LIMIT
PAGE WIDTH
REPORT
REPORT FOOTING
REPORT HEADING
STARTING COLUMN NUMBER
STARTING PAGE DETAIL
STARTING PAGE NUMBER
```

8.4.1 COLUMN BORDER

Fonction :

Spécification du caractère à utiliser pour constituer les lignes de séparation verticale d'un état.

Format :

```
COL[UMN] BORDER "<c>"
```

Description de la clause :

Cette clause permet de spécifier le caractère séparateur de colonnes (séparation des zones constitutives d'une ligne d'état).
Si l'espacement entre colonnes est n (n étant défini par la clause COLUMN SPACING, valeur implicite 1), "c" est imprimé à la position (n/2) si n est pair et à la position (n+1)/2 si n est impair. Le séparateur de colonnes est également imprimé en première position de chaque ligne (avec suppression des espaces début) et après le dernier élément de chaque ligne (avec suppression des espaces fin). Si la clause COLUMN BORDER est omise, le séparateur de colonnes implicite est l'espace.

Si une valeur est affectée à la variable système @COLUMN-BORDER dans une requête, elle prime sur celle spécifiée par la clause COLUMN BORDER dans la description d'état.

Description du paramètre :

"c" Caractère alphanumérique ou caractère spécial, c'est-à-dire n'importe quel caractère pouvant apparaître dans un littéral (voir chapitre 2). Ce caractère doit être entre guillemets (").

EXEMPLE :

Si l'instruction PRINT associée est la suivante :

```
PRINT WITH TITLE GRADE, $MONTANT
❑
```

les clauses de description d'état suivantes :

```
COLUMN SPACING 3
COLUMN BORDER "I"
```

génèreront l'état suivant (lignes détail) :

```
I GRADE          I MONTANT I

I LICENCE        I   001.3 I
I MAITRISE       I   201.6 I
I MAITRISE       I   020.8 I
```

8.4.2 COLUMN SPACING

Fonction :

Spécification du nombre d'espaces à insérer entre les colonnes d'un état.

Format :

```
 -------------------------
|                   { 1 }  |
|  COL[UMN] SPACING {   }  |
|                   {<n>}  |
 -------------------------
```

Description de la clause :

Cette clause permet de spécifier l'espacement entre colonnes (nombre d'espaces entre les zones constitutives d'une ligne d'état).

La valeur implicite est 1 espace.

Au lieu d'utiliser la clause COLUMN SPACING dans une description d'état, il est possible de spécifier l'espacement entre colonnes directement dans une requête en affectant une valeur à la variable système $COLUMN-SPACING ou en utilisant l'option COLUMN dans l'instruction PRINT. En cas de coexistence entre ces différentes méthodes, la règle de priorité est la suivante : l'instruction PRINT avec COLUMN prime sur la variable $COLUMN-SPACING qui elle-même prime sur la clause COLUMN SPACING.

Description du paramètre :

Entier définissant le nombre d'espaces implicite entre les colonnes de l'état. Ce paramètre doit être compatible avec celui de la clause PAGE WIDTH. Valeur maximale 255. Valeur implicite 1.

EXEMPLE :

```
COLUMN SPACING 3
❑
```

Les zones constitutives de la ligne seront séparées par 3 espaces (sauf si la variable système $COLUMN SPACING ou l'option COLUMN de l'instruction PRINT prime sur cette clause).

8.4.3 END

Fonction :

Indication de la fin d'une description d'état (dont le début est indiqué par une clause REPORT).

Format :

```
|--------|
|  END   |
|--------|
```

Description de la clause :

Cette clause permet de marquer la fin d'une description d'état (le début étant indiqué par la clause REPORT).

EXEMPLE :

```
REPORT

  PAGE HEADING ...

  PAGE WIDTH ...
  .
  .
  .

END
```

❑

La clause END indique la fin de la description d'état.

8.4.4 LINE BORDER

Fonction :

Spécification du caractère à utiliser pour constituer les lignes de séparation horizontale d'un état.

Format :

```
|------------------------|
|   LINE BORDER "<c>"    |
|------------------------|
```

Description de la clause :

Cette clause permet de spécifier le caractère séparateur de lignes (séparation entre titres de colonnes et lignes détail).
Si une instruction HEADING, FOOTING ou PRINT WITH TITLE (voir chapitre 6) figure dans la requête, le haut et le bas de colonnes sont délimités par des lignes de séparation horizontale.

Si une valeur est affectée à la variable système @LINE-BORDER dans une requête, elle prime sur celle spécifiée par la clause LINE BORDER dans la description d'état.

Description du paramètre :

Caractère à utiliser pour constituer les lignes de séparation horizontale. La valeur implicite est l'espace. Ce caractère doit être entre guillemets (").

EXEMPLE :

```
LINE BORDER "X"
```

❑

Les lignes de séparation horizontale seront constituées de X.

8.4.5 LINE SPACING

Fonction :

Spécification du nombre de lignes vides entre deux lignes détail. La valeur implicite est zéro.

Format :

```
LINE SPACING <n>
```

Description du paramètre :

Entier positif ou 0, obligatoirement compatible avec les valeurs de PAGE LIMIT. La valeur implicite est 0 (zéro).

8.4.6 Liste-descripteurs-éléments

Fonction :

Spécification de la liste des zones devant figurer dans l'en-tête/fin d'état et dans l'en-tête/fin de page.

Format :

```
 ------------------------------------------------------------
| {<descripteur-littéral>}     {<descripteur-littéral>}     |
| {                       }   [,{                       }] ...|
| {<descripteur-zone>    }     {<descripteur-zone>    }     |
 ------------------------------------------------------------
```

Descripteur-littéral a la forme suivante :

```
|---------------------------------------|
|    <littéral> [ COL[UMN] [+] <n>]     |
|---------------------------------------|
```

Descripteur-zone a la forme suivante :

```
|-------------------------------------------|
| <variable-système> [ COL[UMN] [+] <n>]    |
|                                           |
|         [<clause-cadrage>]                |
|                                           |
|         [<clause-édition>] [SHORT]        |
|-------------------------------------------|
```

Description :

La liste des variables système est fournie au chapitre 3. Littéral, clause-cadrage et clause-édition sont décrits au chapitre 2. COLUMN et SHORT sont décrits dans l'instruction PRINT au chapitre 6.

Notez que l'espacement implicite entre les zones est toujours 1.

8.4.7 NUMBER OF PAGES

Fonction :

Définition du nombre maximum de pages d'un état.

Format :

```
|------------------------|
|  NUMBER OF PAGES <n>   |
|------------------------|
```

Description de la clause :

Cette clause permet de spécifier le nombre maximum de pages d'un état (à l'exclusion de l'en-tête et de la fin d'état). A l'exécution, lorsque ce nombre de pages est atteint, un message de la forme suivante est émis :

```
*** 40 Dépassement du nombre maximum de pages
```

Dans cet exemple, 40 représente le numéro de ligne de l'instruction PRINT ayant provoqué le dépassement. Aucune page n'est plus imprimée pour cet état (en dehors de la fin d'état). Cette possibilité peut être utilisée pour obtenir un échantillon d'état afin de s'assurer que la présentation et le format sont bien ceux désirés.

Si NUMBER OF PAGES n'est pas spécifié, la valeur implicite est nombre de pages illimité.

Si une valeur est affectée à la variable système $MAX-PAGES dans une requête, elle prime sur celle spécifiée par la clause NUMBER OF PAGES dans la description d'état.

Description du paramètre :

Entier (6 chiffres maximum) représentant le nombre maximum de pages d'un état.

EXEMPLE :

```
NUMBER OF PAGES 5
❑
```

Les 5 premières pages seront imprimées (en plus de l'en-tête et de la fin d'état).

8.4.8 PAGE FOOTING

Fonction :

Spécification du format et du contenu du bas de page.

Format :

```
 ---------------------------------------------------------
|              {[LINE] <liste-descripteurs-éléments>}     |
| PAGE FOOTING {                                   }      |
|              {SPACE <n>                          }      |
|                                                         |
|              {LINE <liste-descripteurs-éléments>}       |
|            [ {                                   } ] ... |
|              {SPACE <n>                          }      |
 ---------------------------------------------------------
```

Description de la clause :

Cette clause permet de spécifier le format et le contenu du bloc de bas de page. Ce bloc se compose d'une ou plusieurs lignes à imprimer à la fin de chaque page. Il est imprimé sur toutes les pages, sauf celles contenant l'en-tête d'état et la fin d'état.

A l'intérieur du bas de page, il est possible de sauter des lignes au moyen du paramètre SPACE. Le bas de page peut contenir des informations constantes (chaînes de caractères) et/ou variables (variables système). Le nombre maximum de lignes (lignes sautées incluses) du bas de page est fonction de la valeur des paramètres LINES et FOOTING (voir clause PAGE LIMIT).

Lorsqu'une ligne a une longueur supérieure à la largeur de la page, elle est tronquée à droite si PAGE WIDTH n'a pas été explicitement fixé, ou bien, si PAGE WIDTH a été explicitement fixé, le compilateur signale une erreur.

A noter qu'une instruction SPACE TOP est obligatoire en fin de requête pour obtenir le bas de page sur la dernière page du corps de l'état, la fin d'état étant alors imprimée seule sur une nouvelle page.

Description des paramètres :

LINE	Ce paramètre spécifie que les éléments qui suivent (jusqu'au prochain paramètre LINE, s'il y en a un) constituent une ligne. Si le bas de page ne comporte qu'une ligne, LINE peut être omis.
liste-descripteurs-éléments	Cette liste spécifie les zones constitutives de la ligne. Pour une description détaillée, voir "Liste-descripteurs-éléments" plus haut dans ce chapitre.
SPACE n	Ce paramètre provoque un saut de n lignes, n étant un entier positif dont la valeur maximum est 255. La variable système $LINE est incrémentée de cette valeur. La variable système $PAGE n'est pas affectée. Si n est omis, la valeur implicite est 1 et SPACE génère une ligne blanche. Si le nombre de lignes de la page courante est dépassé, le résultat est indéterminé.

EXEMPLE :

```
PAGE FOOTING SPACE 1 LINE "BLOC BAS DE PAGE"

    LINE "DATE", @DATE, "PAGE" COL +30, $PAGE JUST LEFT
```

❑

Le bas de page sera imprimé en bas de chaque page, sauf celles contenant l'en-tête et la fin d'état.

8.4.9 PAGE HEADING

Fonction :

Spécification du format et du contenu de l'en-tête de page.

Format :

```
 ---------------------------------------------------------
|              {[LINE] <liste-descripteurs-éléments>}     |
| PAGE HEADING {                                   }      |
|              {SPACE <n>                          }      |
|                                                         |
|              {LINE <liste-descripteurs-éléments>}       |
|            [ {                                  } ] ... |
|              {SPACE <n>                          }      |
 ---------------------------------------------------------
```

Description de la clause :

Cette clause permet de spécifier le format et le contenu de l'en-tête de page. Cet en-tête se compose d'une ou plusieurs lignes à imprimer au début de chaque page. Il est imprimé sur toutes les pages, sauf celles contenant l'en-tête d'état et la fin d'état (si cette dernière commence sur une nouvelle page).

A l'intérieur de l'en-tête de page, il est possible de sauter des lignes au moyen du paramètre SPACE. L'en-tête de page peut contenir des informations constantes (chaînes de caractères) et/ou variables (variables système). Le nombre maximum de lignes (lignes sautées incluses) de l'en-tête de page est égal à FIRST DETAIL - HEADING (voir clause PAGE LIMIT). La longueur de la ligne ne doit pas être supérieure à la largeur de la page (PAGE WIDTH). Lorsqu'une ligne a une longueur supérieure à la largeur de la page, elle est tronquée à droite si PAGE WIDTH n'a pas été explicitement fixé, ou bien si PAGE WIDTH a été explicitement fixé, le compilateur signale une erreur.

Il n'y a pas de saut de page automatique en début de requête. Pour que le premier en-tête de page soit correctement affiché, il faut soit utiliser la clause REPORT HEADING, soit une instruction EJECT (pour effacer l'écran) suivie d'une instruction SPACE TOP.

Description des paramètres :

LINE	Ce paramètre spécifie que les éléments qui suivent (jusqu'au prochain paramètre LINE, s'il y en a un) constituent une ligne. Si l'en-tête de page ne comporte qu'une ligne, LINE peut être omis.
liste-descripteurs-éléments	Cette liste spécifie les zones constitutives de la ligne. Pour une description détaillée, voir "Liste-descripteurs-éléments" plus haut dans ce chapitre.
SPACE n	Ce paramètre provoque un saut de n lignes, n étant un entier positif dont la valeur maximum est 255. La variable système $LINE est incrémentée de cette valeur. La variable système $PAGE n'est pas affectée. Si n est omis, la valeur implicite est 1 et SPACE génère une ligne blanche. Si le nombre de lignes de la page courante est dépassé, le résultat est indéterminé.

EXEMPLE :

```
PAGE HEADING SPACE 2 LINE "EN-TETE DE PAGE"
    LINE "DATE", @DATE, "PAGE" COL 40, $PAGE JUST LEFT
```

❑

L'en-tête de page sera imprimé au début de chaque page, sauf celles contenant l'en-tête et la fin d'état (si cette dernière commence sur une nouvelle page).

8.4.10 PAGE LIMIT

Fonction :

Définition du format vertical d'une page d'état.

Format :

```
 ---------------------------------------
|  PAGE LIMIT [<n> LINES]                |
|                                        |
|             [,HEADING <n>]             |
|                                        |
|             [,FOOTING <n>]             |
|                                        |
|             [,FIRST [DETAIL] <n>]      |
|                                        |
|             [,LAST [DETAIL] <n>]       |
 ---------------------------------------
```

Description de la clause :

Cette clause permet de définir le format vertical d'une page d'état. Elle indique le nombre maximum de lignes par page et la position de l'en-tête d'état et/ou de page, de la fin d'état et/ou du bas de page et des lignes détail. Voir la description des paramètres ci-dessous et la figure 8-1 au début du chapitre. A la compilation, les contrôles suivants sont effectués :

```
HEADING + nombre de lignes dans PAGE HEADING <= FIRST DETAIL
```

```
FIRST DETAIL <= LAST DETAIL <= FOOTING
```

```
FOOTING + nombre de lignes dans PAGE FOOTING <= LINES
```

Si l'un des contrôles est négatif, un message d'erreur est émis et la compilation s'arrête prématurément.

A l'exécution, la valeur de LINES est contrôlée pour s'assurer qu'elle est compatible avec l'appareil de sortie utilisé. En cas d'incompatibilité, un message d'erreur est émis et l'exécution s'arrête prématurément.

Tous les paramètres spécifiables dans cette clause ont pour valeur maximum 255.

Description des paramètres :

n LINES	Nombre total de lignes imprimables par page. Si une valeur est affectée à la variable système $PAGE-HEIGHT dans une requête, elle prime sur celle spécifiée par LINES. Sinon, la valeur implicite est de 56 lignes pour une imprimante et illimitée pour un terminal. La valeur spécifiée (ou implicite) doit être compatible avec l'appareil de sortie utilisé à l'exécution.
HEADING n	Numéro de la première ligne de REPORT HEADING et/ou de PAGE HEADING. HEADING est obligatoire si PAGE HEADING ou REPORT HEADING est utilisé. La valeur spécifiée est contrôlée à la compilation de l'état (voir plus haut "Description de la clause").
FOOTING n	Numéro de la ligne précédant la première de PAGE FOOTING. Si PAGE FOOTING n'est pas spécifié, la valeur implicite est soit LAST DETAIL (s'il est spécifié), soit LINES (si LAST DETAIL est omis). La valeur de FOOTING doit être supérieure à celle de LAST DETAIL lorsque LINE BORDER est utilisé ; sinon, il se produit un débordement de page entraînant un arrêt prématuré de l'exécution. La valeur spécifiée est contrôlée à la compilation de l'état (voir plus haut "Description de la clause").
FIRST DETAIL n	Numéro de la première ligne détail de chaque page d'état. Si FIRST DETAIL n'est pas spécifié, la valeur implicite est soit HEADING, si ce dernier est spécifié, soit 1. La valeur spécifiée est contrôlée à la compilation de l'état (voir plus haut "Description de la clause").

LAST DETAIL n

Numéro de la dernière ligne détail de chaque page d'état. Si LAST DETAIL n'est pas spécifié, la valeur implicite est soit FOOTING (s'il est spécifié), soit LINES (si FOOTING est omis).

La valeur de FOOTING doit être supérieure à celle de LAST DETAIL lorsque LINE BORDER est utilisé ; sinon, il se produit un débordement de page entraînant un arrêt prématuré de l'exécution.

La valeur spécifiée est contrôlée à la compilation de l'état (voir plus haut "Description de la clause").

A noter que les lignes détail peuvent être imprimées au-delà de LAST DETAIL jusqu'à FOOTING (inclus) si elles proviennent d'une instruction PRINT figurant dans une boucle BEFORE ou d'une instruction FOOTING.

EXEMPLE :

```
PAGE LIMIT 30 LINES, HEADING 3, FOOTING 26,
             FIRST DETAIL 6, LAST DETAIL 23
```

❑

La page d'état contient 30 lignes. L'en-tête d'état et l'en-tête de page (de chaque page) commencent à la ligne 3. La première ligne détail sera la ligne 6. Le nombre maximum de lignes de l'en-tête de page est 3 (FIRST DETAIL - HEADING). La dernière ligne détail est la ligne 23 et le bas de page commencera à la ligne 27 (soit FOOTING + 1). Le bas de page peut contenir 4 lignes au maximum (LINES - FOOTING).

8.4.11 PAGE WIDTH

Fonction :

Spécification du nombre de positions d'impression par ligne.

Format :

```
|------------------------|
|    PAGE WIDTH <n>      |
|------------------------|
```

Description de la clause :

Cette clause permet de spécifier la largeur de page, c'est-à-dire le nombre de positions d'impression par ligne. Si une valeur est affectée à la variable système $PAGE-WIDTH dans une requête, elle prime sur celle spécifiée par la clause PAGE WIDTH.

Si PAGE WIDTH est spécifié, un contrôle est effectué à la compilation de l'état pour s'assurer que les longueurs de lignes définies dans REPORT HEADING, REPORT FOOTING, PAGE HEADING et PAGE FOOTING ne sont pas supérieures à PAGE WIDTH. Si ce contrôle est négatif, un message d'erreur est émis et la compilation s'arrête prématurément. La largeur de page spécifiée doit être compatible avec l'appareil de sortie utilisé à l'exécution.

Si PAGE WIDTH n'est pas spécifié, une valeur implicite est attribuée à l'exécution en fonction du support de sortie utilisé, comme suit :

- 132 pour une imprimante ligne.
- pour un fichier séquentiel, longueur de l'article si elle est inférieure à 255, ou dans le cas contraire 255.
- longueur de ligne pour les autres terminaux.

A l'exécution, les lignes dont la longueur est supérieure à la largeur de page sont tronquées à droite. En aucun cas leur longueur ne peut être supérieure à 255.

Description du paramètre :

Nombre de positions d'impression par ligne, sous la forme d'un entier de valeur maximum 255.

Une valeur implicite, dépendant du support de sortie utilisé, est attribuée à l'exécution (voir plus haut "Description de la clause").

EXEMPLE :

```
PAGE WIDTH 60
❑
```

Cette clause spécifie une largeur de page de 60 caractères.

8.4.12 REPORT

Fonction :

Indication du début d'une description d'état.

Format :

```
 ------------------
|  REPORT          |
|     ...          |
|     <clauses>    |
|     ...          |
|  END             |
 ------------------
```

Description de la clause :

Cette clause permet de marquer le début d'une description d'état (la fin étant indiquée par la clause END).

EXEMPLE :

```
REPORT
     REPORT HEADING ...
     PAGE HEADING ...
     .
     .
     .
END
```

❑

La clause REPORT indique le début de la description d'état. La clause END indique la fin de la description d'état.

8.4.13 REPORT FOOTING

Fonction :

Spécification du format et du contenu de la fin d'état.

Format :

```
 ------------------------------------------------------------
|                {[LINE] <liste-descripteurs-éléments>}      |
| REPORT FOOTING {SPACE <n>                            }     |
|                {SPACE TOP                            }     |
|                                                            |
|                {LINE <liste-descripteurs-éléments>}        |
|              [ {SPACE <n>                          } ] ... |
|                {SPACE TOP                            }     |
 ------------------------------------------------------------
```

Description de la clause :

Cette clause permet de spécifier le format et le contenu du bloc de fin d'état. Ce bloc est imprimé une seule fois à la fin de l'état. Contrairement à l'en-tête d'état, il ne commence pas automatiquement sur une nouvelle page. Pour ce faire, l'utilisateur doit placer une instruction SPACE TOP à la fin de la requête. Le dernier bas de page avant la fin d'état n'est imprimé que si cette dernière commence sur une nouvelle page.

A l'intérieur de la fin d'état, les sauts de lignes sont commandés par l'utilisateur. SPACE n permet de sauter n lignes et SPACE TOP de passer à une nouvelle page. Le contrôle de compatibilité entre le nombre de lignes imprimées/sautées et le nombre maximum de lignes par page incombe à l'utilisateur. La fin d'état peut s'étendre sur plusieurs pages si nécessaire.

La longueur des lignes ne doit pas dépasser la largeur de page (voir clause PAGE WIDTH). La fin d'état peut contenir des informations constantes (chaînes de caractères) et/ou variables (variables système).

Description des paramètres :

LINE	Ce paramètre spécifie que les éléments qui suivent (jusqu'au prochain paramètre LINE, s'il y en a un) constituent une ligne. Si la fin d'état ne comporte qu'une ligne, LINE peut être omis.
liste-descripteurs-éléments	Cette liste spécifie les zones constitutives de la ligne. Pour une description détaillée, voir "Liste-descripteurs-éléments" (plus haut dans ce chapitre).
SPACE n	Ce paramètre provoque un saut de n lignes, n étant un entier positif dont la valeur maximum est 255. La variable système $LINE est incrémentée de cette valeur. La variable système $PAGE n'est pas affectée. Si n est omis, la valeur implicite est 1 et SPACE génère une ligne blanche. Si le nombre de lignes de la page courante est dépassé, le résultat est indéterminé.
SPACE TOP	Ce paramètre provoque un saut en haut d'une nouvelle page avant l'impression de la ligne suivante. La variable système $LINE est remise à zéro et la variable système $PAGE est incrémentée de un.

EXEMPLE :

```
REPORT FOOTING SPACE 1
               LINE "BLOC FIN D'ETAT"
               SPACE 2
               LINE "CET ETAT A ETE COMPILE PAR", @USER
               SPACE 2
               LINE "DATE", @DATE
```

❑

Cette fin d'état comporte 3 lignes de texte et 5 lignes blanches.

8.4.14 REPORT HEADING

Fonction :

Spécification du format et du contenu de l'en-tête d'état.

Format :

```
 ------------------------------------------------------------
|                {[LINE] <liste-descripteurs-éléments>}      |
| REPORT HEADING {SPACE <n>                            }     |
|                {SPACE TOP                            }     |
|                                                            |
|                {LINE <liste-descripteurs-éléments>}        |
|              [ {SPACE <n>                          } ] ... |
|                {SPACE TOP                            }     |
 ------------------------------------------------------------
```

Description de la clause :

Cette clause permet de spécifier le format et le contenu de l'en-tête d'état. Cet en-tête est imprimé une seule fois au début de l'état. Il commence toujours sur une nouvelle page et se termine par un saut à la page suivante ; cette opération s'effectue automatiquement sans demande explicite de l'utilisateur. A l'intérieur de l'en-tête d'état, les sauts de lignes sont commandés par l'utilisateur. SPACE n permet de sauter n lignes et SPACE TOP de passer à une nouvelle page. Le contrôle de compatibilité entre le nombre de lignes imprimées/sautées et le nombre maximum de lignes par page incombe à l'utilisateur. L'en-tête d'état peut s'étendre sur plusieurs pages si nécessaire.

La longueur des lignes ne doit pas dépasser la largeur de page (voir clause PAGE WIDTH). L'en-tête d'état peut contenir des informations constantes (chaîne de caractères) et/ou des informations variables (variables système).

Description des paramètres

LINE	Ce paramètre spécifie que les éléments qui suivent (jusqu'au prochain paramètre LINE, s'il y en a un) constituent une ligne. Si l'en-tête ne comporte qu'une ligne, LINE peut être omis.
liste-descripteurs-éléments	Cette liste spécifie les zones constitutives de la ligne. Pour une description détaillée, voir "Liste-descripteurs-éléments" (plus haut dans ce chapitre).
SPACE n	Ce paramètre provoque un saut de n lignes, n étant un entier positif dont la valeur maximum est 255. La variable système $LINE est incrémentée de cette valeur. La variable système $PAGE n'est pas affectée. Si n est omis, la valeur implicite est 1 et SPACE génère une ligne blanche. Si le nombre de lignes de la page courante est dépassé, le résultat est indéterminé.
SPACE TOP	Ce paramètre provoque un saut en haut d'une nouvelle page avant l'impression de la ligne suivante. La variable système $LINE est remise à zéro et la variable système $PAGE est incrémentée de un.

EXEMPLE :

```
REPORT HEADING LINE "EN-TETE D'ETAT"

LINE "LA LIGNE SUIVANTE CONTIENT DES INFORMATIONS VARIABLES"

SPACE 2

LINE @USER, @DATE
❑
```

Cet en-tête d'état contient 2 lignes de texte, puis deux lignes blanches et enfin une ligne comportant des informations variables.
Il est imprimé sur une nouvelle page avec saut automatique à la page suivante après la dernière ligne.

8.4.15 STARTING COLUMN NUMBER

Fonction :

Spécification du numéro de colonne sur lequel cadrer l'état à gauche.

Format :

```
 ------------------------------
|  STARTING COLUMN NUMBER <n>  |
 ------------------------------
```

Description du paramètre :

Numéro de la colonne sur lequel l'état doit être cadré à gauche.

EXEMPLE :

```
STARTING COLUMN NUMBER 3
❑
```

Cadrage à gauche de l'état sur la troisième colonne.

8.4.16 STARTING PAGE DETAIL

Fonction :

Spécification du numéro à attribuer à la première page détail de l'état.

Format :

```
 ---------------------------
|  STARTING PAGE DETAIL <n>  |
 ---------------------------
```

Description de la clause :

Cette clause permet de spécifier le numéro de la première page de lignes détail de l'état.

La valeur implicite est le numéro de la première page suivant l'en-tête d'état.

Description du paramètre :

Numéro à affecter à la première page détail d'un état.

EXEMPLE :

```
STARTING PAGE DETAIL 1
```
❑

La première page détail de l'état porte le numéro 1.

8.4.17 STARTING PAGE NUMBER

Fonction :

Spécification du numéro de la première page de l'état.

Format :

```
 ------------------------------
|                        { 1 }  |
|  STARTING PAGE NUMBER {   }  |
|                        {<n>}  |
 ------------------------------
```

Description de la clause :

Cette clause permet de spécifier le numéro de la première page de l'état. Si une valeur est affectée à la variable système $PAGE dans une requête, cette valeur prime sur celle spécifiée par la clause STARTING PAGE NUMBER dans la description d'état.

Description du paramètre :

Numéro de la première page de l'état, sous la forme d'un entier de six chiffres maximum.

La valeur implicite est 1.

EXEMPLE :

```
STARTING PAGE NUMBER 10
❑
```

La première page de l'état porte le numéro 10.

9. Macros IQS

9.1 Généralités

Ce chapitre décrit la création et l'exploitation des macros IQS. En principe, l'utilisateur doit déjà connaître IQS et être capable de comprendre et de créer des requêtes simples.

Une macro IQS comprend :

- une spécification d'appel (définition de macro) et
- un texte (appel de macro)

Pour être utilisable, une macro IQS doit être compilée et sauvegardée dans une bibliothèque.

Elle peut être appelée depuis une requête ou depuis une autre macro ; le processeur de macros IQS remplace alors l'appel de macro par le texte de la macro, les paramètres formels du texte étant remplacés par les valeurs fournies dans l'appel.

La première partie du chapitre fournit un certain nombre d'exemples illustrant la création et l'utilisation de macros simples.
La seconde partie décrit le format général des définitions et des appels de macro.

REMARQUE :

Dans les exemples, un astérisque placé devant un numéro de ligne indique que la ligne provoquera une erreur à la compilation de la requête, ou bien qu'elle constitue un développement incorrect de la macro. Pour plus de clarté, les messages de guidage et les séquences d'échappement d'édition ont été omis.

Les six commandes relatives aux macros IQS sont décrites en détail dans le volume 1 (77UR). Leur fonction est brièvement rappelée ci-dessous :

COMPILE MACRO	Compilation d'une macro. Elle est similaire à la commande COMPILE QUERY (qui s'applique uniquement aux requêtes).
SAVE MACRO	Sauvegarde d'une macro dans une bibliothèque. Elle est similaire à la commande SAVE QUERY (qui s'applique uniquement aux requêtes).
LIST MACRO	Listage d'une macro. Elle est similaire à la commande LIST QUERY (qui s'applique uniquement aux requêtes).
RECOMPILE MACRO	Recompilation d'une macro. Elle est similaire à la commande RECOMPILE QUERY (qui s'applique uniquement aux requêtes).
REPLACE MACRO	Remplacement d'une macro par une autre. Elle est similaire à la commande REPLACE QUERY (qui s'applique uniquement aux requêtes).
DROP MACRO	Suppression d'une macro. Elle est similaire à la commande DROP QUERY (qui s'applique uniquement aux requêtes).

Lorsque le paramètre EXPLIST est utilisé dans les commandes COMPILE QUERY et GO (qui toutes deux compilent des requêtes), l'utilisateur obtient un listage de la requête avec développement des macros appelées.

9.2 Exemples

9.2.1 Macro simple sans paramètres

Tout texte d'usage fréquent peut être transformé en macro.

Un simple appel de la macro permet ensuite d'appeler le texte considéré sans avoir à le saisir à chaque fois.

Par exemple, lors de l'exploitation des schémas de l'annexe A, "RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT" est fréquemment utilisé. Pour en faire une macro, il suffit d'introduire dans l'espace de travail origine, les lignes suivantes au moyen des fonctions d'édition usuelles :

```
10     :  &EXP
20     :  RETRIEVE UNIVERSITE, COLLEGE, ETUDIANT
```

La ligne 10 comporte la chaîne de caractères spéciale &EXP qui indique le début d'un texte de macro. Dans cet exemple, le texte de la macro se compose uniquement de la ligne 20. Il faut ensuite exécuter les commandes suivantes :

```
COMPILE MACRO
SAVE MACRO REC-COMPLETE
```

La première commande compile la macro et la seconde la sauvegarde sous le nom REC-COMPLETE (pour recherche complète). La macro peut alors être utilisée dans une requête, comme suit :

```
10 :  %REC-COMPLETE
20 :    PRINT E-NOM
30 :  END
```

Lorsque cette requête est compilée, la ligne 10 (appel de la macro) est remplacée par le texte de la macro. Si la requête est compilée et exécutée au moyen de la commande :

```
GO EXPLIST
```

elle est listée avec la macro développée (puisque le paramètre EXPLIST est spécifié) :

```
Début de %REC-COMPLETE
    20     RETRIEVE UNIVERSITE,
    20              COLLEGE,
    20              ETUDIANT
Fin de %REC-COMPLETE
    20     PRINT E-NOM
    30  END
```

Le listage est suivi des noms des étudiants fournis par l'exécution de la requête.

La commande GO n'ayant pas spécifié de nom de requête, c'est le contenu de l'espace de travail origine qui a été compilé et exécuté. Si celui-ci avait été sauvegardé entre temps au moyen de la commande SAVE QUERY TEST-REC-COMPLETE par exemple, il aurait fallu, pour compiler et exécuter la requête, utiliser la commande GO TEST-REC-COMPLETE,EXPLIST.

A noter qu'au listage, la macro développée est précédée et suivie de son nom et que les numéros de ligne sont ceux de sa définition.

Dans une requête, un appel de macro ne constitue pas nécessairement une instruction complète. La macro définie ci-dessus pourrait, par exemple, être employée comme suit :

```
10     :  %REC-COMPLETE WHERE GRADE = "LICENCE"
20     :    PRINT E-NOM
30     :  END
```

La ligne 10 contient l'appel de macro mais l'instruction est complétée par une clause WHERE, ce qui, après compilation (par une commande COMPILE QUERY EXPLIST par exemple) donne :

```
Début de %REC-COMPLETE
    20     RETRIEVE UNIVERSITE,
    20              COLLEGE,
    20              ETUDIANT
Fin de %REC-COMPLETE
    10              WHERE GRADE = "LICENCE"
    20         PRINT E-NOM
    30   END
```

L'exécution de cette requête fournit la liste des étudiants dont le grade est LICENCE.

9.2.2 Macro comportant un paramètre

La requête :

```
RETRIEVE ETUDIANT WHERE E-NOM = "WAGNER"
  PRINT GRADE
END
```

fournit l'article ETUDIANT dont le nom est WAGNER et imprime son grade, MAITRISE. La transformer en macro sans paramètres est sans intérêt puisqu'il est plus simple de la compiler et de la sauvegarder directement en tant que requête. Cependant, si elle est employée fréquemment pour impression d'une zone différente à chaque fois, il faut la modifier à chaque utilisation.
Il devient alors intéressant d'en faire une macro dans laquelle le nom de la zone à imprimer est remplacé par un paramètre formel :

```
10     :  &
20     :  &EXP
30     :  RETRIEVE ETUDIANT WHERE E-NOM = "WAGNER"
40     :  PRINT &1
50     :  END
```

Numéro de ligne	Explication
10	Il s'agit de la spécification d'appel de la macro. Elle contient un seul "et commercial" (&) qui indique que la macro ne comporte qu'un paramètre formel. Celui-ci sera remplacé par un seul mot IQS, un séparateur ou un nom par exemple.
20	La chaîne &EXP sépare la spécification d'appel du texte de la macro.
30 à 50	Ces trois lignes constituent le texte de la macro. Dans ce cas, il s'agit d'une requête complète.
40	&1 est un **paramètre formel** qui sera remplacé par un paramètre réel lorsque la macro sera appelée.

Une fois ces lignes introduites dans l'espace de travail origine (au moyen de AUTO etc., comme pour une requête), les commandes suivantes doivent être exécutées :

```
COMPILE MACRO
SAVE MACRO I-ZONE
```

La macro est compilée et sauvegardée dans les bibliothèques origine et résultante sous le nom I-ZONE.

La macro peut maintenant être appelée par une requête sous la forme suivante par exemple :

```
10    :  %I-ZONE GRADE
```

%I-ZONE étant le nom de la macro et GRADE le paramètre réel. La macro I-ZONE étant à elle seule une requête complète, il n'est pas nécessaire d'introduire autre chose que l'appel de macro. Lorsque cette requête est compilée au moyen de la commande COMPILE QUERY EXPLIST, le listage obtenu comporte la macro développée, le paramètre formel étant remplacé par le paramètre réel :

```
Début de %I-ZONE
   30      RETRIEVE ETUDIANT
   30               WHERE E-NOM = "WAGNER"
   40        PRINT GRADE
   50      END
Fin de %I-ZONE
```

Si l'appel de macro avait été %I-ZONE FRAIS-SCOL, le résultat aurait été le même si ce n'est que GRADE aurait été remplacé par FRAIS-SCOL, comme suit :

```
Début de %I-ZONE
   30      RETRIEVE ETUDIANT
   30               WHERE E-NOM = "WAGNER"
   40        PRINT FRAIS-SCOL
   50      END
Fin de %I-ZONE
```

Il aurait également été possible d'introduire :

```
10     :  %I-ZONE GRADE
20     :  %I-ZONE FRAIS-SCOL
```

ce qui aurait donné :

```
Début de %I-ZONE
   30      RETRIEVE ETUDIANT
   30               WHERE E-NOM = "WAGNER"
   40        PRINT GRADE
   50      END
Fin de %I-ZONE

Début de %I-ZONE
   30      RETRIEVE ETUDIANT
   30               WHERE E-NOM = "WAGNER"
   40        PRINT FRAIS-SCOL
   50      END
Fin de %I-ZONE
```

Cette requête serait peu performante puisque le fichier devrait être lu deux fois de suite. Son exécution fournirait :

```
MAITRISE
       1900
```

9.2.3 Macro avec paramètre à séparateur spécifique

Dans l'exemple précédent, la macro I-ZONE avait été définie avec un paramètre sans séparateur spécifique ; le paramètre était donc considéré comme terminé au premier séparateur rencontré. Par conséquent, la compilation de la requête :

```
* 10  : %I-ZONE GRADE,FRAIS-SCOL
```

dans laquelle figure le séparateur virgule, aurait donné :

```
Début de %I-ZONE
   30      RETRIEVE ETUDIANT
   30               WHERE E-NOM = "WAGNER"
   40        PRINT GRADE
   50      END
Fin de %I-ZONE
*  10      ,FRAIS-SCOL
```

et la dernière ligne ",FRAIS-SCOL", aurait constitué une erreur. A noter que l'espace étant un séparateur, la requête suivante :

```
* 10  : %I-ZONE E-NOM SHORT
```

aurait également été incorrecte puisqu'après compilation, la dernière ligne obtenue aurait été :

```
* 10 SHORT
```

Pour résoudre ce problème, il suffit de spécifier dans la définition de macro un séparateur spécifique pour le paramètre :

```
10     :  &;
20     :  &EXP
30     :  RETRIEVE ETUDIANT WHERE E-NOM = "WAGNER"
40     :   PRINT &1
50     :  END
```

Le paramètre défini par la ligne 10 ne sera considéré comme terminé que lorsqu'un point-virgule sera rencontré. En supposant que cette macro ait été compilée et sauvegardée sous le nom IMP-ART (pour impression article), la compilation de la requête :

```
    10     :  %IMP-ART E-NOM SHORT,FRAIS-SCOL;
```

produit :

```
Début de %IMP-ART
   30      RETRIEVE ETUDIANT
   30               WHERE E-NOM = "WAGNER"
   40        PRINT E-NOM SHORT,
   40              FRAIS-SCOL
   50      END
Fin de %IMP-ART
```

Les espaces contigus aux séparateurs étant ignorés par IQS, la première ligne de la définition de la macro pourrait également être :

```
10 : & ;
```

La requête ci-dessus, qui ne comporte que l'appel de la macro IMP-ART, donne accès uniquement à l'article ETUDIANT. Il est possible d'avoir accès à d'autres articles de la même hiérarchie en appelant la macro de l'intérieur d'une autre instruction RETRIEVE, ce qui permet de fournir comme paramètres réels les noms de zones de ces articles :

```
10     :  RETRIEVE UNIVERSITE, COLLEGE
20     :   PRINT U-NOM
30     :   %IMP-ART C-NOM COL 10, E-NOM, FRAIS-SCOL ;
40     :  END
```

ce qui donne, après compilation :

```
   10   : RETRIEVE UNIVERSITE,
   10   :          COLLEGE
   20   :   PRINT U-NOM
Début de %IMP-ART
   30       RETRIEVE ETUDIANT
   30                WHERE E-NOM = "WAGNER"
   40         PRINT C-NOM COL 10,
   40               E-NOM,
   40               FRAIS-SCOL
   50       END
Fin de %IMP-ART
   40     END
```

9.2.4 Macro simple emboîtée

Lorsque la requête qui appelle la macro IMP-ART ne comporte que cet appel, seules les zones de ETUDIANT peuvent être fournies comme paramètres réels. Sinon, il se produit une erreur à la compilation. Pour pouvoir fournir comme paramètres des zones de UNIVERSITE et COLLEGE, il existe deux solutions. La première a été vue à la fin de l'exemple précédent. La seconde consiste à créer une nouvelle macro :

```
10     :  &;
20     :  &EXP
30     :  %REC-COMPLETE WHERE E-NOM = "WAGNER"
40     :   PRINT &1
50     :  END
```

Cette macro, qui contient l'appel de la macro REC-COMPLETE, est compilée et sauvegardée sous le nom IMP-TOUT (pour imprimer tout). La compilation de la requête :

```
10 : %IMP-TOUT U-NOM, C-NOM, FRAIS-SCOL ;
```

produit :

```
Début de %IMP-TOUT
Début de %REC-COMPLETE
   20     RETRIEVE UNIVERSITE,
   20              COLLEGE
   20              ETUDIANT
Fin de %REC-COMPLETE
   30             WHERE E-NOM = "WAGNER"
   40       PRINT U-NOM,
   40             C-NOM,
   40             FRAIS-SCOL
   50     END
Fin de %IMP-TOUT
```

Une macro ne peut être appelée que si elle a été préalablement compilée et sauvegardée.

9.2.5 Spécification d'appel simple avec plusieurs paramètres

Les exemples proposés jusqu'ici ne contenaient qu'un seul paramètre, mais dans la pratique le nombre de paramètres d'une macro n'est pas limité. Chaque paramètre doit être représenté dans la spécification d'appel par un "et commercial" (&) qui peut être précédé et suivi d'une chaîne de caractères séparateurs. Dans les macros IMP-ART et IMP-TOUT, le point virgule était utilisé comme séparateur après le paramètre. Toute chaîne de caractères est autorisée à l'exception du mot EXP juste après un "et commercial" (puisque l'ensemble &EXP serait interprété comme le symbole début de texte), des mots PIC et PICTURE et du caractère guillemet (").

Soit la macro suivante compilée et sauvegardée sous le nom I-CS (pour impression courte simple) :

```
10     :  & & &
20     :  &EXP
30     :  PRINT &1 SHORT, &2 SHORT, &3 SHORT
```

La ligne 10 spécifie trois paramètres, chacun d'entre eux devant être un seul mot IQS. Elle aurait également pu être écrite : &&&. Dans ce cas, les séparateurs de paramètres étant omis, l'appel de macro %I-CS A, B, C serait interprété comme suit : A serait considéré comme le premier paramètre, la virgule comme le deuxième et B comme le troisième, ", C" étant considéré comme du texte. Dans l'exemple d'appel fourni plus loin, les trois paramètres réels sont des identificateurs et sont séparés par des espaces (qui ne seront pas pris en compte) ; l'appel est donc valide.

La ligne 20 indique le début du texte de la macro. La chaîne &EXP se compose en fait de deux mots IQS (& et EXP) qui sont en général accolés et placés sur une ligne séparée pour plus de clarté, mais la macro I-CS pourrait également être définie comme suit :

```
10     :  &&&& EXP PRINT &1 SHORT, & 2 SHORT, &3 SHORT
```

La ligne 30 comporte le texte de la macro dans lequel figurent les paramètres formels sous la forme &n, n étant un entier. Un paramètre formel se compose en fait de deux mots IQS ; & et n peuvent donc être séparés par des espaces. Le chiffre n peut comporter des zéros à gauche. Lorsque la macro est développée, chaque paramètre formel est remplacé par le paramètre réel correspondant : &1 est remplacé par le premier paramètre, &2 par le deuxième, etc. ... Un même paramètre formel peut figurer plusieurs fois dans le texte de la macro.
La macro I-CS pourrait par exemple être appelée par la requête suivante :

```
10     :  RETRIEVE ETUDIANT
20     :   %I-CS E-NOM GRADE FRAIS-SCOL
30     :  END
```

ce qui donne après compilation :

```
   10   : RETRIEVE ETUDIANT
Début de %I-CS
   30        PRINT E-NOM SHORT,
   30              GRADE SHORT,
   30              FRAIS-SCOL SHORT
Fin de %I-CS
   30     END
```

9.2.6 Spécification d'appel plus complexe

Dans l'exemple précédent, la spécification d'appel comportait uniquement des "et commerciaux" ; dans un autre exemple, le paramètre était défini avec un point-virgule comme séparateur. Soit maintenant la définition suivante :

```
10     :  & AND &;
20     :  &EXP
30     :  PRINT &1 SHORT, &2
```

Cette macro est compilée et sauvegardée sous le nom IMP-CT (pour impression courte). Elle comporte une spécification d'appel plus complexe : là encore, le dernier paramètre est suivi d'un point-virgule mais cette fois-ci, les deux paramètres sont séparés par la chaîne de caractères "AND".

Lorsque la macro est appelée, la spécification d'appel doit être copiée exactement : les "et commerciaux" doivent être remplacés par des paramètres réels et les autres caractères reproduits tels quels. Par exemple, la requête :

```
10     :  RETRIEVE ETUDIANT
20     :   %IMP-CT E-NOM AND GRADE ;
30     :  END
```

donne après compilation :

```
   10    : RETRIEVE ETUDIANT
Début de %IMP-CT
   30         PRINT E-NOM SHORT,
   30               GRADE
Fin de %IMP-CT
   30      END
```

Les règles usuelles concernant les espaces s'appliquent à la fois à la spécification d'appel et à l'appel lui-même. La définition de la macro pourrait donc être :

```
10     :  &AND&;&EXP PRINT&1 SHORT,&2
```

L'espace après EXP est obligatoire dans la définition, de même que les espaces avant et après AND (en ligne 20) dans l'appel correspondant.

9.2.7 Transfert de paramètres à une macro emboîtée

Cet exemple illustre l'utilisation de paramètres formels à la place de paramètres réels dans l'appel d'une macro emboîtée. Il donne des informations complémentaires sur les spécifications d'appel.

Soit la macro suivante compilée et sauvegardée sous le nom REC-I (pour recherche et impression) :

```
10     :  (&,&,&)
20     :  &EXP
30     :  RETRIEVE &1
40     :    %IMP-CT &2 AND &3;
50     :  END
```

La spécification d'appel commençant par une parenthèse gauche, le premier paramètre de l'appel de la macro doit également être précédé d'une parenthèse gauche

```
10     :  %REC-I (ETUDIANT, E-NOM, GRADE)
```

ce qui donne, après compilation :

```
Début de %REC-I
   30      RETRIEVE ETUDIANT
Début de %IMP-CT
   30         PRINT E-NOM SHORT,
   30               GRADE
Fin de %IMP-CT
   50      END
Fin de %REC-I
```

Le second paramètre étant suivi d'une virgule dans la spécification d'appel, le paramètre réel correspondant ne peut se composer que de mots IQS non séparés par des virgules. Par contre, le troisième paramètre réel peut comporter des virgules puisque son séparateur est une parenthèse droite.

L'appel de macro :

```
10     :  %REC-I (ETUDIANT,E-NOM,"GRADE=",GRADE)
```

est donc valide. La valeur du deuxième paramètre est E-NOM et celle du troisième "GRADE=",GRADE ce qui donne après compilation :

```
Début de %REC-I
   30      RETRIEVE ETUDIANT
Début de %IMP-CT
   30         PRINT E-NOM SHORT,
   30               "GRADE=",
                  GRADE
Fin de %IMP-CT
   50      END
Fin de %REC-I
```

9.2.8 Utilisation simple d'un paramètre formel protégé

Pour être protégé, un paramètre formel doit être seul entre guillemets. Les chaînes protégées sont traitées de manière spéciale par le système. Soit la macro suivante compilée et sauvegardée sous le nom CPR (pour chaîne protégée) :

```
10 : (&/&)
20 : &EXP
30 : &1, "&2"
```

Lorsque cette macro est appelée par %CPR(E-NOM/E-NOM), elle est développée normalement comme suit : E-NOM, "E-NOM". Cependant, lorsque le paramètre réel correspondant au paramètre formel protégé contient des guillemets, le traitement est différent. Par exemple, l'appel :

```
%CPR("E-NOM"/ENTRE "GUILLEMETS")
```

ne donne pas :

```
* "E-NOM", "ENTRE "GUILLEMETS""
```

En effet, le paramètre formel étant protégé, le système vérifie si le paramètre réel correspondant contient des guillemets et, s'il en trouve, il les double afin de le rendre conforme aux règles concernant les chaînes protégées.
L'appel ci-dessus est donc développé comme suit :

```
"E-NOM", "ENTRE ""GUILLEMETS"""
```

les trois derniers guillemets comprenant le guillemet doublé de la chaîne plus le guillemet marquant la fin de la chaîne.

9.2.9 Appels de macro emboîtés et chaînes protégées

Soit la macro suivante compilée et sauvegardée sous le nom EMBOIT :

```
10     :  & &EXP &1 COL 10
```

L'appel :

```
%EMBOIT E-NOM
```

donne :

```
E-NOM COL 10
```

L'appel de la macro EMBOIT peut être utilisé comme paramètre réel dans un autre appel de macro, par exemple :

```
10     :  %IMP-ART %EMBOIT GRADE;
```

ce qui donne :

```
Début de %IMP-ART
   30      RETRIEVE ETUDIANT
   30               WHERE E-NOM = "WAGNER"
   40        PRINT
Début de %EMBOIT
   10               GRADE COL 10
Fin de %EMBOIT
   50      END
Fin de %IMP-ART
```

La macro IMP-ART est développée, la ligne 40 devenant PRINT %EMBOIT GRADE, puis la macro EMBOIT est développée ce qui donne le résultat ci-dessus.

De même, l'appel :

```
10     :  %IMP-ART %CPR(GRADE/GRADE);
```

donne :

```
Début de %IMP-ART
   30      RETRIEVE ETUDIANT
   30               WHERE E-NOM = "WAGNER"
   40        PRINT
Début de %CPR
   30               GRADE,
   30               "GRADE"
Fin de %CPR
   50      END
Fin de %IMP-ART
```

Dans les exemples ci-dessus, les appels de macro correspondent à des paramètres formels non protégés ; ceux-ci sont donc remplacés par le texte de la macro.

Lorsqu'ils correspondent à des paramètres formels protégés, ils ne sont pas développés, le système ne faisant que rechercher des guillemets dans les paramètres réels pour les doubler, si nécessaire.

L'exemple ci-dessous illustre ce cas :

```
10    : RETRIEVE ETUDIANT WHERE E-NOM = "WAGNER"
20    :   PRINT %CPR(%EMBOIT GRADE/%EMBOIT GRADE)
30    : END
```

ce qui donne :

```
   10      RETRIEVE ETUDIANT
   10               WHERE E-NOM = "WAGNER"
   20         PRINT
Début de %CPR
Début de %EMBOIT
   10               GRADE COL 10
Fin de %EMBOIT
   30               ,"%EMBOIT GRADE"
Fin de %CPR
   30      END
```

L'appel de la macro EMBOIT constituant le premier paramètre réel est développé alors que celui constituant le deuxième ne l'est pas puisque, dans la définition de la macro CPR, le deuxième paramètre formel est protégé (c'est-à-dire entre guillemets). Le même phénomène se produit lorsqu'un paramètre contient des guillemets. Par exemple :

```
10    : RETRIEVE ETUDIANT WHERE E-NOM = "WAGNER"
20    :   PRINT %CPR(%EMBOIT "GRADE"/%EMBOIT "GRADE")
30    : END
```

ce qui donne :

```
   10      RETRIEVE ETUDIANT
   10               WHERE E-NOM = "WAGNER"
   20         PRINT
Début de %CPR
Début de %EMBOIT
   10               "GRADE" COL 10
Fin de %EMBOIT
   30               ,"%EMBOIT ""GRADE"""
Fin de %CPR
   30      END
```

Là encore, l'appel de macro constituant le premier paramètre est développé alors que celui constituant le deuxième ne l'est pas puisque le système ne fait qu'ajouter les guillemets nécessaires.

9.2.10 Cas complexe

Soit la macro suivante compilée et sauvegardée sous le nom VISU-ETUDIANT :

```
10     :  & &;
20     :  &EXP
30     :  PRINT "&1", "DES ETUDIANTS", "&2"
40     :  RETRIEVE ETUDIANT WHERE &2
50     :   PRINT &1 COL 5
60     :  END
```

La compilation de l'appel de macro :

```
10     :  %VISU-ETUDIANT E-NOM GRADE="LICENCE";
```

donne :

```
Début de %VISU-ETUDIANT
   30      PRINT "E-NOM",
   30            "DES ETUDIANTS",
   30            "GRADE=""LICENCE"""
   40      RETRIEVE ETUDIANT
   40               WHERE GRADE="LICENCE"
   50        PRINT E-NOM COL 5
   60      END
Fin de %VISU-ETUDIANT
```

Cet exemple illustre le traitement des paramètres formels protégés. Le second paramètre réel contient des guillemets : ceux-ci sont doublés lorsqu'il remplace le paramètre formel protégé et ne le sont pas lorsqu'il remplace le paramètre formel non protégé.

Soit maintenant la macro suivante compilée et sauvegardée sous le nom NE-ENTRE :

```
10  :  & AND &
20  :  &EXP
30  : DATE-NAISSANCE >= "&1" AND DATE-NAISSANCE <= "&2"
```

L'appel :

```
* %NE-ENTRE 580101 AND 591231
```

donnerait :

```
* DATE-NAISSANCE >= "580101" AND DATE-NAISSANCE <= "591231"
```

Considérons maintenant l'appel :

```
10  : %VISU-ETUDIANT E-NOM %NE-ENTRE 580101 AND 591231;
```

Le traitement du premier paramètre réel E-NOM est simple : il remplace directement chaque paramètre formel &1.

Le second paramètre réel est l'appel de la macro NE-ENTRE. Le premier paramètre formel auquel il correspond est entre guillemets : le système recherche donc des guillemets (inexistants) dans le paramètre réel, c'est-à-dire dans l'appel de macro, mais il ne le développe pas. Le second paramètre formel n'est pas protégé : l'appel de la macro NE-ENTRE est donc développé.

Lorsqu'un paramètre réel remplace un paramètre formel protégé, le système recherche uniquement des guillemets puis passe au premier mot suivant le paramètre formel. Lorsqu'il remplace un paramètre formel non protégé, il est traité normalement : par conséquent, s'il s'agit d'un appel de macro, celui-ci est développé.

L'appel ci-dessus donne donc, après compilation :

```
Début de %VISU-ETUDIANT
   30      PRINT "E-NOM",
   30            "DES ETUDIANTS",
   30            "%NE-ENTRE 580101 AND 591231"
   40      RETRIEVE ETUDIANT
   40               WHERE
Début de %NE-ENTRE
   20                   DATE-NAISSANCE >= "580101"
   20                   AND DATE-NAISSANCE <= "591231"
Fin de %NE-ENTRE
   50        PRINT E-NOM COL 5
   60      END
Fin de %VISU-ETUDIANT
```

L'exécution de cette requête produit :

```
E-NOM DES ETUDIANTS %NE-ENTRE 580101 AND 591231
    SERREAU
    MARTY
```

9.2.11 Définition d'un paramètre facultatif

L'exemple ci-dessous dérive des exemples fournis aux paragraphes 9.2.8 et 9.2.9 (définition et utilisation de la macro CPR).

La macro CPR a été définie comme suit :

```
(&/&) &EXP &1, "&2"
```

et utilisée au paragraphe 9.2.9 de la manière suivante :

```
%CPR (GRADE/GRADE)
```

Si le second paramètre n'est pas indispensable, il est possible de le déclarer comme étant facultatif au moyen d'une définition de la forme suivante :

```
10     :  (&/"&") &EXP &1 "&2"
```

Supposons cette macro compilée et sauvegardée sous le nom PARFAC (pour paramètre facultatif). L'appel :

```
10    : RETRIEVE ETUDIANT
20    :     PRINT %PARFAC (GRADE/GRADE)
30    : END
```

donne :

```
   10      RETRIEVE ETUDIANT
   20          PRINT
Début de %PARFAC
   10               GRADE
   10               "GRADE"
Fin de %PARFAC
   30      END
```

(Le développement de cette macro est presque identique à celui de la macro CPR).

REMARQUE :

L'absence de virgule dans l'instruction PRINT ne constitue pas une erreur puisque les virgules ne sont obligatoires que dans les instructions DEFINE et PARAMETER.

Un appel de la macro PARFAC dans lequel le second paramètre réel est omis, par exemple :

```
10    : RETRIEVE ETUDIANT
20    :   PRINT %PARFAC (GRADE/)
30    : END
```

donne :

```
   10      RETRIEVE ETUDIANT
   20        PRINT
Début de %PARFAC
   10               GRADE
Fin de %PARFAC
   30      END
```

La déclaration d'un paramètre facultatif se fait dans la spécification d'appel en mettant le & correspondant entre guillemets ; il est obligatoire de définir un séparateur propre à un tel paramètre. L'utilisation d'un paramètre facultatif est semblable à celle des autres paramètres formels.

9.2.12 Définition de paramètres facultatifs avec valeurs implicites

Dans l'exemple précédent (macro PARFAC), le second paramètre était facultatif. L'utilisateur peut vouloir lui donner une valeur implicite, c'est-à-dire une valeur à utiliser lorsqu'il est omis. Cette valeur est déclarée dans le texte de la macro aux endroits où figure le paramètre facultatif.

Par exemple, dans la macro PARFAC, l'utilisateur peut vouloir imprimer le premier paramètre sous forme de chaîne protégée à la place du deuxième paramètre lorsque ce dernier est omis. Pour ce faire, il peut compiler et sauvegarder, sous le nom VALIMP (pour valeur implicite) par exemple, la macro suivante :

```
10 : (&/"&") &EXP &1 "&2":"&1":
```

L'appel :

```
10    : %IMP-ART %VALIMP (E-NOM/) ;
```

donne :

```
Début de %IMP-ART
   30      RETRIEVE ETUDIANT
   30               WHERE E-NOM = "WAGNER"
   40         PRINT
Début de %VALIMP
   30               E-NOM
   30               "E-NOM"
Fin de %VALIMP
   50      END
Fin de %IMP-ART
```

Dans la spécification d'appel, le second paramètre a été déclaré comme étant facultatif (& entre guillemets). Dans le texte de la macro, la valeur implicite qui lui a été attribuée est celle du premier paramètre entre guillemets. La valeur implicite est signalée par les deux points (:) qui la précèdent et la suivent. Toute séquence interprétable par IQS peut être utilisée comme valeur implicite à condition toutefois qu'elle ne contienne pas de référence à un paramètre facultatif.

EXEMPLE :

Soit la macro suivante compilée et sauvegardée sous le nom IM-ZONES (impression zones) :

```
10     :  LIST="&",FROM=&,WHERE="&";
20     :  &EXP
30     :  RETRIEVE &2
40     :    WHERE &3 :&2 PRESENT;
50     :    PRINT WITH TITLE &1
60     :  END
```

❑

Les premier et troisième paramètres sont facultatifs. Le premier n'a pas de valeur implicite. Le troisième a pour valeur implicite celle du deuxième paramètre réel suivie de la condition PRESENT. La condition PRESENT appliquée à la totalité d'un article sera vraisemblablement vérifiée.

L'appel :

```
10    : %IM-ZONES LIST=, FROM=ETUDIANT, WHERE=;
```

donne :

```
Début de %IM-ZONES
   30      RETRIEVE ETUDIANT
   40               WHERE ETUDIANT PRESENT
   50         PRINT WITH TITLE
   60      END
Fin de %IM-ZONES
```

L'exécution de cette requête provoquera l'impression d'un état contenant toutes les zones de tous les articles ETUDIANT.

L'appel :

```
10     :  %IM-ZONES LIST=E-NOM FRAIS-SCOL,FROM=ETUDIANT
                   ,WHERE=E-NOM BEGINS "B";
```

donne :

```
Début de %IM-ZONES
   30      RETRIEVE ETUDIANT
   40               WHERE E-NOM BEGINS "B"
   50         PRINT WITH TITLE E-NOM
   50               FRAIS-SCOL
   60      END
Fin de %IM-ZONES
```

L'exécution de cette requête provoquera l'impression d'un état contenant les zones E-NOM et FRAIS-SCOL des étudiants dont le nom commence par B.

9.3 Définition de macro IQS

```
|-----------------------------------------------------------|
|[spécification-d'appel-de-macro] &EXP [texte-de-la-macro] |
|-----------------------------------------------------------|
```

- & et EXP sont deux mots distincts, bien qu'il soient généralement accolés et placés sur une ligne séparée.
- Une macro IQS doit avoir été compilée (au moyen de COMPILE MACRO) et sauvegardée (au moyen de SAVE MACRO ou REPLACE MACRO) pour pouvoir être appelée.

9.3.1 Spécification d'appel de macro

```
 -------------------------------------
| {séparateur-de-paramètre     }      |
| { &                          } ...  |
| { "&" séparateur-de-paramètre}      |
 -------------------------------------
```

- Chaque "et commercial" (&) indique la présence et la position d'un paramètre réel dans l'appel de macro.
- Un "et commercial" entre guillemets signifie que le paramètre réel correspondant peut être omis dans l'appel de macro (c'est-à-dire qu'il est facultatif). Il doit obligatoirement être suivi d'un séparateur.

9.3.2 Séparateur de paramètre

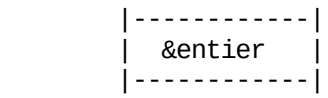
```
 ----------------------------------
|   { identificateur       }       |
|   { constante-numérique  }       |
|   { séparateur-1         }       |
| [ { séparateur-2         } ... ]  |
|   { séparateur-3         }       |
|   { élément-non-standard }       |
 ----------------------------------
```

- Un séparateur de paramètre est une chaîne de caractères qui ne contient ni "et commercial", ni guillemets.
- L'identificateur EXP ne doit pas être utilisé comme séparateur de paramètre juste après un "et commercial".
- Les mots PIC et PICTURE ne doivent pas être utilisés comme séparateurs de paramètre.

9.3.3 Texte de macro

- Le texte de la macro remplace l'appel de macro, les paramètres formels du texte étant remplacés par les paramètres réels fournis dans l'appel.
- Tout texte conforme à la syntaxe est admis à condition toutefois que la requête qui appelle la macro soit correcte après développement de celle-ci. A noter que le compilateur de macros IQS ne vérifie pas la syntaxe du texte.
- Les valeurs implicites ne peuvent être spécifiées que pour les paramètres facultatifs (entre guillemets dans la spécification d'appel). Elles sont précédées et suivies de deux points (:).

9.3.4 Paramètre formel



- La valeur de l'entier doit être supérieure ou égale à 1.
- L'entier indique quel est le paramètre réel correspondant dans l'appel de la macro : &1 correspond au premier paramètre, &2 au deuxième, etc.
- Un paramètre formel est remplacé, dans le texte de la macro, soit par le paramètre réel correspondant fourni dans l'appel de macro, soit par la valeur implicite éventuellement spécifiée s'il s'agit d'un paramètre facultatif et qu'il est omis dans l'appel.
- Une fois le remplacement effectué, le traitement du texte de la macro continue à partir du début du paramètre réel inséré. Si celui-ci est un appel de macro, il est donc développé.

9.3.5 Valeur implicite (paramètres facultatifs)

```
 -------------------------------
| {mot-IQS                   }  |
| {paramètre-formel          }  |
| {paramètre-formel-protégé  }  |
 -------------------------------
```

- La valeur implicite décrit le texte à générer lorsqu'un paramètre facultatif est omis dans l'appel de macro. A noter que la valeur implicite n'est pas obligatoire.
- Les paramètres formels utilisés dans la valeur implicite ne doivent pas avoir été déclarés comme étant facultatifs (entre guillemets) dans la spécification d'appel.
- Toute valeur implicite conforme à la syntaxe est admise à condition toutefois que la requête qui appelle la macro soit correcte après développement de celle-ci.
- A noter que la syntaxe du texte de la macro n'est pas vérifiée par le compilateur de macros IQS.

9.3.6 Paramètre formel protégé

```
|------------------------|
|   "paramètre-formel"   |
|------------------------|
```

- Un paramètre formel protégé est remplacé, dans le texte de la macro, soit par le paramètre réel correspondant fourni dans l'appel de macro après traitement, soit par la valeur implicite éventuellement spécifiée s'il s'agit d'un paramètre facultatif et qu'il est omis dans l'appel.
- Le traitement d'un paramètre réel correspondant à un paramètre formel protégé consiste à doubler les éventuels guillemets qu'il contient, puis à le mettre entre guillemets.
- Une fois le remplacement effectué, le traitement du texte de la macro continue à partir de la fin du paramètre réel inséré. Si celui-ci est un appel de macro, il n'est donc pas développé.

9.4 Appel de macro

```
 ------------------------------------------------
|                {appel-de-macro         }       |
| %nom-de-macro [ {séparateur-de-paramètre} ...]  |
|                {paramètre-réel         }       |
 ------------------------------------------------
```

- Les appels de macro emboîtés sont traités les premiers, avant les séparateurs de paramètre et les paramètres réels, à partir de l'appel le plus intérieur.
- Le développement d'un appel de macro consiste à le remplacer par le texte fourni dans la définition de la macro.
- Pour plus de détails sur le développement des appels de macro et le remplacement des paramètres, reportez-vous au paragraphe "Définition de macro IQS".

REMARQUE :

Un appel de macro constitue un seul mot IQS. Il permet d'insérer un texte écrit en langage de requêtes dans un autre appel de macro de niveau supérieur.

9.4.1 Séparateur de paramètre

```
 ----------------------------------
| {séparateur-de paramètre    }    |
| {&                          } ... |
| {"&" séparateur-de-paramètre}    |
 ----------------------------------
```

- Son format est identique à celui du séparateur de paramètre et de la spécification d'appel (voir "Définition de macro IQS").
- Les séparateurs de paramètre de l'appel de macro doivent correspondre exactement, en ce qui concerne l'ordre et le contenu, à ceux spécifiés dans la définition de la macro.
- L'absence de séparateur de paramètre dans une définition de macro signifie que le paramètre réel correspondant dans l'appel est constitué d'un seul mot IQS.

9.4.2 Paramètre réel

```
|----------------|
|  mot-IQS ...   |
|----------------|
```

- Un paramètre réel est un mot ou une suite de mots IQS ne contenant pas le séparateur de paramètre qui lui est propre.
- Un paramètre réel qui n'a pas été déclaré dans la spécification d'appel comme étant facultatif doit être présent et comporter au moins un mot IQS.
- Les paramètres réels sont représentés dans le texte de la macro par des paramètres formels. Ils sont numérotés en ordre croissant à partir de 1 : le premier paramètre a le numéro 1, le deuxième le numéro 2, etc. ...

10. Métabase IQS

10.1 Généralités

Le schéma de la métabase **H_METAIQS** est un dictionnaire de données qui répertorie tous les objets et éléments utilisés tels que les requêtes, aires, schémas, articles, descriptions d'état, etc. Les données sont automatiquement actualisées par IQS. Le contenu de la métabase est accessible en ligne au moyen d'instructions du langage de requêtes ou de commandes IQS.

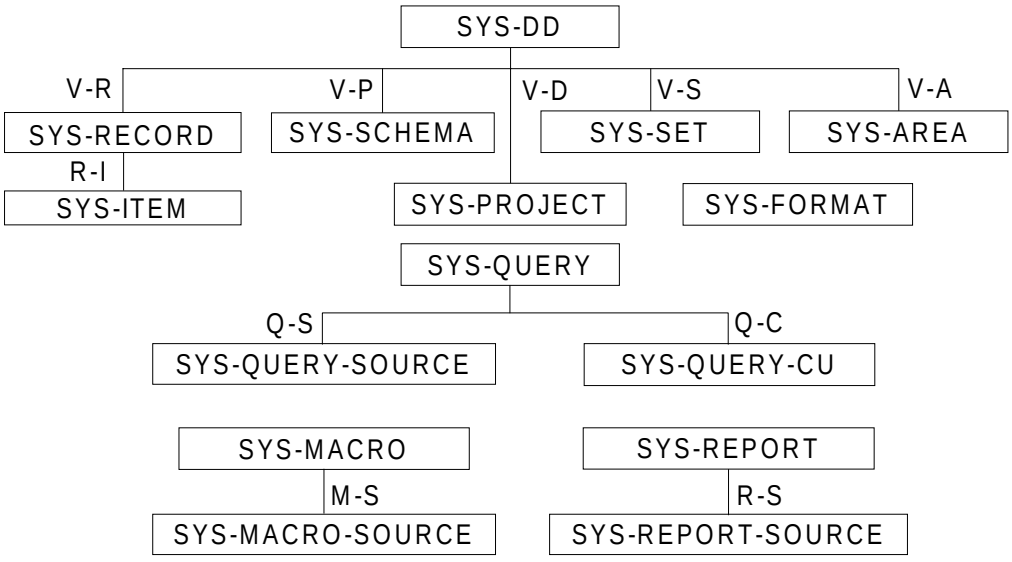


Figure 10-1. Structure du métaschéma H_METAIQS

En cas d'utilisation du langage procédural (instructions IQS), l'accès au schéma de la métabase dépend d'un certain nombre de critères détaillés dans le guide de l'administrateur IQS-V4.

Avec le langage non-procédural (commandes IQS), le schéma doit être sélectionné au moyen d'une commande SELECT et l'aire correspondante ouverte au moyen d'une commande OPEN. Il ne peut pas y avoir d'autre schéma courant.

La métabase est constituée des bibliothèques contenant les objets utilisés. Son contenu est actualisé en permanence.

Les objets décrits par la métabase sont :

- **les objets de description de données :** schémas, aires, vues, structures, articles, zones, ensembles, fichiers de travail, projets.
- **les objets de préparation de programmes :** requêtes origine et résultantes, macros origine et résultantes, descriptions d'état origine et résultantes, formats.

La mise à jour de la métabase s'effectue de manière indirecte au moyen de commandes IQS comme DEFINE/UPDATE/SAVE/REPLACE qui mettent à jour les objets qu'elle décrit. Toute tentative de mise à jour directe, au moyen des instructions INSERT ou MODIFY ou de la commande UPDATE, est rejetée par IQS.

L'accès aux articles de la métabase n'est possible que par l'intermédiaire de leurs articles maîtres. Il est donc impossible de passer de détail à maître.

10.2 Description des articles de la métabase

Article SYS-DD

L'article SYS-DD identifie les descriptions de données disponibles au cours de la session IQS. Chaque occurrence de cet article identifie une vue utilisateur définie au moyen du processeur de vues IQS, un schéma compilé par MNDD, puis par IQS, une structure définie au moyen du processeur de structures IQS ou un fichier créé au moyen de commandes IQS telles que EXTRACT, WRITE, SORT, MERGE ou des instructions du langage de requêtes WRITE et SORT.

L'accès direct est possible sur la clé DD-NAME.

DD-NAME	Identification de la description. Constitue la clé CALC de l'article. Les valeurs de clés identiques sont autorisées.
DD-DESCRIPTION	Commentaire explicatif.
DD-TYPE	Type de la description : **"VDD"** : vue **"SDD"** : schéma **"FDD"** : structure **"WDD"** : fichier de travail défini par les commandes WRITE, SORT, EXTRACT, MERGE ... ou par les instructions SORT et WRITE.
DD-DATE	Date de création de la description, sous la forme aammjj.
DD-TIME	Heure de création de la description, sous la forme hhmmsscc.
DD-VERSION	Version d'IQS sous laquelle la description a été créée. Les valeurs possibles sont : "2.2", "2.3", "3.1", "3.2", "4.0". Des numéros de version supplémentaires seront ajoutés si nécessaire.

DD-PRIVACY	Droits d'accès à la description (six sous-zones) :
DD-RETRIEVAL	Si la valeur est 1 : lecture autorisée.
DD-UPDATE	Si la valeur est 1 : mise à jour autorisée.
DD-IDS	Si la valeur est 1, il s'agit d'un article IDS/II.
DD-VERSIDS-V5	La valeur 0 identifie un article IDS/II-V2. La valeur 1 identifie un article IDS/II-V5.
DD-VIEW-DDL-V5	Processeur sous lequel le DDL a été compilé : Valeur 0 : DDLPROC. Valeur 1 : MAINTAIN_DD.
DD-ACCESS-RIGHT	Si la valeur est 0 : droits d'accès applicables. Si la valeur est 1 : pas de droits d'accès applicables.
NUMBER-OF-SCHEMAS	Pour une vue utilisateur, nombre de schémas sous-jacents.
NUMBER-OF-AREAS	Nombre d'aires (à spécifier uniquement si DD-TYPE a pour valeur "SDD" ou "FDD").
NUMBER-OF-RECORDS	Nombre d'articles.
NUMBER-OF-SETS	Nombre d'ensembles.

Article SYS-SCHEMA

L'article SYS-SCHEMA identifie les schémas sous-jacents à une vue utilisateur.

Sa présence n'a de sens que lorsque la valeur de DD-TYPE est "VDD".

SCHEMA-NAME	Identification du schéma.
SCHEMA-DATE	Date de création du schéma, sous la forme aammjj.
SCHEMA-TIME	Heure de création du schéma, sous la forme hhmmsscc.

Article SYS-AREA

L'article SYS-AREA identifie les aires constitutives d'un schéma ou sous-jacentes à une vue ou à une structure.

Sa présence n'a de sens que lorsque la valeur de DD-TYPE est "SDD" ou "FDD".

AREA-NAME	Identification de l'aire
AREA-ORG	Organisation de l'aire :
	"SEQ" : UFAS séquentiel ou sous-fichier bibliothèque
	"IND" : UFAS séquentiel indexé
	"INTG" : IDS/II
	"REL" : UFAS relatif

Article SYS-RECORD

L'article SYS-RECORD décrit les articles utilisés dans un schéma, une vue ou une structure : articles logiques pour une vue, articles réels dans les autres cas.

RECORD-NAME	Identification de l'article.
RECORD-DESCRIPTION	Brève description de l'article.
RECORD-PRIVACY	Droits d'accès à l'article (quatre sous-zones) :
RECORD-RETRIEVAL	Si la valeur est 1 : lecture autorisée.
RECORD-UPDATE	Si la valeur est 1 : mise à jour autorisée.
RECORD-INSERT	Si la valeur est 1 : insertion autorisée.
RECORD-DELETE	Si la valeur est 1 : suppression autorisée.
NUMBER-OF-ITEMS	Nombre de zones de l'article.

Article SYS-ITEM

L'article SYS-ITEM décrit les zones constituant un article. Son article maître est SYS-RECORD.

ITEM-NAME	Identification de la zone.
ITEM-DESCRIPTION	Brève description de la zone.
ITEM-TYPE	Type de la zone : **"UPKS"** : décimal éclaté signé **"UPK"** : décimal éclaté non signé **"PKS"** : décimal condensé signé **"PK2"** : décimal condensé-2 non signé **"PK"** : décimal condensé non signé **"BIN"** : binaire **"CHAR"** : caractère **"STR"** : structure **"GRP"** : groupe **"REC"** : article
ITEM-LENGTH	Longueur de la zone : - nombre de chiffres pour les zones de type décimal, - 15 ou 31 pour les zones de type binaire, - nombre d'octets pour les zones de type caractère, structure et article, - nombre d'octets d'une occurrence d'un groupe.
ITEM-SCALE	Nombre de chiffres après la virgule.
ITEM-OFFSET	Position relative de la zone par rapport au début de l'article ou au début du groupe répétitif.
ITEM-LEVEL	Niveau de la zone.
ITEM-NOCC	Nombre d'occurrences de la zone.
ITEM-NSUBS	Nombre de niveaux d'indiçage nécessaires pour qualifier la zone.
ITEM-CHECK	Clauses CHECK portant sur la zone (deux sous-zones) :
LOCAL-CHECK	Si la valeur est 1 : contrôle au niveau de la zone.
GLOBAL-CHECK	Si la valeur est 1 : contrôle au niveau de l'article.

ITEM-PRIVACY	Droits d'accès à la zone (deux sous-zones) :
ITEM-RETRIEVAL	Si la valeur est 1 : lecture autorisée.
ITEM-UPDATE	Si la valeur est 1 : mise à jour autorisée.

Article SYS-SET

L'article SYS-SET décrit les relations existant entre les articles d'une vue ou d'un schéma : relations logiques pour une vue utilisateur, relations physiques pour un schéma.

Sa présence n'a de sens que lorsque la valeur de DD-TYPE est "VDD" ou "SDD".

SET-NAME	Identification de l'ensemble.
OWNER-RECORD-NAME	Nom de l'article maître de l'ensemble.
MEMBER-RECORD-NAME	Nom de l'article détail de l'ensemble.
SET-KEY	Caractéristiques de la clé (5 sous-zones) :
SET-CALC	La valeur 1 indique une clé calculée.
SET-SORTED	La valeur 1 indique un ensemble trié.
SET-ASCENDING	La valeur 1 indique un ensemble trié en ordre croissant.
SET-DESCENDING	La valeur 1 indique un ensemble trié en ordre décroissant.
SET-NODUP	La valeur 1 signifie : valeurs de clé identiques interdites.

Article SYS-PROJECT

L'article SYS-PROJECT identifie un projet ayant le droit d'accéder à un schéma ou une vue (lorsque DD-TYPE a pour valeur "SDD" ou "VDD"). Son article maître est SYS-DD.

PROJECT-NAME	Nom de projet GCOS 7.
USER-NAME	Nom d'utilisateur GCOS 7.
ACCESS-TYPE	Type d'accès autorisé.

Article SYS-QUERY

L'article SYS-QUERY décrit les requêtes compilées disponibles au cours de la session. Il ne donne accès qu'aux requêtes se trouvant dans BINLIB. Des informations concernant la date et l'heure de création du schéma/de la vue sous-jacent(e) sont également fournis.

L'accès direct est possible sur la clé QUERY-NAME.

QUERY-NAME	Identification de la requête.
QUERY-DESCRIPTION	Brève description de la requête.
QUERY-DATE	Date de la dernière compilation, sous la forme aammjj.
QUERY-TIME	Heure de la dernière compilation, sous la forme hhmmsscc.
QUERY-VERSION	Numéro de version d'IQS. Les valeurs possibles sont : "2.2", "2.3", "3.1", "3.2", "4.0".
QUERY-VIEW	Nom de la vue à laquelle est associée la requête. Si elle est indépendante, cette zone est entièrement à espaces.
QUERY-VIEW-DATE	Date de création du schéma/de la vue sous-jacent(e), sous la forme aammjj.
QUERY-VIEW-TIME	Heure de création du schéma/de la vue sous-jacent(e), sous la forme hhmmsscc.

Article SYS-QUERY-SOURCE

L'article SYS-QUERY-SOURCE décrit chaque ligne de la version origine de la requête identifiée par SYS-QUERY.

Il ne donne accès qu'aux requêtes origine enregistrées dans SLLIB.

QUERY-LINE-LENGTH	Longueur de la ligne origine.
QUERY-LINE-NUMBER	Numéro de la ligne origine.
QUERY-LINE	Contenu de la ligne origine.

Article SYS-QUERY-CU

L'article SYS-QUERY-CU fournit la date et l'heure de création de l'unité compilée utilisable sous TDS (si elle existe), correspondant à la requête identifiée par SYS-QUERY.

Il ne donne accès qu'aux unités compilées (CU).

QUERY-CU-DATE	Date de la dernière compilation de l'unité, sous la forme aammjj.
QUERY-CU-TIME	Heure de la dernière compilation de l'unité, sous la forme hhmmsscc.

Article SYS-MACRO

L'article SYS-MACRO décrit les macros compilées disponibles au cours de la session. Il ne donne accès qu'aux macros enregistrées dans BINLIB.

L'accès direct est possible sur la clé calculée (CALC) MACRO-NAME.

MACRO-NAME	Identification de la macro.
MACRO-DESCRIPTION	Brève description de la macro.
MACRO-DATE	Date de la dernière compilation, sous la forme aammjj.
MACRO-TIME	Heure de la dernière compilation, sous la forme hhmmsscc.

Article SYS-MACRO-SOURCE

L'article SYS-MACRO-SOURCE décrit chaque ligne de la version origine de la macro identifiée par SYS-MACRO. C'est un article détail de SYS-MACRO.

Il ne donne accès qu'aux macros origine enregistrées dans SLLIB.

MACRO-LINE-LENGTH	Longueur de la ligne origine.
MACRO-LINE-NUMBER	Numéro de la ligne origine.
MACRO-LINE	Contenu de la ligne origine.

Article SYS-REPORT

L'article SYS-REPORT identifie les descriptions d'état compilées disponibles au cours de la session. Il ne donne accès qu'aux descriptions d'état enregistrées dans BINLIB.

L'accès direct est possible sur la clé calculée (CALC) REPORT-NAME.

REPORT-NAME	Identification de la description d'état.
REPORT-DESCRIPTION	Commentaire explicatif.
REPORT-DATE	Date de la dernière compilation, sous la forme aammjj.
REPORT-TIME	Heure de la dernière compilation, sous la forme hhmmsscc.

Article SYS-REPORT-SOURCE

L'article SYS-REPORT-SOURCE décrit chaque ligne de la version origine de la description d'état identifiée par SYS-REPORT. C'est un article détail de SYS-REPORT.

Il ne donne accès qu'aux descriptions d'état origine enregistrées dans SLLIB.

REPORT-LINE-LENGTH	Longueur de la ligne origine.
REPORT-LINE-NUMBER	Numéro de la ligne origine.
REPORT-LINE	Contenu de la ligne origine.

Article SYS-FORMAT

L'article SYS-FORMAT décrit les formats compilés disponibles au cours de la session. Il ne donne accès qu'aux formats enregistrés dans BINLIB.

L'accès direct est possible sur la clé calculée (CALC) FORMAT-NAME.

FORMAT-NAME	Identification du format.
FORMAT-DESCRIPTION	Brève description du format.
FORMAT-DATE	Date de la dernière compilation, sous la forme aammjj.
FORMAT-TIME	Heure de la dernière compilation, sous la forme hhmmsscc.

10.3 Définition du schéma H_METAIQS

```
SCHEMA NAME IS 'H_METAIQS'.

AREA NAME IS SYSIQS.

RECORD NAME IS SYS-DD
      LOCATION MODE IS CALC USING DD-NAME  DUP ALLOWED
      WITHIN SYSIQS.
   02 DD-NAME             TYPE CHAR 30.
   02 DD-DESCRIPTION      TYPE CHAR 60.
   02 DD-TYPE             TYPE CHAR 3 CHECK VALUE "VDD" "SDD"
                          "FDD" "WDD".
   02 DD-DATE             TYPE UNSIGNED UNPACKED DECIMAL 6.
   02 DD-TIME             TYPE UNSIGNED UNPACKED DECIMAL 8.
   02 DD-VERSION          TYPE CHAR 5.
   02 DD-PRIVACY.
      03 DD-RETRIEVAL     TYPE CHAR 1 CHECK VALUE "0" "1".
      03 DD-UPDATE        TYPE CHAR 1 CHECK VALUE "0" "1".
      03 DD-IDS           TYPE CHAR 1 CHECK VALUE "0" "1".
      03 DD-VERSIDS-V5    TYPE CHAR 1 CHECK VALUE "0" "1".
      03 DD-VIEW-DDL-V5   TYPE CHAR 1 CHECK VALUE "0" "1".
      03 DD-ACCESS-RIGHT  TYPE CHAR 1 CHECK VALUE "0" "1".
   02 NUMBER-OF-SCHEMAS   TYPE UNSIGNED UNPACKED DECIMAL 2.
   02 NUMBER-OF-AREAS     TYPE UNSIGNED UNPACKED DECIMAL 4.
   02 NUMBER-OF-RECORDS   TYPE UNSIGNED UNPACKED DECIMAL 4.
   02 NUMBER-OF-SETS      TYPE UNSIGNED UNPACKED DECIMAL 4.

RECORD NAME IS SYS-SCHEMA
      LOCATION MODE IS VIA V-P SET
      WITHIN SYSIQS.
   02 SCHEMA-NAME         TYPE CHAR 30.
   02 SCHEMA-DATE         TYPE UNSIGNED UNPACKED DECIMAL 6.
   02 SCHEMA-TIME         TYPE UNSIGNED UNPACKED DECIMAL 8.

RECORD NAME IS SYS-AREA
      LOCATION MODE IS VIA V-A SET
      WITHIN SYSIQS.
   02 AREA-NAME           TYPE CHAR 30.
   02 AREA-ORG            TYPE CHAR 4 CHECK VALUE "SEQ" "IND"
                               "INTG" "REL".
```

```
RECORD NAME IS SYS-RECORD
      LOCATION MODE IS VIA V-R SET
      WITHIN SYSIQS.
   02 RECORD-NAME         TYPE CHAR 30.
   02 RECORD-DESCRIPTION  TYPE CHAR 60.
   02 RECORD-PRIVACY
      03 RECORD-RETRIEVAL TYPE CHAR 1 CHECK VALUE "0" "1".
      03 RECORD-UPDATE    TYPE CHAR 1 CHECK VALUE "0" "1".
      03 RECORD-INSERT    TYPE CHAR 1 CHECK VALUE "0" "1".
      03 RECORD-DELETE    TYPE CHAR 1 CHECK VALUE "0" "1".
   02 NUMBER-OF-ITEMS     TYPE UNSIGNED UNPACKED DECIMAL 4.

RECORD NAME IS SYS-ITEM
      LOCATION MODE IS VIA R-I SET
      WITHIN SYSIQS.
   02 ITEM-NAME           TYPE CHAR 30.
   02 ITEM-DESCRIPTION    TYPE CHAR 60.
   02 ITEM-TYPE           TYPE CHAR 4 CHECK VALUE
                          "UPKS" "UPK" "PKS"
                          "PK2"  "PK"  "BIN"
                          "CHAR" "STR" "GRP"
                          "REC".
   02 ITEM-LENGTH         TYPE UNSIGNED UNPACKED DECIMAL 5.
   02 ITEM-SCALE          TYPE SIGNED UNPACKED DECIMAL 2.
   02 ITEM-OFFSET         TYPE UNSIGNED UNPACKED DECIMAL 5.
   02 ITEM-LEVEL          TYPE UNSIGNED UNPACKED DECIMAL 2.
   02 ITEM-NOCC           TYPE UNSIGNED UNPACKED DECIMAL 5.
   02 ITEM-NSUBS          TYPE UNSIGNED UNPACKED DECIMAL 1.
   02 ITEM-CHECK.
      03 LOCAL-CHECK      TYPE CHAR 1 CHECK VALUE "0" "1".
      03 GLOBAL-CHECK     TYPE CHAR 1 CHECK VALUE "0" "1".
   02 ITEM-PRIVACY.
      03 ITEM-RETRIEVAL   TYPE CHAR 1 CHECK VALUE "0" "1".
      03 ITEM-UPDATE      TYPE CHAR 1 CHECK VALUE "0" "1".

RECORD NAME IS SYS-SET
      LOCATION MODE IS VIA V-S SET
      WITHIN SYSIQS.
   02 SET-NAME            TYPE CHAR 30.
   02 OWNER-RECORD-NAME   TYPE CHAR 30.
   02 MEMBER-RECORD-NAME  TYPE CHAR 30.
   02 SET-KEY.
      03 SET-CALC         TYPE CHAR 1 CHECK VALUE "0" "1".
      03 SET-SORTED       TYPE CHAR 1 CHECK VALUE "0" "1".
      03 SET-ASCENDING    TYPE CHAR 1 CHECK VALUE "0" "1".
      03 SET-DESCENDING   TYPE CHAR 1 CHECK VALUE "0" "1".
      03 SET-NODUP        TYPE CHAR 1 CHECK VALUE "0" "1".
```

```
RECORD TYPE IS SYS-PROJECT
LOCATION MODE IS VIA V-D
WITHIN SYSIQS.
   02 PROJECT-NAME        CHAR 12.
   02 USER-NAME           CHAR 12.
   02 ACCESS-TYPE         CHAR 2.

SCHEMA NAME IS SYS-QUERY
      LOCATION MODE IS CALC USING QUERY-NAME DUP NOT
      WITHIN SYSIQS.
   02 QUERY-NAME          TYPE CHAR 30.
   02 QUERY-DESCRIPTION   TYPE CHAR 60.
   02 QUERY-DATE          TYPE UNSIGNED UNPACKED DECIMAL 6.
   02 QUERY-TIME          TYPE UNSIGNED UNPACKED DECIMAL 8.
   02 QUERY-VIEW.         TYPE CHAR 30.

RECORD NAME IS SYS-QUERY-SOURCE
      LOCATION MODE IS VIA Q-S SET
      WITHIN SYSIQS.
   02 QUERY-LINE-LENGTH   TYPE BINARY 15.
   02 QUERY-LINE-NUMBER   TYPE UNSIGNED UNPACKED DECIMAL 5.
   02 QUERY-LINE          TYPE CHAR 255.

RECORD NAME IS SYS-QUERY-CU
      LOCATION MODE IS VIA Q-C SET WITHIN SYSIQS.
   02 QUERY-CU-DATE       TYPE UNSIGNED UNPACKED DECIMAL 6.
   02 QUERY-CU-TIME       TYPE UNSIGNED UNPACKED DECIMAL 8.

RECORD NAME IS SYS-MACRO
      LOCATION MODE IS CALC USING MACRO-NAME DUP NOT
      WITHIN SYSIQS.
   02 MACRO-NAME          TYPE CHAR 30.
   02 MACRO-DESCRIPTION   TYPE CHAR 60.
   02 MACRO-DATE          TYPE UNSIGNED UNPACKED DECIMAL 6.
   02 MACRO-TIME          TYPE UNSIGNED UNPACKED DECIMAL 8.

RECORD NAME IS SYS-MACRO-SOURCE
      LOCATION MODE IS VIA M-S SET
      WITHIN SYSIQS.
   02 MACRO-LINE-LENGTH   TYPE BINARY 15.
   02 MACRO-LINE-NUMBER   TYPE UNSIGNED UNPACKED DECIMAL 5.
   02 MACRO-LINE          TYPE CHAR 255.

RECORD NAME IS SYS-REPORT
      LOCATION MODE IS CALC USING REPORT-NAME DUP NOT
      WITHIN SYSIQS.
   02 REPORT-NAME         TYPE CHAR 30.
   02 REPORT-DESCRIPTION  TYPE CHAR 60.
   02 REPORT-DATE         TYPE UNSIGNED UNPACKED DECIMAL 6.
   02 REPORT-TIME         TYPE UNSIGNED UNPACKED DECIMAL 8.
```

```
RECORD NAME IS SYS-REPORT-SOURCE
      LOCATION MODE IS VIA R-S SET
      WITHIN SYSIQS.
   02 REPORT-LINE-LENGTH  TYPE BINARY 15.
   02 REPORT-LINE-NUMBER  TYPE UNSIGNED UNPACKED DECIMAL 5.
   02 REPORT-LINE         TYPE CHAR 255.

SCHEMA NAME IS SYS-FORMAT
      LOCATION MODE IS CALC USING FORMAT-NAME DUP NOT
      WITHIN SYSIQS.
   02 FORMAT-NAME         TYPE CHAR 30.
   02 FORMAT-DESCRIPTION  TYPE CHAR 60.
   02 FORMAT-DATE         TYPE UNSIGNED UNPACKED DECIMAL 6.
   02 FORMAT-TIME         TYPE UNSIGNED UNPACKED DECIMAL 8.

SET NAME IS V-P
   OWNER IS SYS-DD
      ORDER IS PERMANENT
      INSERTION IS NEXT.
   MEMBER IS SYS-SCHEMA
      INSERTION IS AUTOMATIC
      RETENTION IS MANDATORY
      SET SELECTION FOR V-P THRU V-P
      OWNER IDENTIFIED BY APPLICATION.

SET NAME IS V-A
   OWNER IS SYS-DD
      ORDER IS PERMANENT
      INSERTION IS NEXT.
   MEMBER IS SYS-AREA
      INSERTION IS AUTOMATIC
      RETENTION IS MANDATORY
      SET SELECTION FOR V-A IS THRU V-A
      OWNER IDENTIFIED BY APPLICATION.

SET NAME IS V-R
   OWNER IS SYS-DD
      ORDER IS PERMANENT
      INSERTION IS NEXT.
   MEMBER IS SYS-RECORD
      INSERTION IS AUTOMATIC
      RETENTION IS MANDATORY
      SET SELECTION FOR V-R IS THRU V-R
      OWNER IDENTIFIED BY APPLICATION.
```

```
SET NAME IS R-I
   OWNER IS SYS-RECORD
      ORDER IS PERMANENT
      INSERTION IS NEXT.
   MEMBER IS SYS-ITEM
      INSERTION IS AUTOMATIC
      RETENTION IS MANDATORY
      SET SELECTION FOR R-I IS THRU R-I
      OWNER IDENTIFIED BY APPLICATION.

SET NAME IS V-S
   OWNER IS SYS-DD
      ORDER IS PERMANENT
      INSERTION IS NEXT.
   MEMBER IS SYS-SET
      INSERTION IS AUTOMATIC
      RETENTION IS MANDATORY
      SET SELECTION FOR V-S IS THRU V-S
      OWNER IDENTIFIED BY APPLICATION.

SET NAME IS V-D
   OWNER IS SYS-DD
      ORDER IS PERMANENT
      INSERTION IS NEXT.
MEMBER IS SYS-PROJECT
      INSERTION IS AUTOMATIC
      RETENTION IS MANDATORY
      SET SELECTION FOR V-D THRU V-D
      OWNER IDENTIFIED BY APPLICATION.

SET NAME IS Q-S
   OWNER IS SYS-QUERY
      ORDER IS PERMANENT
      INSERTION IS NEXT.
   MEMBER IS SYS-QUERY-SOURCE
      INSERTION IS AUTOMATIC
      RETENTION IS MANDATORY
      SET SELECTION FOR Q-S IS THRU Q-S
      OWNER IDENTIFIED BY APPLICATION.

SET NAME IS Q-C
   OWNER IS SYS-QUERY
      ORDER IS PERMANENT
      INSERTION IS NEXT.
   MEMBER IS SYS-QUERY-CU
      INSERTION IS AUTOMATIC
      RETENTION IS MANDATORY
      SET SELECTION FOR Q-C IS THRU Q-C
      OWNER IDENTIFIED BY APPLICATION.
```

```
SET NAME IS M-S
   OWNER IS SYS-MACRO
      ORDER IS PERMANENT
      INSERTION IS NEXT.
   MEMBER IS SYS-MACRO-SOURCE
      INSERTION IS AUTOMATIC
      RETENTION IS MANDATORY
      SET SELECTION FOR M-S IS THRU M-S
      OWNER IDENTIFIED BY APPLICATION.

SET NAME IS R-S
   OWNER IS SYS-REPORT
      ORDER IS PERMANENT
      INSERTION IS NEXT.
   MEMBER IS SYS-REPORT-SOURCE
      INSERTION IS AUTOMATIC
      RETENTION IS MANDATORY
      SET SELECTION FOR R-S IS THRU R-S
      OWNER IDENTIFIED BY APPLICATION.
END-SCHEMA.
```

10.4 Accès à la métabase

L'accès à la métabase peut s'effectuer soit par l'intermédiaire d'une requête, soit au moyen d'une suite de commandes IQS. Dans le premier cas, la sélection du métaschéma est automatique, même si une vue a déjà été sélectionnée. Dans le second cas, l'utilisateur doit obligatoirement sélectionner le schéma de la métabase au moyen des commandes SELECT et OPEN.

Ces deux cas sont illustrés dans les exemples ci-dessous : chacun permet de visualiser le nom, la date de compilation et la vue sous-jacente de toutes les requêtes dont le nom commence par "Q".

EXEMPLE AVEC INSTRUCTIONS DU LANGAGE DE REQUETES :

```
C: AUTO
   10: RETRIEVE SYS-QUERY
   20: WHERE QUERY-NAME BEGINS "Q"
   30: PRINT WITH TITLE QUERY-NAME,QUERY-DATE,QUERY-VIEW
   40: END
   50: /

C: GO

QUERY-NAME                   QUERY-DATE QUERY-VIEW
QS-CALC                          880510 COCKTAIL-IDS
  .......       .........       ............
  .......       .........       ............
❑
```

EXEMPLE AVEC COMMANDES IQS :

```
C: SELECT H_METAIQS
V: OPEN
V: RT QUERY-NAME,QUERY-DATE,QUERY-VIEW -
-: FROM SYS-QUERY WHERE QUERY-NAME BG "Q"
V: RV SYS-QUERY
R: SN
   SYS-QUERY
       QUERY-NAME : QS-CALC
       QUERY-DATE : 880510
       QUERY-VIEW : COCKTAIL-IDS
   ..........................
   ..........................
❑
```

A. Exemples de schémas

L'utilisateur doit normalement disposer d'un diagramme et d'une définition de schéma pour chacun des schémas avec lesquels il travaille. Ces informations sont généralement fournies par l'administrateur IQS ou par toute autre personne compétente.

Cette annexe détaille les schémas utilisés dans la plupart des exemples figurant dans les chapitres précédents.

A.1 Exemple de schéma UFAS séquentiel

Un schéma UFAS séquentiel présente une structure arborescente.

Lorsqu'une aire comporte plusieurs types d'articles, chaque article doit comporter une zone définissant son type ; cette zone (TYP) est spécifiée dans la définition de schéma.

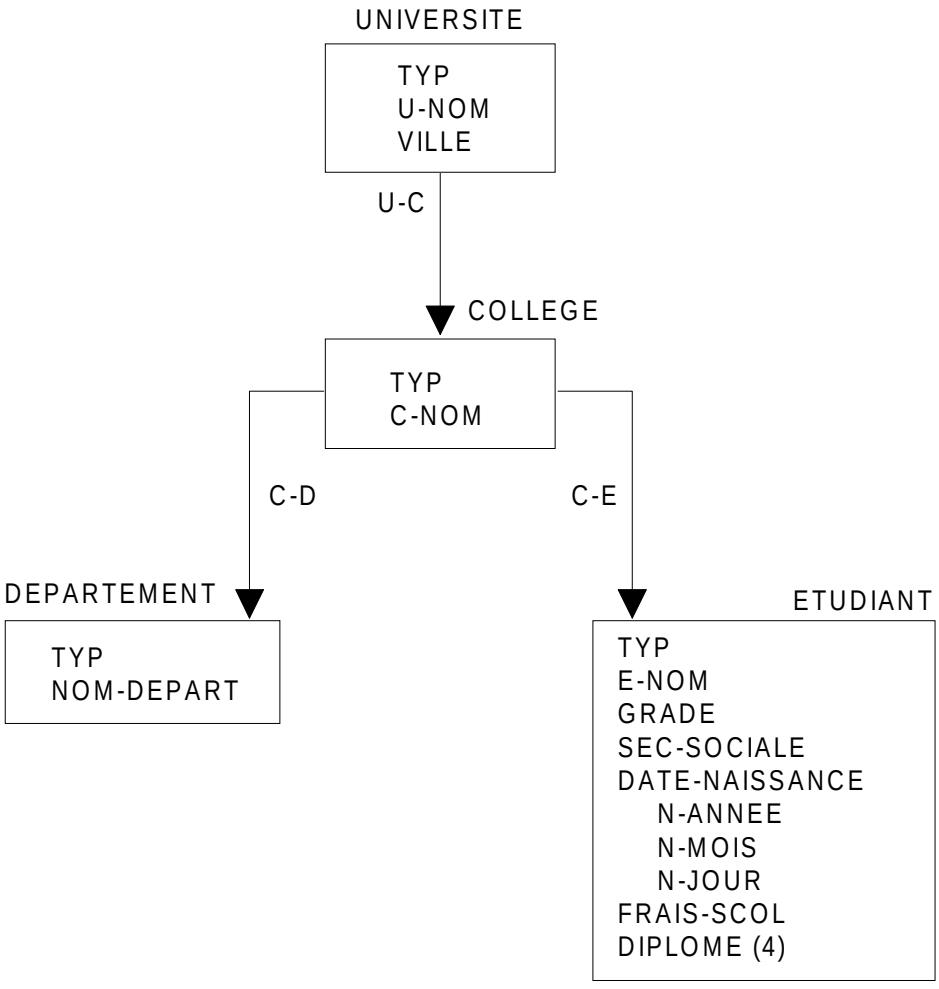
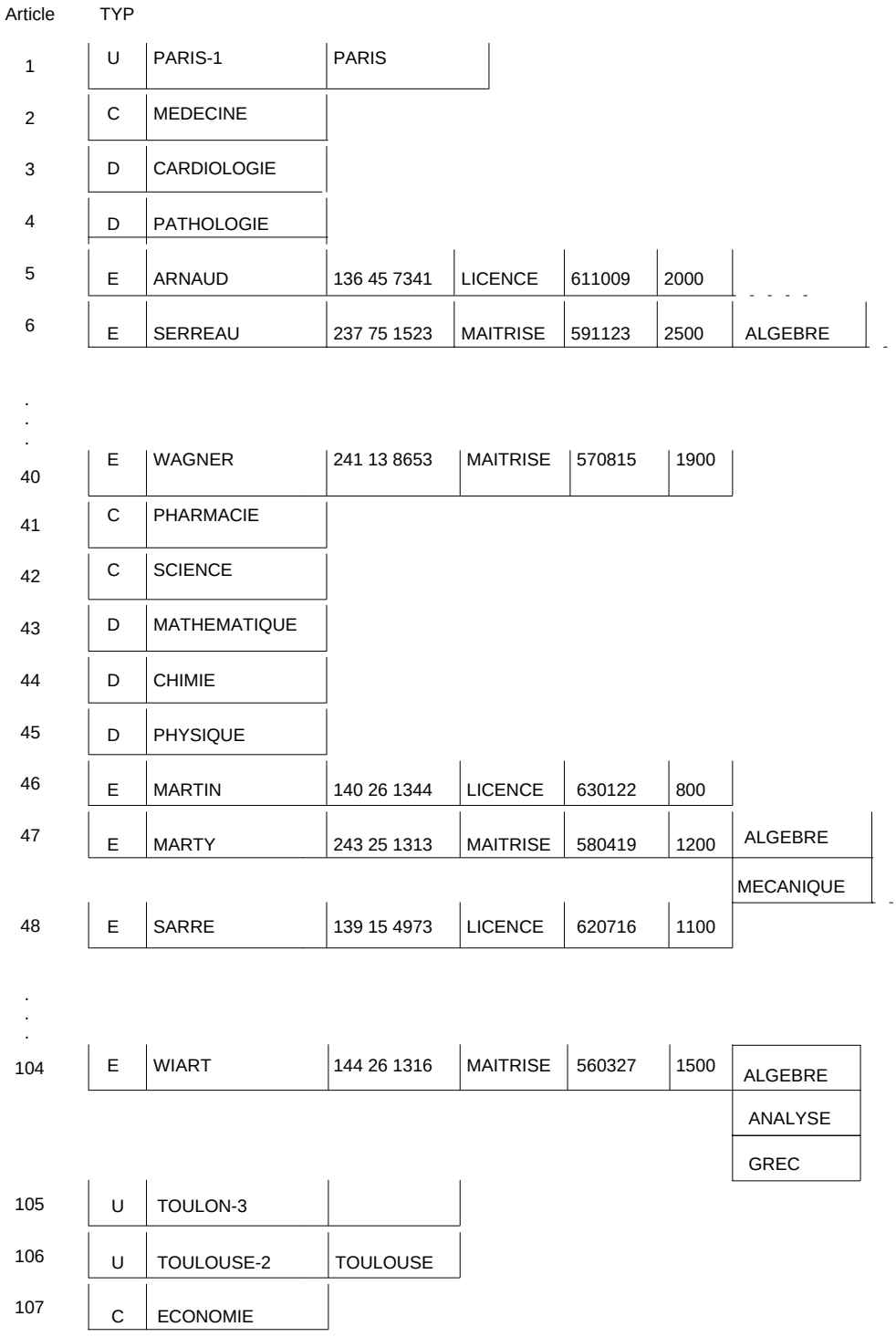


Figure A-1. Diagramme d'un schéma UFAS séquentiel

```
 10   SCHEMA NAME IS SEQ.
 20   AREA NAME IS SEQ-A.
 30   RECORD NAME IS UNIVERSITE WITHIN SEQ-A
 40        RECORD-TYPE DEFINED BY CHECK ON TYP.
 50     02 TYP TYPE IS CHARACTER 1 CHECK VALUE "U".
 60     02 U-NOM TYPE IS CHARACTER 20.
 70     02 VILLE TYPE IS CHARACTER 25.
 80   RECORD NAME IS COLLEGE WITHIN SEQ-A
 90        RECORD-TYPE DEFINED BY CHECK ON TYP.
100     02 TYP TYPE IS CHARACTER 1 CHECK VALUE "C".
110     02 C-NOM TYPE IS CHARACTER 20.
120   RECORD NAME IS DEPARTEMENT WITHIN SEQ-A
130        RECORD-TYPE DEFINED BY CHECK ON TYP.
140     02 TYP TYPE IS CHARACTER 1 CHECK VALUE "D"
150     02 NOM-DEPART TYPE IS CHARACTER 20.
160   RECORD NAME IS ETUDIANT WITHIN SEQ-A
170        RECORD-TYPE DEFINED BY CHECK ON TYP.
180     02 TYP TYPE IS CHARACTER 1 CHECK VALUE "E".
190     02 E-NOM TYPE IS CHARACTER 20.
200     02 SEC-SOCIALE TYPE IS CHARACTER 11.
210     02 GRADE TYPE IS CHARACTER 9
220        CHECK VALUE "LICENCE" "MAITRISE".
230     02 DATE-NAISSANCE.
240        04 N-ANNEE TYPE IS UNSIGNED UNPACKED DECIMAL 2.
250        04 N-MOIS TYPE IS UNSIGNE UNPACKED DECIMAL 2.
260           CHECK VALUE "1" THRU "12".
270        04 N-JOUR TYPE IS UNSIGNED UNPACKED DECIMAL 2
280           CHECK VALUE "1" THRU "31".
290     02 FRAIS-SCOL TYPE IS SIGNED BINARY 31.
300     02 DIPLOME TYPE IS CHARACTER 12 OCCURS 4.
310   SET NAME IS U-C
320       OWNER IS UNIVERSITE.
330       MEMBER IS COLLEGE.
340   SET NAME IS C-D
350       OWNER IS COLLEGE.
360       MEMBER IS DEPARTEMENT.
370   SET NAME IS C-E
380       OWNER IS COLLEGE.
390       MEMBER IS ETUDIANT.
400   END-SCHEMA.
```

Figure A-2. Exemple de définition de schéma pour un fichier UFAS séquentiel



Article	TYP						
1	U	PARIS-1	PARIS				
2	C	MEDECINE					
3	D	CARDIOLOGIE					
4	D	PATHOLOGIE					
5	E	ARNAUD	136 45 7341	LICENCE	611009	2000	
6	E	SERREAU	237 75 1523	MAITRISE	591123	2500	ALGEBRE
. . .							
40	E	WAGNER	241 13 8653	MAITRISE	570815	1900	
41	C	PHARMACIE					
42	C	SCIENCE					
43	D	MATHEMATIQUE					
44	D	CHIMIE					
45	D	PHYSIQUE					
46	E	MARTIN	140 26 1344	LICENCE	630122	800	
47	E	MARTY	243 25 1313	MAITRISE	580419	1200	ALGEBRE
							MECANIQUE
48	E	SARRE	139 15 4973	LICENCE	620716	1100	
. . .							
104	E	WIART	144 26 1316	MAITRISE	560327	1500	ALGEBRE
							ANALYSE
							GREC
105	U	TOULON-3					
106	U	TOULOUSE-2	TOULOUSE				
107	C	ECONOMIE					

Figure A-3. Exemple de placement des articles dans un fichier UFAS séquentiel

A.2 Exemple de schéma UFAS séquentiel indexé

Un schéma UFAS séquentiel indexé présente une structure hiérarchique linéaire.

L'accès direct à chaque article est possible par l'intermédiaire d'une clé primaire. Les zones significatives de la clé primaire dépendent du type d'article. La façon de déterminer ce type est indiquée dans le schéma. Il est également possible d'accéder aux articles par l'intermédiaire d'une ou plusieurs clés secondaires. Une clé peut se composer de plusieurs zones consécutives.

Pour que l'exploitation des relations hiérarchiques soit possible, l'ordre de classement du fichier doit être tel que toutes les occurrences détail d'une occurrence maîtresse figurent entre cette dernière et l'occurrence maîtresse suivante.

Les articles de type différent n'ont pas obligatoirement la même longueur. Cependant, celle-ci doit être suffisante pour contenir toutes les clés, même si elles comportent des zones non significatives.

La figure A-4 illustre la correspondance entre les relations hiérarchiques et l'organisation du fichier.

Dans cet exemple, la clé primaire se compose de trois zones :

- une zone identifiant une occurrence de UNIVERSITE (U-NOM)
- une zone identifiant une occurrence de COLLEGE dans chaque université (C-NOM)
- une zone identifiant une occurrence de ETUDIANT dans chaque collège (E-NOM).

La clé secondaire se compose d'une seule zone : SEC-SOCIALE ; elle permet d'accéder directement à une occurrence de ETUDIANT. La zone SEC-SOCIALE n'est pas significative dans les articles UNIVERSITE et COLLEGE.

La valeur des zones C-NOM et E-NOM n'est pas significative dans les occurrences de l'article UNIVERSITE, celle de la zone E-NOM ne l'est pas dans les occurrences de l'article COLLEGE.

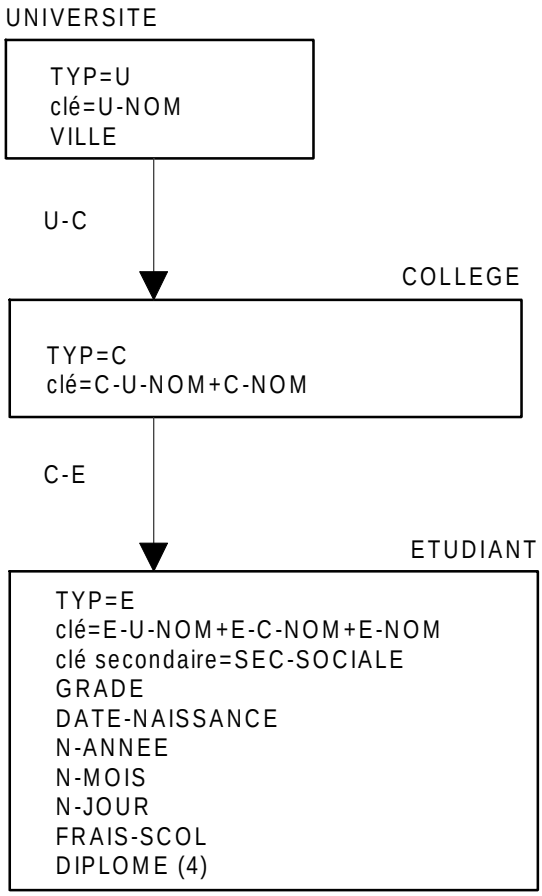


Figure A-4. Diagramme d'un schéma UFAS séquentiel indexé

```
 10  SCHEMA NAME IS IND.
 20  AREA NAME IS IND-A ORGANIZATION INDEXED USING KEY-1.
 30  RECORD NAME IS UNIVERSITE WITHIN IND-A
 40        RECORD-TYPE DEFINED BY CHECK ON TYP
 50        KEY KEY-1 IS ASCENDING U-NOM,FIL2,FIL3 DUP NOT
 60        KEY KEY-2 IS ASCENDING FIL4 DUP LAST.
 70     02 TYP TYPE IS CHARACTER 1 CHECK VALUE "U".
 80     02 U-NOM TYPE IS CHARACTER 20.
 90     02 FIL2 TYPE IS CHARACTER 20 CHECK VALUE " ".
100     02 FIL3 TYPE IS CHARACTER 20 CHECK VALUE " ".
110     02 FIL4 TYPE IS CHARACTER 11 CHECK VALUE " ".
120     02 VILLE TYPE IS CHARACTER 25.
130  RECORD NAME IS COLLEGE WITHIN IND-A
140        RECORD-TYPE DEFINED BY CHECK ON TYP
150        KEY KEY-1 IS ASCENDING C-U-NOM,C-NOM,FIL3 DUP NOT
160        KEY KEY-2 IS ASCENDING FIL4 DUP LAST.
170     02 TYP TYPE IS CHARACTER 1 CHECK VALUE "C".
180     02 C-U-NOM TYPE IS CHARACTER 20.
190     02 C-NOM TYPE IS CHARACTER 20.
200     02 FIL3 TYPE IS CHARACTER 20 CHECK VALUE " ".
210     02 FIL4 TYPE IS CHARACTER 11 CHECK VALUE " ".
220  RECORD NAME IS ETUDIANT WITHIN IND-A
230        RECORD-TYPE DEFINED BY CHECK ON TYP
240        KEY KEY-1 IS ASCENDING E-U-NOM,E-C-NOM,
250              E-NOM DUP NOT
260        KEY KEY-2 IS ASCENDING SEC-SOCIALE DUP LAST.
270     02 TYP TYPE IS CHARACTER 1 CHECK VALUE "E".
280     02 E-U-NOM TYPE IS CHARACTER 20.
290     02 E-C-NOM TYPE IS CHARACTER 20.
300     02 E-NOM TYPE IS CHARACTER 20.
310     02 SEC-SOCIALE TYPE  IS CHARACTER 11.
320     02 GRADE TYPE IS CHARACTER 9
330        CHECK VALUE IS "LICENCE" "MAITRISE".
340     02 DATE-NAISSANCE.
350        04 N-ANNEE TYPE IS UNSIGNED UNPACKED DECIMAL 2.
360        04 N-MOIS TYPE IS UNSIGNED UNPACKED DECIMAL 2
370              CHECK VALUE 1 THRU 12.
380        04 N-JOUR TYPE IS UNSIGNED UNPACKED DECIMAL 2
300              CHECK VALUE 1 THRU 31.
400     02 FRAIS-SCOL TYPE IS SIGNED BINARY 31.
410     02 DIPLOME TYPE IS CHARACTER 12 OCCURS 4.
420  SET NAME IS U-C
430     OWNER IS UNIVERSITE.
440     MEMBER IS COLLEGE.
450  SET NAME IS C-E
460     OWNER IS COLLEGE.
470     MEMBER IS ETUDIANT.
480  END-SCHEMA.
```

Figure A-5. Exemple de définition de schéma pour un fichier UFAS séquentiel indexé

REMARQUE :

Une clé primaire de 3 zones et une clé secondaire de 1 zone.

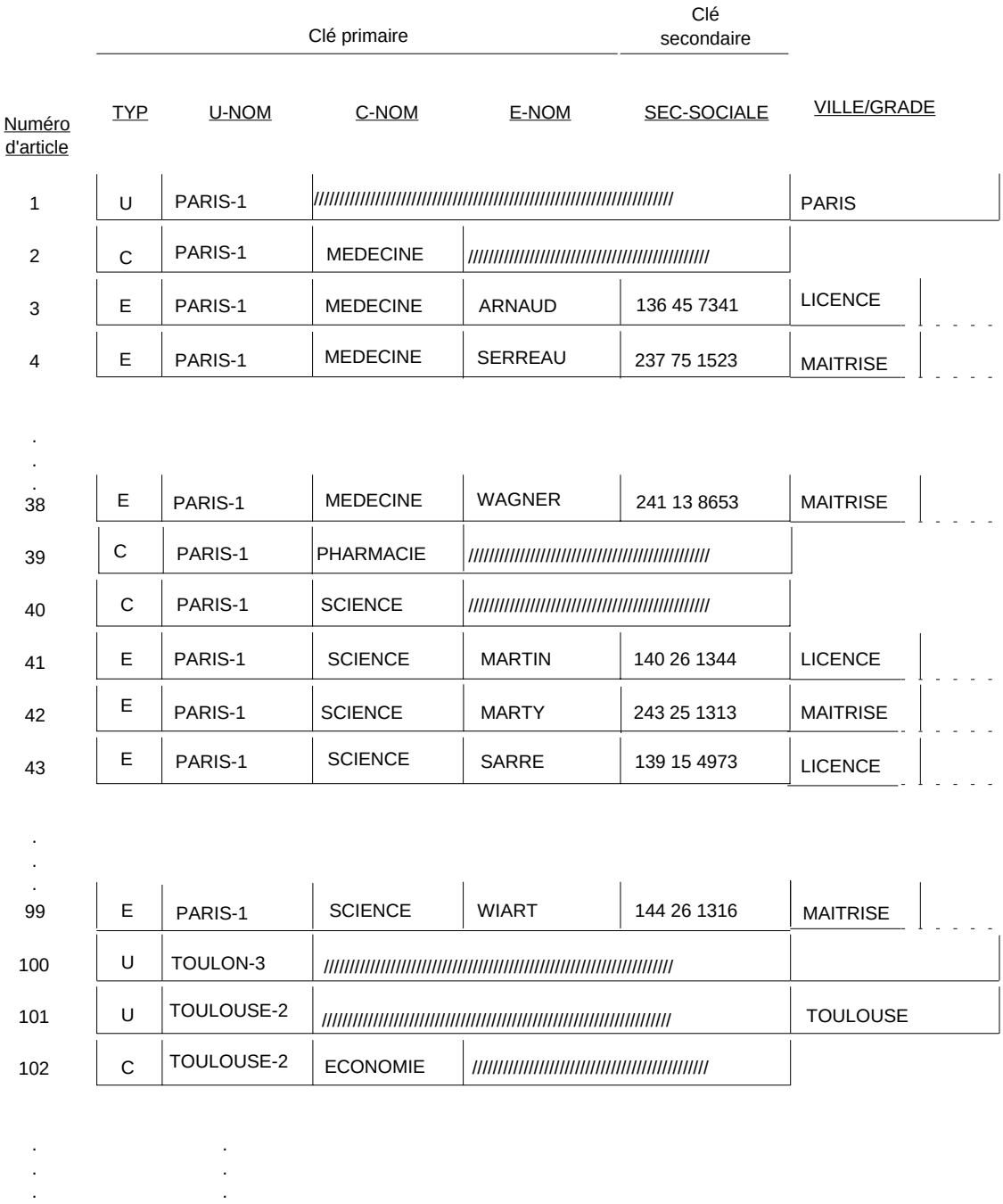


Figure A-6. Exemple de placement des articles dans un fichier UFAS séquentiel indexé

A.3 Exemple de schéma IDS/II

A.3.1 Schéma

La représentation du schéma sous forme de diagramme (cf. figure A-7) montre les éléments de la base de données vus sous l'angle utilisateur. La définition du schéma est en principe introduite par l'administrateur de la base de données, qui utilise pour ce faire le langage DDL ; lui seul peut la modifier. Pour plus de détails, se reporter au guide de l'administrateur IQS-V4 (80UR).

A.3.2 Articles réels

Les articles réels de la base de données sont symbolisés par des rectangles. Chaque article réel porte un nom unique qui est utilisé dans les instructions CREATE et RETRIEVE pour accéder à cet article. Dans cet exemple, les noms des articles réels sont les suivants :

```
UNIVERSITE
COLLEGE
DEPARTEMENT
ENSEIGNANT
ETUDIANT
COURS
HORAIRE
```

A.3.3 Zones

Les données que contient un article réel sont découpées en zones. La zone est le plus petit élément de donnée auquel puisse accéder l'utilisateur.

La liste des zones de chaque article réel est donnée dans le rectangle le symbolisant dans la figure A-7. Les zones de ETUDIANT, par exemple, sont les suivantes :

```
E-NOM
SEC-SOCIALE
GRADE
DATE-NAISSANCE
     N-ANNEE
     N-MOIS
     N-JOUR
FRAIS-SCOL
DIPLOME (4)
```

Les zones N-ANNEE, N-MOIS et N-JOUR sont des subdivisions de la zone DATE-NAISSANCE.

Les zones d'un article réel ne peuvent être spécifiées que dans le contexte d'une instruction CREATE ou RETRIEVE portant sur cet article.

EXEMPLE :

```
          .
          .
          .
100   :   RETRIEVE UNIVERSITE, COLLEGE
              .
              .
              .
200   :       RETRIEVE ENSEIGNANT
                  .
                  .
                  .
300   :       END
              .
              .
              .
400   :   END
          .
          .
          .
600   :   PRINT "FIN DE LA REQUETE"
```

❑

L'instruction RETRIEVE extérieure couvre les lignes 100 à 400. L'instruction RETRIEVE intérieure couvre les lignes 200 à 300. La zone U-NOM, qui fait partie de l'article UNIVERSITE, peut être utilisée entre les lignes 100 et 400 ; avant la ligne 100 et après la ligne 400, elle n'est pas utilisable.
La zone DEGRE, qui fait partie de ENSEIGNANT, peut être utilisée entre les lignes 200 et 300 ; ailleurs, elle ne peut pas être spécifiée. La zone GRADE ne peut en aucun cas figurer dans la requête puisque l'article ETUDIANT, dont elle fait partie, n'a pas été spécifié par une instruction RETRIEVE.

A.3.4 Ensembles

Les articles réels sont reliés logiquement par des relations qui déterminent des ensembles. Ceux-ci sont symbolisés dans la figure A-7 par des flèches. Chaque ensemble porte un nom unique et comporte un article réel maître et des articles réels détail. La relation maître-détail est une relation 1 à n, c'est-à-dire que chaque ensemble comporte un seul article maître dont chaque occurrence peut avoir plusieurs occurrences détail.

Les noms des ensembles de la figure A-7 sont les suivants :

```
U-C
C-D
D-F
C-EN
EN-C
C-E
EN-E
C-HR
E-HR
```

L'ensemble C-E par exemple, détermine la relation COLLEGE-ETUDIANT, ETUDIANT étant l'article détail de COLLEGE. Pour chaque occurrence de COLLEGE, il peut donc y avoir plusieurs occurrences de ETUDIANT. Par exemple, à l'occurrence C1 de COLLEGE pourraient correspondre 200 occurrences de ETUDIANT (E1, E2, E3 ..., E200).

L'instruction :

```
RETRIEVE COLLEGE, ETUDIANT
```

fournirait d'abord :

```
C1 E1
C1 E2
C1 E3
.
.
.
C1 E200
.
.
```

puis passerait à l'occurrence suivante de COLLEGE.

Dans les bases de données présentant une structure en réseau (IDS II, par exemple), un article détail peut avoir plusieurs articles maîtres et donc être accessible par l'intermédiaire de plusieurs ensembles. En cas d'ambiguïté, il est nécessaire de spécifier la clause VIA nom-ensemble.

Par exemple, dans l'instruction :

```
RETRIEVE UNIVERSITE, COLLEGE, DEPARTEMENT, ENSEIGNANT VIA D-F
```

cette clause permet de préciser par l'intermédiaire de quel ensemble doit s'effectuer l'accès à ENSEIGNANT puisqu'il y a deux possibilités : D-F et C-EN.

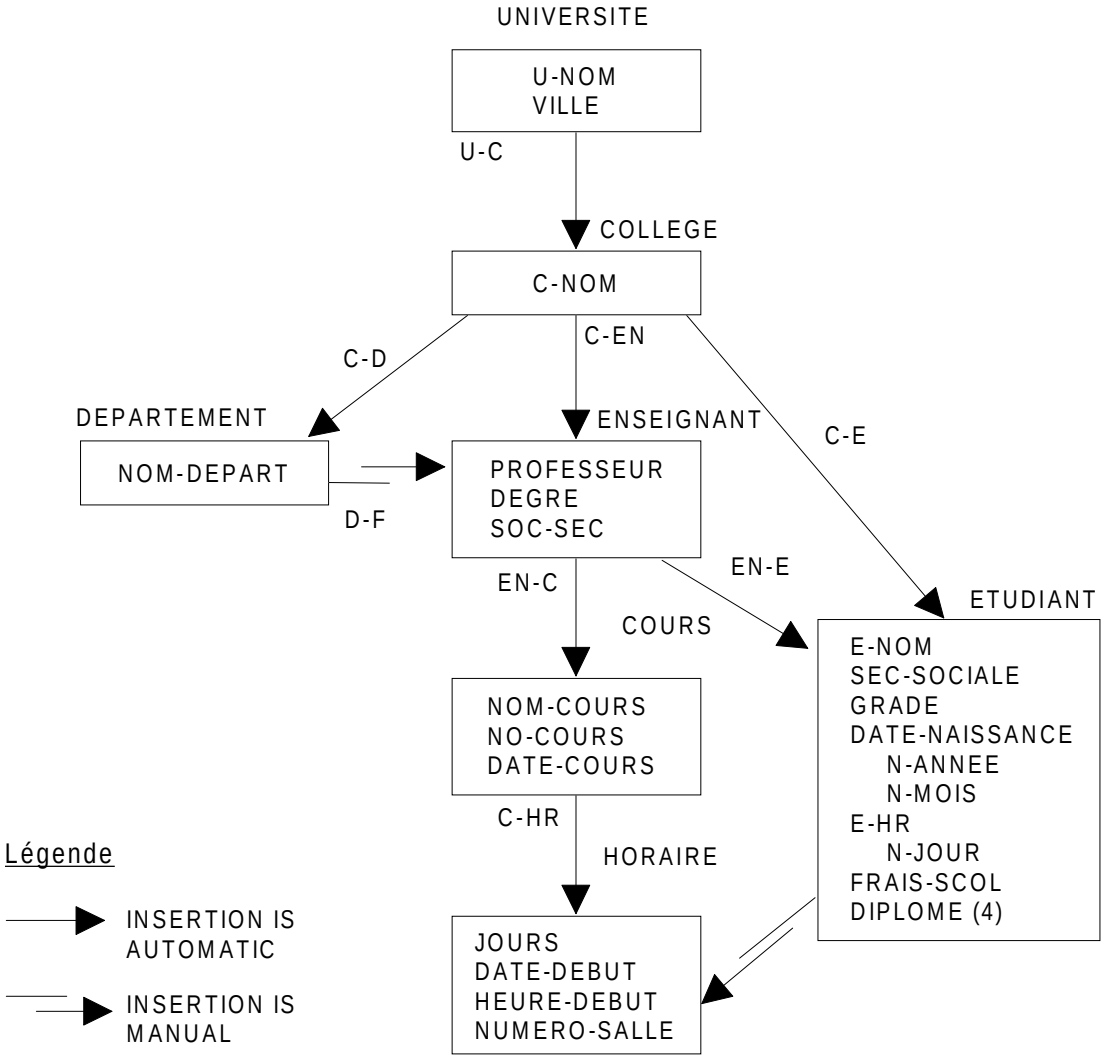


Figure A-7. Diagramme d'un schéma IDS/II

```
 10  SCHEMA NAME IS IDS.
 20  AREA NAME IS IDS-A.
 30  RECORD NAME IS UNIVERSITE.
 40        LOCATION MODE IS CALC USING U-NOM
 50        DUPLICATES NOT ALLOWED
 60        WITHIN IDS-A.
 70     02 U-NOM TYPE CHARACTER 20.
 80     02 VILLE TYPE CHARACTER 25.
 90  RECORD NAME IS COLLEGE
100        LOCATION MODE IS VIA U-C SET
110        WITHIN IDS-A.
120     02 C-NOM TYPE CHARACTER 20.
130  RECORD NAME IS ETUDIANT
140        LOCATION MODE IS CALC USING SEC-SOCIALE
150        DUPLICATES NOT ALLOWED
160        WITHIN IDS-A.
170     02 E-NOM TYPE CHARACTER 20.
180     02 SEC-SOCIALE TYPE CHARACTER 11.
190     02 GRADE TYPE CHARACTER 9.
200     02 DATE-NAISSANCE.
210        03 N-ANNEE TYPE UNSIGNED UNPACKED DECIMAL 2.
220        03 N-MOIS TYPE UNSIGNED UNPACKED DECIMAL 2.
230        03 N-JOUR TYPE UNSIGNED UNPACKED DECIMAL 2.
240     02 FRAIS-SCOL TYPE SIGNED BINARY 31.
250     02 DIPLOME TYPE CHARACTER 12 OCCURS 4 TIMES.
260  RECORD NAME IS DEPARTEMENT
270        LOCATION MODE IS VIA C-D SET
280        WITHIN IDS-A.
290     02 NOM-DEPART TYPE CHARACTER 20.
300  RECORD NAME IS ENSEIGNANT
310        LOCATION MODE IS VIA C-EN SET
320        WITHIN IDS-A.
330     02 PROFESSEUR TYPE CHARACTER 20.
340     02 DEGRE TYPE UNSIGNED PACKED DECIMAL 2.
350     03 SEC-SOC TYPE CHARACTER 11.
360  RECORD NAME IS COURS
370        LOCATION MODE IS VIA EN-C SET
380        WITHIN IDS-A.
390     02 NOM-COURS TYPE CHARACTER 20.
400     02 NO-COURS TYPE UNSIGNED DECIMAL 3.
410     02 DATE-COURS TYPE UNSIGNED UNPACKED DECIMAL 6.
420  RECORD NAME IS HORAIRE
430        LOCATION MODE IS VIA C-HR SET
440        WITHIN IDS-A.
450     02 JOURS TYPE CHARACTER 3.
460     02 DATE-DEBUT TYPE UNSIGNED UNPACKED DECIMAL 6.
470     02 HEURE-DEBUT TYPE UNSIGNED DECIMAL 4.
480     02 NUMERO-SALLE TYPE UNSIGNED DECIMAL 3.
```

Figure A-8. Exemple de définition de schéma IDS/II (1/3)

```
490  SET NAME IS U-C
500        OWNER IS UNIVERSITE
510        ORDER IS PERMANENT
520        INSERTION IS SORTED BY DEFINED KEYS
530        DUPLICATES NOT ALLOWED.
540     MEMBER IS COLLEGE
550           INSERTION IS AUTOMATIC
560           RETENTION IS MANDATORY
570           KEY IS ASCENDING C-NOM
580           SET SELECTION FOR U-C IS THRU U-C
590           OWNER IDENTIFIED BY APPLICATION.
600  SET NAME IS C-E
610        OWNER IS COLLEGE
620        ORDER IS PERMANENT
630        INSERTION IS SORTED BY DEFINED KEYS
640        DUPLICATES LAST.
650     MEMBER IS ETUDIANT
660           INSERTION IS AUTOMATIC
670           RETENTION IS MANDATORY
680           KEY IS ASCENDING E-NOM
690           SET SELECTION FOR C-E IS THRU C-E
700           OWNER IDENTIFIED BY APPLICATION.
710  SET NAME IS C-D
720        OWNER IS COLLEGE
730        ORDER IS PERMANENT
740        INSERTION IS SORTED BY DEFINED KEYS
750        DUPLICATES NOT ALLOWED.
760     MEMBER IS DEPARTEMENT
770           INSERTION IS AUTOMATIC
780           RETENTION IS MANDATORY
790           KEY IS ASCENDING NOM-DEPART
800           SET SELECTION FOR C-D IS THRU C-D
810           OWNER IDENTIFIED BY APPLICATION.
820  SET NAME IS C-EN
830        OWNER IS COLLEGE
840        ORDER IS PERMANENT
850        INSERTION IS NEXT.
860     MEMBER IS ENSEIGNANT
870           INSERTION IS AUTOMATIC
880           RETENTION IS MANDATORY
890           SET SELECTION FOR C-EN IS THRU C-EN
900           OWNER IDENTIFIED BY APPLICATION.
910  SET NAME IS D-F
920        OWNER IS DEPARTEMENT
930        ORDER IS PERMANENT
940        INSERTION IS NEXT.
950     MEMBER IS ENSEIGNANT
960           INSERTION IS MANUAL
970           RETENTION IS OPTIONAL
980           SET SELECTION FOR D-F IS THRU D-F
990           OWNER IDENTIFIED BY APPLICATION.
```

Figure A-9. Exemple de définition de schéma IDS/II (2/3)

```
1000  SET NAME IS EN-E
1010        OWNER IS ENSEIGNANT
1020        ORDER IS PERMANENT
1030        INSERTION IS LAST.
1040     MEMBER IS ETUDIANT
1050           INSERTION IS AUTOMATIC
1060           RETENTION IS MANDATORY
1070           SET SELECTION FOR EN-E IS THRU EN-E
1080           OWNER IDENTIFIED BY APPLICATION.
1090  SET NAME IS EN-C
1100        OWNER IS ENSEIGNANT
1110        ORDER IS PERMANENT
1120        INSERTION IS NEXT.
1130     MEMBER IS COURS
1140           INSERTION IS AUTOMATIC
1150           RETENTION IS MANDATORY
1160           SET SELECTION FOR EN-C IS THRU EN-C
1170           OWNER IDENTIFIED BY APPLICATION.
1180  SET NAME IS C-HR
1190        OWNER IS COURS
1200        ORDER IS PERMANENT
1210        INSERTION IS NEXT.
1220     MEMBER IS HORAIRE
1230           INSERTION IS AUTOMATIC
1240           RETENTION IS MANDATORY
1250           SET SELECTION FOR C-HR IS THRU C-HR
1260           OWNER IDENTIFIED BY APPLICATION.
1270  SET NAME IS E-HR
1280        OWNER IS ETUDIANT
1290        ORDER IS PERMANENT
1300        INSERTION IS NEXT.
1310     MEMBER IS HORAIRE
1320           INSERTION IS MANUAL
1330           RETENTION IS OPTIONAL
1340           SET SELECTION FOR E-HR IS THRU E-HR
1350           OWNER IDENTIFIED BY APPLICATION.
1360  END-SCHEMA.
```

Figure A-10. Exemple de définition de schéma IDS/II (3/3)

B. Mots réservés du langage de requêtes

Les mots figurant dans cette annexe ne doivent pas être spécifiés par l'utilisateur pour désigner des objets du langage de requêtes. De même, il est recommandé de ne pas utiliser les commandes IQS en tant que noms d'objets pour éviter les risques d'incompatibilité avec une prochaine version d'IQS.

AB	ABSENT	ABSREC	ACCEPT
AFTER	AG	ALTER	AMONG
AND	APPEND	AREA	AS
ASC	ASCENDING	ASG	ASSIGN
AVERAGE	BEFORE	BEGIN	BEGINS
BETWEEN	BG	BIN	BINARY
BT	BY	CANCEL	CASE
CENTER	CHANGE	CHAR	CHARACTER
CHECKPOINT	CHKFAIL	COL	COLUMN
COMMIT	CONC	CONCATENATE	CONNECT
CONTAIN	CONTAINS	COUNT	CREATE
CT	DATALIM	DATAOV	DEC
DECIMAL	DEF	DEFINE	DELETE
DESCENDING	DISCONNECT	DISPLAY	DIVZR
DO	DOES	DONE	DSC
EJECT	ELSE	END	EQ
EQUAL	EXEC	EXECUTE	EXIT
FILEMPTY	FILENCR	FOOTING	FORMAT
FREE	FROM	FUNCNAV	GE
GT	HEADING	HEXA	IDEM
IF	IFN	ILLCTR	ILLDATA
IN	INDEX	INSERT	JUST
JUSTIFIED	KEYUNKN	LE	LEFT
LENGTH	LET	LOWER	LT
MAX	MAXCTR	MEMBERS	MIN
MODIFY	NAG	NBG	NBT
NCOL	NCT	NE	NEXT
NODUP	NODUP1	NOT	NOTCON
OF	ON	ONLY	OR
ORDER	OTHER	OTHERS	PACKED
PARAM	PARAMETER	PERCENTAGE	PIC
PICTURE	PR	PRESENT	PRINT

PROMPT	PRT	RATIO	READ
RECCON	RECNFD	RECONNECT	REDEF
REDEFINES	REPEAT	REPORT	RESTART
RETAIN	RETRIEVE	RETURN	REWIND
RIGHT	ROLLBACK	SEQUENCE	SHORT
SIGNED	SORT	SPACE	SUBSTR
SUBSTRING	SUM	SYSTEM	THEN
THRU	TIMES	TITLE	TO
TOP	UNPACKED	UNSIGNED	UPPER
USE	VALUE	VIA	WHEN
WHERE	WITH	WITHIN	WRITE

En outre, l'utilisateur ne doit en aucun cas spécifier un nom commençant par l'un des caractères suivants :

$ & @ # %

C. Cas particuliers de @MESSAGE

C.1 Cas 1 : option COBOL DECIMAL POINT IS COMMA

En IQS, la syntaxe de cette option devient :

```
                                                {COMMA}
Let @MESSAGE = "SYSOPT DECIMAL POINT IS         {     }"
                                                {POINT}
```

et le résultat s'applique avec l'utilisation des verbes PRINT et REPORT.

Il s'applique également avec l'utilisation de la commande PRINT à partir de gestion FILE, (mais pas avec REVIEW par exemple).

DECIMAL POINT IS POINT vous renvoie à . par défaut sans quitter IQS.

Au niveau commande, cette syntaxe est acceptée en mode ligne.

```
 10    : DEFINE a          DECIMAL 6,2
 20    : DEFINE b          CHARACTER 10
 30    : DEFINE c          CHARACTER 12
 40    : PRINT " test option : DECIMAL POINT IS COMMA , on PRINT and REPORT
                               verbs"
 50    : LET a = 123.45
 60    : LET b = a
 70    : LET c = a pic"*****.99"
 80    : DISPLAY a,b,c
 90    : PRINT a,b,c
100    : PRINT a, a PIC"***B***.***" , a PIC"ZZZBZZZ.999"
110    : SPACE
120    : PRINT "Option is activated"
130    : LET @MESSAGE = "SYSOPT DECIMAL POINT IS COMMA"
140    : DISPLAY a,b,c
150    : PRINT a,b,c
160    : PRINT a, a PIC"***B***.***" , a PIC"ZZZBZZZ.999"
170    : SPACE
180    : PRINT "var. char are reassigned:"
190    : LET b = a
200    : LET c = a pic"*****.99"
```

```
  210    : DISPLAY a,b,c
  220    : PRINT a,b,c
  230    : PRINT a, a PIC"***B***.***" , a PIC"ZZZBZZZ.999"
  240    : SPACE
  250    : PRINT "Option is deactivated"
  260    : LET @MESSAGE = "SYSOPT DECIMAL POINT IS POINT"
  270    : DISPLAY a,b,c
  280    : PRINT a,b,c
  290    : PRINT a, a PIC"***B***.***" , a PIC"ZZZBZZZ.999"


C: compile
C: eject
C: exec
 test option : DECIMAL POINT IS COMMA , on PRINT and REPORT verbs
   A : 123.45
   B : 123.45
   C :     **123.45
  123.45 123.45         **123.45
  123.45 ****123.450     123.450

Option is activated
   A : 123.45
   B : 123.45
   C :     **123.45
  123,45 123.45         **123.45
  123,45 ****123,450     123,450

var. char are reassigned:
   A : 123.45
   B : 123.45
   C :     **123.45
  123,45 123.45         **123.45
  123,45 ****123,450     123,450

Option is deactivated
   A : 123.45
   B : 123.45
   C :     **123.45
  123.45 123.45         **123.45
  123.45 ****123.450     123.450
```

C.2 Cas 2 : accès aux variables GCL

En lecture uniquement :
`LET @Message="READVAR gcl-var-name[(index)]"`

Pour les mises à jour :
`LET Message ="MODVAR gel-var-name [(index)] text"`

Le résultat du mode lecture (READ) est stocké dans la variable IQS @Message définie comme CHAR 75.

Si le résultat est tronqué, le message "right truncation" s'affiche.

Si READVAR renvoie INDERR ou NAMERR, l'utilisateur trouve le texte de $H_EDITRC correspondant dans la variable @MESSAGE.

En mode mise à jour (UPDATE), l'ensemble de la chaîne à droite du signe = est limité à 255 caractères.

Le texte est à insérer au niveau du premier caractère non vide. Il ne peut pas commencer par un caractère vide (pour éviter l'ambiguïté avec les index facultatifs). Aucun code retour n'est renvoyé.

```
 10    : /*  GCL variables #SWITCHES , #SEV , #STATUS : readvar and modvar */
 20    : /*  caution: jcl SW0 is #SWITCHES(1)                               */
 30    : def swi (32) char(1)
 40    : do $i=1 to 32
 50    :   let @c=$i
 60    :   let @message= CONC (  "READVAR #SWITCHES (" ,@c short ," ) " )
 70    :   let swi($i)=@message
 80    : end
 90    : print "SWITCHES input :",swi
100    : SPACE
110    : /*  new values for switches                              */
120    : let swi="10000001001111000101010100001110"
130    : do $i=1 to 32
140    :   let @c=$i
150    :   let @message= CONC (  "MODVAR #SWITCHES (" ,@c short ,")",swi($i) )
160    : end
170    : print "SWITCHES values:",swi
180    : let swi=" "
190    : do $i=1 to 32
200    :   let @c=$i
210    :   let @message= CONC (  "READVAR #SWITCHES (" ,@c short ," ) " )
220    :   let swi($i)=@message
230    : end
240    : print "SWITCHES found :",swi
250    : SPACE
260    : print " GCL variables #SEV et #STATUS: read and modify"  col 10
270    : let @message = "READVAR #STATUS "
280    : print "#STATUS=", @message
```

```
  290    : let @message = "READVAR #SEV    "
  300    : print "#SEV   =", @message
  310    : let @message = "MODVAR  #STATUS 4000  "
  320    : let @message = "READVAR #STATUS "
  330    : print "#STATUS=" @message
  340    : let @message = "READVAR #SEV    "
  350    : print "#SEV   =" @message
  360    : let @message = "MODVAR  #SEV 3  "
  370    : let @message = "READVAR #SEV    "
  380    : print "#SEV   =" @message
  390    : let @message = "READVAR #STATUS "
  400    : print "#STATUS=" @message


C: compile
C: eject
C: exec
SWITCHES input : 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

SWITCHES values: 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 1 1 1 0
SWITCHES found : 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 1 1 1 0

          GCL variables #SEV et #STATUS: read and modify
#STATUS= 0
#SEV   = 0
#STATUS= 4000
#SEV   = 2
#SEV   = 3
#STATUS= 10000

   10    : /*  user GCL variable defined under IOF , or LMN (Batch)  :   */
   20    : /*       GB VAR-DEF  numval(0,5)                              */
   30    : /*   RC = NAMEERR when variable is not defined                */
   40    : /*   RC = INDERR  when variable has no value                  */
   50    : /*   on READVAR these errors are returned in @MESSAGE var.    */
   60    : /*   on MODVAR , a severity 3 message is edited and the query */
   70    : /* is stopped, when the variable is undefined                 */
   80    : def mes char 60
   90    : def @x char 2
  100    :
  110    : let @message="READVAR VAR-UNDEFINED"
  120    : print @message
  130    : let @message="READVAR VAR-DEF(1)"
  140    : print @message
  150    : let @message="MODVAR VAR-DEF ( 1 )  abcde""f "
  160    : let $i= 2
  170    : let @char1=$i pic "***B999CR" just left
  180    : let @x= $i pic "Z9"
  190    : let @message=conc ("MODVAR VAR-DEF ( " , @x " ) " , @char1 )
  200    : do $i=1 to 4
  210    :    let @x = $i pic "99"
```

```
  220    :    let mes=conc("READVAR VAR-DEF ( " @x " ) " )
  230    :    let @message= mes
  240    :    print $i,@message
  250    :    let @message=" "
  260    : end
  270    :
  280    : let @message=" READVAR #BLIB        "
  290    : print @message
  300    : let @message=" READVAR #BINLIB ( 2 )  "
  310    : print "#Binlib(2)=",@message
  320    : let @message=" READVAR #WTDS ( 1 )  "
  330    : print @message
  340    : let @message=" MODVAR  #WTDS QTDS "
  350    : let @message=" READVAR #wtds "
  360    : let @message=" MODVAR  #wtds ( 1 )   azer "
  370    : let @message=" READVAR #wtds ( 1 ) "
  380    : print "Some GCL variables cannot be modified by MODVAR :"
  390    : let @message=" MODVAR  #BINLIB ( 1 ) QUERY.TEST.BINTST "
C: compile
C: eject
C: exec
RC=98081870->GCL    8,NAMEERR
RC=9808181A->GCL    8,INDERR
          1 abcde"f
          2 ****002
          3 RC=9808181A->GCL    8,INDERR
          4 RC=9808181A->GCL    8,INDERR
RC=9814181A->GCL   20,INDERR
#Binlib(2)= RC=9814181A->GCL   20,INDERR

Some GCL variables cannot be modified by MODVAR :
***RC=98141878->GCL   20,NOMATCH  at address 8.85.8B60
   Parameter(s) :  MODVAR name=#BINLIB ,index=1,iln=17,value=QUERY.TEST.BINTST;
    at line   390
```

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

Z