UNIVERSITE DE NANCY I

U. E. R. DE MATHEMATIQUES

acquisition et édition des fichiers, analyse des données dans le projet civa

thèse

pour l'obtention
du doctorat de spécialité
mathématiques appliquées
(informatique)

soutenue le 19 décembre 1973
par
jean-jacques chabrier

jury :

M.	C. PAIR	Président
M.	M. DEPAIX	Examinateur
M.	J. C. DERNIAME	Examinateur
Mme	C. ROLLAND	Examinateur

UNIVERSITE DE NANCY I
U. E. R. DE MATHEMATIQUES

acquisition et édition des fichiers, analyse des données dans le projet civa

thèse

pour l'obtention
du doctorat de spécialité
mathématiques appliquées
(informatique)

soutenue le 19 décembre 1973
par
jean-jacques chabrier

jury :

M.	C. PAIR	Président
M.	M. DEPAIX	Examinateur
M.	J. C. DERNIAME	Examinateur
Mme	C. ROLLAND	Examinateur

Je tiens à exprimer toute ma gratitude à Monsieur le Professeur PAIR, Directeur de l'Institut Universitaire de Calcul Automatique de NANCY pour la bienveillance qu'il n'a cessé de me témoigner et pour la formation que j'ai reçue à l'Institut.

Je remercie vivement Monsieur le Professeur DEPAIX pour l'honneur qu'il me fait en participant au jury.

Je remercie également Madame Colette ROLLAND chargée d'Enseignement à l'Université I qui a bien voulu accepter de participer au Jury.

Ce travail a été réalisé sous la direction de Monsieur DERNIAME chargé d'Enseignement à l'Université I à qui j'exprime ma profonde et amicale reconnaissance pour l'attention, les précieux conseils et les orientations fructueuses qu'il m'a prodigués.

J'adresse mes remerciements aux personnes de l'équipe CIVA pour la confiance qu'elles m'ont toujours manifestée.

Enfin je remercie Madame MARCHAND et Mademoiselle MESSEZ qui ont assuré avec rapidité la réalisation matérielle de cette thèse.

S O M M A I R E

...

INTRODUCTION

Dans le but de faciliter l'analyse générale, la programmation et l'exploitation des applications d'informatique de gestion, le projet CIVA [1] vise à donner la possibilité à l'utilisateur de formuler dans un langage unique les différentes phases de la mise en oeuvre d'un travail-conception, compte rendu d'analyse, programmation, mise au point, exécution, maintenance.

Pour atteindre ce but, notre rôle dans ce projet a été d'une part de contribuer à la définition des informations internes, d'autre part d'analyser et de décrire les informations externes en entrée ou en sortie de l'ordinateur, de définir des instructions élémentaires de transfert pour l'Acquisition, de formaliser les contrôles de validité et enfin d'élaborer des techniques d'aide à l'analyse et à la programmation CIVA.

Après avoir présenté dans le Chapitre 1 les problèmes rencontrés pour l'analyse, la programmation de l'Acquisition et de l'Edition de fichiers, nous donnerons brièvement les principes que nous étudierons par la suite en détail.

Le chapitre 2 est une étude théorique de certaines relations sémantiques sur les informations externes qui permettra de développer une analyse cohérente des fichiers externes en entrée et des contrôles de validité. Cette étude utilisera certains résultats de travaux de C. DELOBEL [24] et F. PECCOUD [25] sur les structures de l'information dans les Banques de données, nous présenterons ensuite la syntaxe du langage de description des fichiers externes en entrée, dans le chapitre 3.

Dans le Chapitre 4, nous étudierons en détail la conception des services d'Acquisition des fichiers externes. Enfin, nous décrirons les principes de conception des Services d'Edition dans le Chapitre 5.

CHAPITRE 1. PRESENTATION GENERALE

1.1. Introduction.

Dans le cas des applications d'informatique de gestion où nous nous plaçons dans le projet CIVA [1] : chaîne d'exploitation dont la durée de vie n'est pas trop courte ; le passage de la conception à l'exécution pourra être relativement long. L'exécution devra être rapide.

Pour que les traitements en ordinateur soient le plus efficace possible, l'utilisateur est amené à faire des choix, de codification de l'information, d'organisation logique des fichiers. Or, les fichiers que nous qualifions par la suite d'externes sont avant tout conçus pour faciliter leur utilisation à l'extérieur de l'ordinateur. De ce fait, les critères de choix retenus pour la codification et l'organisation logique des fichiers externes ne peuvent généralement pas être retenus pour des traitements efficaces en ordinateur.

Nous avons alors pensé à séparer les informations : en informations externes et en informations internes. Notons qu'avec certains langages de programmation, par exemple COBOL [13], il est possible d'utiliser dans la description externe des fichiers, des clauses (COMP 1, COMP 2, COMP 3) qui adaptent la codification externe aux traitements internes ultérieurs. Toutefois, l'utilisation d'une seule description de fichier impose pour la création des fichiers internes, soit une structure d'information externe au détriment de l'interne résultant d'une analyse des traitements, soit une structure d'information interne qui rend alors difficile l'utilisation des documents externes en entrée.

Dans notre travail, nous avons donc commencé d'une part à étudier les informations externes et internes, d'autre part à caractériser les informations externes par certaines relations sémantiques, de façon à élaborer des aides à l'analyse de descriptions de fichiers externes indépendantes de la structure d'information interne.

...

De cette manière, l'"entrée" de fichier externe en ordinateur que nous appellerons par la suite Acquisition de fichiers consistera à :

- assurer la correspondance entre article logique externe et article logique interne ;
- traiter les contrôles de validité quasi automatiques ;
- traiter les conversions de format externe en format interne ;
- éditer les données erronées avec des messages d'erreurs.

Quant à la"sortie" de fichier interne de l'ordinateur que nous appellerons Edition de fichier, les différentes fonctions à assurer seront essentiellement :

- la correspondance entre article logique interne et la description de fichier externe ;
- les conversions de format interne en format externe édité.

Comme nous l'étudierons en détail dans les chapitres 3, 4 et 5, l'Acquisition et l'Edition de fichiers externes ne pourront être traitées par une seule instruction, beaucoup trop de conventions étant possibles. Notre but sera de définir les instructions élémentaires qui donneront la possibilité à l'utilisateur de définir facilement des services adaptés à leurs problèmes. C'est d'ailleurs en ce sens que nous développerons des techniques d'aide à l'analyse et à la programmation.

1.2. Rappels sur les principaux contrôles de validité de données.

1.2.1. Généralités.

Résultant, soit de nombreuses actions humaines lors de la saisie, soit de création de fichiers de "sortie" par des chaînes de traitement de logiques complexes, les données se trouvent souvent entâchées d'erreurs. (référence 1.2.2.).

Un système de contrôles doit alors réaliser un compromis difficile entre le degré de fiabilité attaché à l'information, et le coût de la mise en oeuvre des contrôles. Une approche de ce compromis pourrait être traitée par des évaluations sur des jeux d'essais.

...

Toutefois pour que ces principes ne tombent pas en désuétude, il convient qu'à l'exploitation, une organisation de responsables soit mise en place. En effet, devant le volume important et la diversité des données manipulées, il sera utile qu'en amont, à l'exploitation et en aval de l'ordinateur, des responsables efficients soient en mesure de prendre en charge les données erronées, la rectification des erreurs ainsi que le recyclage dans le système de contrôles. Sans vouloir entrer dans le détail de l'organisation pour le traitement d'une application importante d'informatique de gestion [20][19], nous pourrons préciser les rôles des trois centres de responsabilités.

1.2.1.1. Le centre "en amont" de l'ordinateur est responsable de la saisie des données - codification, transcription, rectification d'erreurs -, d'un premier contrôle des données (perforation, vérification) et de la présentation des données au deuxième centre.

1.2.1.2. Le centre de traitement que l'on subdivise en trois services :

- le service d'Acquisition responsable de l'"entrée" des données externes fournies par le centre "en amont".
- le service d'exploitation qui a la charge de tous les traitements d'informations internes.
- le service d'édition responsable de la "sortie" des résultats en format externe.

1.2.1.3. Le centre de dépouillement des résultats, responsable des résultats fournis par le centre de traitement (résultats complets,...).

L'expérience montre que la situation stratégique des contrôles de données se situe au service d'Acquisition. En effet, de la qualité des données neuves entrées dépend la qualité des résultats. De plus, certaines erreurs de conception des programmes entraînent des résultats incomplets détectés dans la phase 1.2.1.3. de dépouillement, ce qui impose des contrôles souvent visuels.

...

Nous devons effectuer une distinction entre les contrôles "fournis" par le software du constructeur et les contrôles à "réaliser" par l'utilisateur.

Les contrôles "fournis" appelés aussi contrôles de sécurité ou de protection sont devenus des contrôles quasiment standard, indépendants des langages utilisés et à la charge des systèmes d'exploitation. Nous ne les étudierons pas dans le cadre de ce travail. Rappelons cependant quelques fonctions assurées [10] : contrôles de labels à la lecture ou à l'écriture de fichiers, contrôles de transmission physique de blocs de données, d'enregistrements, contrôles de reprises...

Cependant, il sera indispensable d'insérer les contrôles "fournis" dans le dossier d'analyse. Disons seulement qu'ils seront à la charge et relèverons de la responsabilité de l'exploitation en général.

1.2.2. Principaux contrôles de validité à "réaliser".

Du fait de l'insuffisance des langages classiques - COBOL, PL 1 - dans la description des données et d'un manque important de formalisation des contrôles de validité, les analystes-programmeurs se trouvent contraints à utiliser toute une panoplie de contrôles de validité [20] ce qui nécessite une analyse et une programmation toujours longues et onéreuses pour obtenir en définitive des résultats souvent décevants.

En outre, comme nous l'étudierons en détail dans les chapitres 3 et 4, les contrôles de validité doivent vérifier que les données satisfont bien à des critères de validation de format (type - taille) de vraisemblance (ensemble de données de référence réputées sûres), de compatibilité (donnée permettant une vérification supplémentaire) ou de structure de présentation d'une collection de données. Les différents contrôles d'erreurs envisagés devront assurer la détection de trois types d'erreurs sur les données : l'omission, la substitution, l'adjonction de caractères, ou de rubriques, ou d'enregistrements, ou même d'articles.

...

Sur certaines données, d'ailleurs, nous pouvons avoir plusieurs de ces erreurs qui se cumulent. Ce n'est généralement qu'à l'analyse de la donnée erronée (s'il y a eu détection) que l'on a la possibilité de diagnostiquer toutes les erreurs. Il se peut aussi qu'une erreur, ou même qu'un cumul d'erreurs ne soit pas détecté malgrè tous les contrôles (exemple : inversions de chiffres ou de lettres dans un code).

Les principaux contrôles exploités actuellement dans les différentes applications de gestion, utilisent encore souvent des principes de la mécanographie.

1.2.2.1. Contrôle du total par lot.

Ce contrôle suppose que l'on totalise manuellement dans un lot les valeurs successives d'une rubrique significative et que l'on transmette le total général du lot avec le lot de données. Ce total est recalculé par programme au cours de l'entrée des données en ordinateur. Si le total calculé manuellement correspond au total calculé par programme, le lot est considéré 'bien' transmis.

Exemple. Dans une application de comptabilité, le lot serait une page d'un Journal. Chaque page étant constituée de lignes d'écritures ; la dernière ligne de la page sera composée d'un total débit, d'un Total crédit et d'un Total solde calculés manuellement.

Cette technique est une forme de contrôle de parité.

1.2.2.2. Contrôle d'assortiment (ou de ménage, ou de famille, ou de séquence).

Il s'agit avec ce contrôle de vérifier :

- qu'une famille d'enregistrements est en séquence normale avec la famille précédente. Par exemple, un numéro de facture est supérieur au numéro de facture enregistrée précédemment.

...

- qu'une famille d'enregistrements est complète.
Par exemple, une facture comporte au minimum un "en-tête" (identification client, numéro facture, ...), une condition de livraison (mode de transport, date, ...) et au moins un article (code, quantité, ...).

- qu'une famille d'enregistrements est correcte.
Par exemple, le total des lignes d'une facture correspond bien au nombre d'articles facturés.

1.2.2.3. Contrôle de logique ou de compatibilité.

Ce type de contrôle est le plus mal résolu actuellement, ce qui entraîne une gamme étendue de contrôles possibles.

- contrôles portant sur la nature des opérations (une opération comptable doit à chaque débit d'un compte avoir un crédit correspondant).

- contrôles de vraisemblance par rapport à des fourchettes (code département $[1,99]$).

1.2.2.4. Contrôle de validité du format :

Il s'agit de vérifier la présence de données obligatoires, le format de toutes les données, c'est à dire la classe, la taille de chaque donnée (un code de numéro de facture doit être numérique et avoir une taille donnée). On en profite généralement pour effectuer un contrôle de vraisemblance par rapport à une fourchette.

1.2.2.5. Contrôle de codes par lettre-clé.

On attribue, par un algorithme simple, une lettre-clé calculée à un code. Cette lettre doit être recalculée à l'entrée en machine. Ce procédé est utilisé sur certains codes (codes autocorrecteurs, auto-détecteur). Ceci permet de détecter des inversions de caractères dans le code $[6]$ généralement sur une seule rubrique contrairement à 1.3.

...

Exemple de code autodétecteur :

Supposons que l'on ait le code 7 2 3 4 5.

Pour acquérir une plus grande présomption sur l'exactitude de ce code, on peut par exemple ajouter une lettre de contrôle au code calculé par l'algorithme suivant.

On multiplie chaque chiffre du code en partant de la droite par les entiers 1, 2, 3,.... La somme des produits obtenus est divisée par 23. Le reste de la division par 23 (plus grand entier premier inférieur à 26) correspond au rang d'une lettre de contrôle dans l'alphabet.

Soit :

$$\binom{7}{5}+\binom{2}{4}+\binom{3}{3}+\binom{4}{2}+\binom{5}{1}$$

$$35 + 8 + 9 + 8 + 5 = 65$$

d'où :

$$65 \;/\; 23 = 2 \text{ reste } 19$$

La lettre de contrôle est la 19 ème de l'alphabet, soit S.
Nous en déduisons le code autodétecteur suivant 7 2 3 4 5 S.

1.3. Problèmes rencontrés à l'analyse des contrôles de validité.

La règle retenue lors de l'analyse générale d'une phase de validation utilisant les contrôles d'erreurs précédemment énumérés en 1.2.1. consiste pour chaque donnée à détecter et signaler toutes les erreurs possibles, et si la donnée est erronée, l'éditer puis éventuellement la rectifier. Cependant certaines données peuvent comporter plusieurs erreurs que le contrôle de validité dans un premier passage, ne détecte pas. De plus, la rectification restant une action humaine, on ne peut donc pas considérer qu'une donnée rectifiée est exacte. Aussi l'analyste se trouve-t-il confronté à un choix difficile dépendant à la fois d'impératifs de coût et de temps mais aussi de la fiabilité de l'information prévue.

...

Deux conceptions sont possibles :

- soit les données erronées sont recyclées, après rectification, un minimum de fois jusqu'à la validation ou le rejet, la donnée validée possédant alors la fiabilité escomptée.

- soit certaines données erronées ne sont pas recyclées mais traitées par des procédures d'exception. La donnée est alors validée plus rapidement au détriment de la fiabilité.

Exemple. Contrôle de famille.

Supposons que nous désirions vérifier que, sur une facture, le montant total d'un bordereau est bien égal à la somme des montants de chaque article facturé et que certains codes articles ne soient pas exacts. Alors, après la détection d'un ou plusieurs codes articles erronés de la facture :

- soit nous effectuons un "blocage" de la facture erronée, c'est à dire qu'elle est refusée et doit être éditée avec les messages d'erreurs appropriés sur un fichier de rebut, en attente de rectification. Ce n'est qu'après avoir été corrigée que la facture sera recyclée dans le système complet de contrôles.

- soit nous effectuons une validation partielle avec seulement des messages d'anomalie - les amendements nécessaires seront, dans ce cas, traités dans une phase ultérieure de mise à jour.

Cet exemple montre que les choix des contrôles sont difficiles à déterminer. La compléxité de la tache est si grande que très souvent certains contrôles sont conçus à postériori.

1.4. Eléments de conception de l'Acquisition et de l'Edition de fichiers externes.

Ces éléments sont de trois sortes :

1.4.1. Déterminer une technique d'aide à une analyse fine des fichiers externes, et définir un langage de description permettant de décrire facilement les résultats de cette analyse.

1.4.2. Définir des instructions de base d'Entrée-Sortie de CIVA.

...

1.4.3. Donner la possibilité à l'utilisateur de définir un certain nombre de métamodules et de méta fonctions pour la réalisation d'Acquisition et d'Edition de fichiers externes adaptés à leur application.

Bien que tous ces points soient étudiés en détail par la suite, nous les reprenons en apportant quelques précisions.

1.4.1. Langage de description des fichiers externes.

Partant de l'analyse de problèmes très différents en apparence (gestion de stocks, facturation, ...) traités avec le langage COBOL (ou PL 1), nous en avons retenu plusieurs problèmes communs pour l'entrée et la sortie des fichiers en format externe et que nous énumérerons ci-dessous :

1.4.1.1. Détermination des données externes.

1.4.1.2. Détermination de la structure d'information associée aux données externes (DATA STRUCTURE).

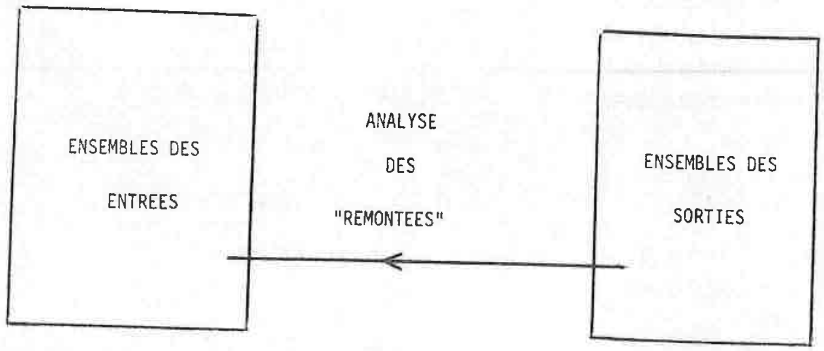
1.4.1.3. Détermination de la structure de présentation des données externes (STORAGE STRUCTURE)

1.4.1.1. Détermination des données externes.

Nous distinguerons ici les données externes en entrée et les données externes en sortie; celles-ci étant supposées connues.

Rappelons le principe de détermination des données externes en entrée au niveau de la conception générale d'un système de gestion [9] [23] à partir des données externes en sortie du système. Au début de la conception d'un nouveau système, seules sont connues les sorties ; donc la détermination des entrées s'effectuera par "remontée" à partir de la description des données externes en sortie.

...



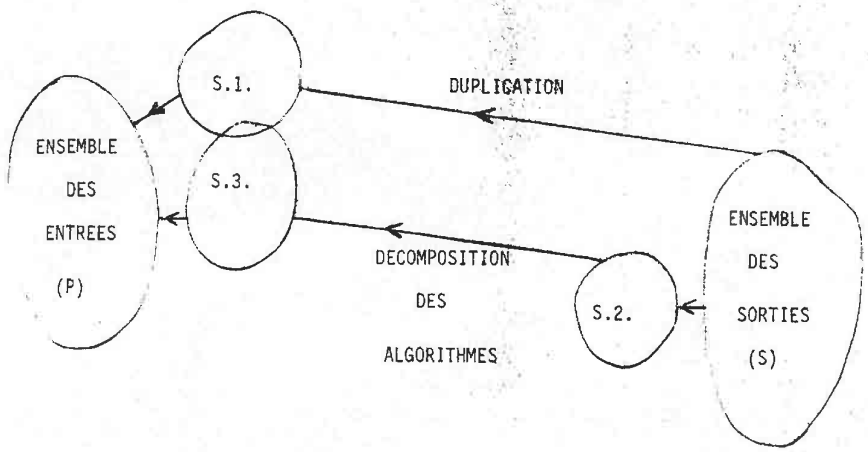
Le principe de l'analyse des "remontées" est le suivant :

- certaines rubriques de sortie obtenues par recopie d'une rubrique d'entrée appartiennent à l'ensemble des entrées.

- les autres rubriques de sortie se déduisent de l'application d'un algorithme. On "remonte" dans l'algorithme jusqu'aux opérandes primaires qui appartiennent obligatoirement à l'ensemble des entrées.

L'ensemble des rubriques de sorties (S) est divisé en deux sous-ensembles disjoints (S.1.) et (S.2.). Une rubrique appartient à (S.1.) si elle est une recopie d'une rubrique d'entrée ; sinon elle appartient à (S.2.).

...



L'analyse de toutes les rubriques de (S.2.) fournit grâce à la décomposition des algorithmes, un ensemble (S.3.) de rubriques d'entrée. Les ensembles (S.1.) et (S.3.) ne sont pas obligatoirement disjoints. L'ensemble (P), obtenu par union de (S.1.) et (S.3) réduction des éléments communs constitue l'ensemble des rubriques d'entrée du système.

Nous disposons donc d'un ensemble de rubriques élémentaires que nous devons caractériser par les points 1.4.1.2. , 1.4.1.3. et 1.4.1.4.

...

1.4.1.2. Détermination de la structure d'information associée aux données externes.

L'utilisation de COBOL ou PL1 pour la description des structures d'information associées aux données externes en entrée ou en sortie du système nous amène à faire les constatations suivantes :

1.4.1.2.1. Les descriptions des structures d'information dépendent de la structure de présentation des données externes. Cette dépendance nécessite une nouvelle description à chaque structure de présentation. Il est certain que cela entraîne de nombreuses descriptions, ce qui alourdit énormément les programmes. Sur ce point, notre conception se rapproche des systèmes de banques de données qui rendent indépendantes les structures d'information [24]. Cependant ces systèmes de banques de données introduisent des relations entre données dans la structure de présentation (ou organisation physique) ce qui impose une conception d'organisation de fichiers complexe difficile à maintenir.

Le principe que nous avons adopté est basé sur une analyse des données en entrée et en sortie qui nous conduit à définir des groupes de données que l'on appellera des groupes d'entrée ou groupes de sortie (groupes logiques externes) indissociables sémantiquement et pouvant être communs à plusieurs fichiers en entrée ou en sortie du système.

1.4.1.2.2. Les descriptions de structures d'information ne décrivent que des structures arborescentes simples, ce qui oblige l'utilisateur à employer des subtilités de description et de programmation très lourdes lorsqu'il désire manipuler des structures plus complexes :(graphes non arborescents).

Avec l'introduction des groupes d'entrée et de sortie, nous pourrons utiliser des structures très complexes résultant d'une analyse des données externes en décrivant les liens structuraux dans des tables de décision ou de sélection associées à la description des groupes d'entrée ou de sortie.

...

1.4.1.2.3. La définition des rubriques des articles ne permet pas de décrire un certain nombre de résultats de l'analyse des fichiers externes : présence facultative ou obligatoire de la donnée, condition de vraisemblance, de compatibilité.

1.4.1.3. Détermination de la structure de présentation des informations.

Nous étudierons au chapitre 2 les relations entre l'organisation logique et l'organisation physique (ou structure de présentation) des fichiers externes. Nous montrerons notamment qu'il est possible de déterminer par une technique d'analyse un ensemble de relations caractérisant la structure d'information des fichiers externes [24].

1.4.2. INSTRUCTIONS D'ENTREE/SORTIES DE FICHIERS DU LANGAGE CIVA.

Aux instructions CIVA définies pour effectuer tous les traitements en mémoire centrale, nous adjoignons les instructions d'Entrées/Sorties suivantes :

- "LIREXT" (resp. ECRIREXT) qui sont les seules instructions d'Entrées (resp. de Sorties) permettant d'effectuer des lectures (resp. des écritures) de fichiers en format externe.

- "ECRIRE" est une instruction d'Entrées/Sorties de fichiers uniquement en format interne.

Ces deux groupes d'instructions seront définis avec différentes options possibles, compatibles avec le système de gestion de fichiers sous SIRIS 7 (CII 10 070) [10]. En particulier, selon le mode de traitement des fichiers, le type de périphérique, le format des enregistrements, ces options peuvent être déclarées implicitement ou explicitement.

Ces instructions CIVA permettront d'effectuer des opérations d'Entrées/Sorties de base.

...

1.4.3. META-MODULES ET META-FONCTIONS DE BASE DU META-LANGAGE CIVA.

Le métalangage CIVA [6] doit répondre à trois préoccupations essentielles :

1.4.3.1. Permettre la conception et la programmation en mode déclaratif, en utilisant des méta-modules, des méta-fonctions standard .

1.4.3.2. Donner la possibilité de définir aisément d'autres méta-objets, notamment des services.

1.4.3.3. Permettre la modification aisée et rapide des programmes.

L'utilisation des méta-modules d'Acquisition et d'Edition se fera par un appel du nom du méta-module avec des paramètres effectifs. La conception et la programmation dans le méta-langage de CIVA permettront à l'utilisateur de mieux appréhender la logique de l'application. En effet, dans la phase d'analyse générale, il lui sera possible de décrire une Acquisition ou une Edition sans connaître précisément les fichiers externes, cette description se poursuivra de manière évolutive et modulaire jusqu'à la phase de programmation généralement, l'emploi de méta-module évitera de mener l'analyse jusqu'au niveau de la programmation (emploi de méta-modules standard). Par ailleurs, la formulation des dossiers d'analyse s'effectuera dans un langage "évolutif" (METACIVA vers CIVA) de la conception à la programmation. Il est donc certain que l'on obtiendra une meilleure "productivité" dans la réalisation des programmes.

1.5. RAPPELS SUR LES OBJETS DE CIVA.

Donnons auparavant les définitions fondamentales du projet CIVA que nous utiliserons par la suite.

Module : c'est une unité de description d'actions figurant dans la bibliothèque des modules d'une application. Un module est utilisable par tous les autres modules de l'application. Un appel de module se fera par le nom de ce module.

...

Classe : c'est une description d'informations. Une classe figure dans la bibliothèque des classes de l'application sous son nom, l'organisation entre les différentes déclarations s'effectuant par la relation utilise.

Application : une application est liée au niveau intermédiaire entre les objets CIVA (modules, classes, ...) et le système d'exploitation. C'est un niveau où il sera possible de donner un certain nombre d'options implicites (constantes, types, descriptions de fichiers, protections, tables, ...).

Objets autonomes : c'est un ensemble d'"outils software" évitant à l'utilisateur de contrôler et même de connaître la séquence d'instructions à exécuter (par exemple méta-module).

Méta-module : objets autonomes pouvant être écrits par l'utilisateur. Ils dépendent de paramètres et en particulier de fichiers.

Nous distinguerons les objets CIVA de base et les objets METACIVA.

Compilation des objets CIVA.

Rappelons brièvement que la compilation se déroule en deux phases :

. la codification et la génération du module objet.

. La codification est chargée de traiter les relations de classes "utilise" [5] [1], les problèmes de transformation des noms utilisés dans le programme source en adresses relatives d'un module objet et d'identifier chacune des instructions. Cette phase de codification aboutit à la génération d'une chaîne codée.

...

. la génération, à partir de la chaîne codée, produit une zone de constantes appelée module.

Compilation des objets METACIVA. [6][4]

Dans la réalisation proposée par le projet, un méta-traitement sera effectué pendant la codification, ce qui permettra aux méta-modules d'accéder, par l'intermédiaire de méta-fonctions standard, aux descriptions des objets CIVA (leur type, leur structure, leur taille ...).

CHAPITRE 2

ASPECTS THEORIQUES SUR LA STRUCTURE D'INFORMATION

utilisée pour les fichiers externes de CIVA.

2.1 Généralités

Notre but a été de fournir aux utilisateurs un langage relativement simple permettant de décrire facilement les différentes phases de la mise en oeuvre d'une Acquisition (contrôles, conversions,...) et d'une Edition (édition, impression ...) de fichiers externes. Nous nous sommes alors préoccupés de fournir un ensemble d'outils software qui permette de passer aisément de l'analyse de l'Acquisition ou de l'Edition à des modules exécutables par l'ordinateur.

Un aspect important de ce travail a été d'étudier la manière de définir les objets externes indépendamment de la structure de présentation des données sur les supports physiques. En effet, dans les traitements classiques d'informatique de gestion, la préoccupation essentielle de l'utilisateur est de sélectionner un critère de présentation des articles de telle sorte que le traitement en ordinateur soit le plus efficace possible. Il s'avère alors qu'avec cette conception, il est difficile de décrire les résultats de l'analyse des informations en vue de l'Acquisition ou de l'Edition.

L'étude que nous présentons dans ce chapitre utilise de nombreux résultats de C. DELOBEL [24] et F. PECCOUD [25] . Cependant, ces travaux ayant été effectués essentiellement dans une optique Banques de données nous avons donc pensé utile d'adapter ces résultats à nos problèmes.

2.2 Définitions préliminaires

2.2.1. Définition d'une notion

Définition 1 : Une notion N est un concept défini par l'utilisateur auquel on peut associer un ensemble de valeurs.
Une valeur désignée par "n" sera appelée réalisation de la notion N.

Une réalisation n de la notion N peut être une chaine de caractères numériques, alphabétiques ou alphanumériques.

Définition 2 : On appelle caractérisation d'une notion N une "propriété" définie sur l'ensemble des réalisations.
Une notion N pourra alors être définie
- soit par l'énumération de ses réalisations.
N sera dite définie en extension
- soit par l'expression de caractérisations :
N sera définie alors en compréhension.

Exemples :

Exemple 1 : notions définies en extension.

Notion	Ensemble de réalisations
"Code numérique d'état matrimonial"	{0, 1, 2, 3}
"Code alphabétique d'état matrimonial"	{CELIBATAIRE, MARIE, VEUF, DIVORCE}
"Code numérique des départements français"	{01, 02, 03, ——————, 99}

Exemple 2 : notions définies en compréhension.

Dans cet exemple, nous introduirons comme propriété la longueur d'une réalisation notée : $\ell g\,(n)$.

Notion	Ensemble de réalisations
"Code numérique des départements français"	$\{ n \in \mathbb{N} \ / \ell g\,(n) \leq 2 \text{ ET } 01 \leq n \}$

L'analyse du contenu du concept permet donc de définir formellement une notion soit en extension, soit en compréhension.

2.2.2. Notation des notions et des réalisations

Précisons de suite, que l'ensemble des notions que nous noterons $\mathcal{N}$ sera fini ou au plus dénombrable.

Les notions seront identifiées par des noms en lettres majuscules : NUMERO, PRIX, AGE, TOT, etc... qui pourront être éventuellement indicés par des lettres minuscules entre crochets pour faciliter la compréhension.

Les réalisations de notions seront identifiées par des noms en lettres minuscules éventuellement indicés, par exemple numéro (i), numéro (j) appartiennent à la notion NUMERO.

Si une notion est déjà indicée, les réalisations posséderont l'indice de notion suivi de l'indice propre à la réalisation. Par exemple ; soit TOT [i] une notion indicée, alors une réalisation courante sera identifiée par tot (i,j).

Remarques :

1 - L'ensemble des réalisations d'une notion est fini, même si à priori on ne connait pas le nombre de ses éléments.

2 - Toute notion pourra posséder une réalisation particulière "e" correspondant à une réalisation absente ou inconnue. Par exemple, lors d'une facturation d'articles, il peut arriver qu'une quantité commandée soit absente, ce qui correspond à une zone blanche sur une carte perforée, ou soit invraisemblable, ce qui peut être considéré comme code inconnu.

2.2.3. Notion élémentaire - Notion structurée

Soient ;

- A, B, C une suite de notions de $\mathcal{N}$ supposées définies en extension pour faciliter la compréhension.

$A = \{a(1), a(2), a(3), \_\_\_\_\_, a(i), \_\_\_\_\_, a(p)\}$

$B = \{b(1), b(2), b(3), \_\_\_\_\_, b(j), \_\_\_\_\_, b(q)\}$

$C = \{c(1), c(2), c(3), \_\_\_\_\_, c(k), \_\_\_\_\_, c(\ell)\}$

- l'opération concaténation sur l'ensemble des réalisations des notions notée '.' ;

Définition 3 : L'opération produit de notions sur $\mathcal{N}$ notée 'x' est définie par :

1 - $A \times B = \{a_{(i)} . b_{(j)} \;/\; a(i) \in A, b(j) \in B\}$

2 - $A \times B \in \mathcal{N}$.

Remarques : 1 - L'opération produit est associative.

2 - Lorsque l'on itère n fois l'opération produit à une notion nous parlerons de produit d'ordre n.

Définition 4 : Une notion E de $\mathcal{N}$ est dite élémentaire, si d'après la connaissance qu'en a l'utilisateur, elle ne peut pas être obtenue par produit de deux ou plusieurs autres notions de $\mathcal{N}$.

Exemples :

1 - La notion "codes numériques des départements français" est une notion élémentaire.

2 - Par contre, si l'on considère la notion "code de numéro de sécurité sociale", elle est obtenue par produit d'autres notions : "code du sexe", "code année de naissance", "code mois de naissance", "code de département", "code de commune", "code du numéro de registre dans la commune".
Ce n'est donc pas une notion élémentaire.

Définition 5 : Une notion S de N est dite structurée, si elle est produit de deux ou plusieurs autres notions de $\mathcal{N}$.

$S = A \times B$ où $C = \{c(k) \;/\; 1 \leq k \leq \ell\}$
et $c(k) = a(i) . b(j)$; $a(i) \in A$ et $b(j) \in B$

Exemple :

Notion "numéro de sécurité sociale" notée SECSOC.

Alors, SECSOC = X x A x M x D x C x R où,

X = "code numérique de sexe"
A = "code numérique d'année de naissance"
M = "code numérique de mois de naissance"
D = "code numérique de département"
C = "code numérique de commune"
R = "code numérique de registre"

sont les notions composant la notion structurée SECSOC. Une réalisation de SEC SOC sera par exemple, 1.14.02.54.157.049, que l'on écrira par la suite 1 14 02 54 157 049 lorsqu'il n'y aura pas d'ambiguité possible.

Considérons les opérations '$\cup$' et '$\cap$' sur l'ensemble des réalisations de notion.

Définition 6 : L'opération réunion de notions sur $\mathcal{R}$ notée '$\cup$' est définie par :

1) $A \cup B = \{a(1), a(2), \ldots, a(p)\} \cup \{b(1), b(2), \ldots, b(q)\}$

2) $A \cup B \in \mathcal{R}$

Définition 7 : L'opération intersection de notions sur $\mathcal{R}$ notée '$\cap$' est définie par:

1) $A \cap B = \{a(1), a(2), \ldots, a(p)\} \cap \{b(1), b(2), \ldots, b(q)\}$

2) $A \cap B \in \mathcal{R}$

Exemples : Dans l'exemple du numéro de sécurité sociale considérons $A = \{0,73\}$; $D = \{01,99\}$ alors :

$A \cup D = \{0, 1, \ldots, 73\} \cup \{1, 2, \ldots, 99\}$

$= \{0, 1, \ldots, 99\}$

$A \cap D = \{0, 1, \ldots, 73\} \cap \{1, 2, \ldots, 99\} = \{1, 2, \ldots, 73\}$.

2.2.4. Définitions des notions de fichier, d'article

Soit $\mathcal{E} = \{A[1], A[2], A[3], \ldots, A[n]\}$ un ensemble de notions élémentaires ou structurées.

Un fichier F est une notion définie par

$$F = (A[1] \cup A[2] \cup \ldots \cup A[i] \cup \ldots \cup A[n])^m$$

ce que l'on peut encore écrire :

$$F = \left(\bigcup_{i=1}^{i=n} A[i] \right)^m \qquad \text{avec } m \in \mathbb{N}$$

Remarque : Chaque réalisation de la notion fichier F est une réalisation d'une notion ou d'une réunion de notions de $\mathcal{E}$.

Un article T d'un fichier F est une notion dont l'ensemble des réalisations appartient à la notion F.

Exemple : Cet exemple que nous citons est extrait de C. DELOBEL [24]

Considérons l'ensemble $\mathcal{E} = \{P, C, S, H, M\}$ de notions définies par les concepts suivants :

P = "Nom de professeur d'un établissement scolaire"

C = "classe de l'établissement"

S = "salle de cours"

H = "heure de cours"

M = "matière enseignée".

Supposons que le Directeur des Etudes de l'établissement chargé de l'emploi du temps ait défini les notions associées aux concepts donnés ci-dessus, comme suit :

$P = \{$DURAND, DUPONT, MARTIN$\}$

$C = \{6, 5, 4, 3\}$,

$S = \{A1, A2, A3, B2\}$,

$H = \{8, 9, 10, 11, \ldots, 18\}$

$M = \{$INFO, MATH, FRANÇAIS$\}$

La figure suivante peut être considérée comme définissant une représentation d'un emploi du temps de l'établissement.

notions / réalisations	P	M	C	S	H
f(1)	MARTIN	MATH	3	B2	8
f(2)	DUPONT	INFO	6	A1	9
f(3)	DUPONT	INFO	5	A1	11
f(4)	DURAND	FRANÇAIS	5	A2	16
f(5)	DUPONT	INFO	6	A1	14
f(6)	DURAND	FRANÇAIS	3	A2	10
f(7)	MARTIN	MATH	4	A3	10

D'après nos définitions, cette représentation d'emploi du temps peut être vue comme un fichier F défini par

$$F = (P \times M \times C \times S \times H)^7$$

où le produit de notions constitue l'article unique $T = P \times M \times C \times S \times M$ du fichier F.

En d'autres termes ici, $F = \{f(1), f(2), \ldots, f(7)\}$ peut être défini par $F = (T)^7$ avec $T = P \times M \times C \times S \times H$.

Définition 8 : Opérateur de projection $\Pi_{\mathcal{S}}$

Soit $\mathcal{S}$ un sous-ensemble de $\mathcal{E}$, $\mathcal{S} = \{A[1], A[2], \ldots, A[p]\}$; avec $p \leq n$. La projection d'un fichier F par rapport

à $\mathcal{S}$ est une notion telle que ;

$\Pi_{\mathcal{S}}(F) = \{ (a(i,1), a(i,2), \ldots, a(i,p)) \ / \ \forall i, 1 \leq i \leq m \}$.

Dans la suite, pour ne pas alourdir les notations, nous désignerons par $F_{\mathcal{S}}$ la projection par rapport à $\mathcal{S}$ du fichier F

$F_{\mathcal{S}} = (\bigcup_{i=1}^{i=p} A[i])^m$, $m \in \mathbb{N}$ et par $f_{\mathcal{S}} = \Pi_{\mathcal{S}}(f)$ la projection d'une réalisation courante f de F par rapport à $\mathcal{S}$.

Exemple :

Reprenons l'exemple de l'emploi du temps. Si l'on considère comme ensemble de notions $\mathcal{S} = \{P, C\}$, alors le fichier $F_{\mathcal{S}}$ obtenu par projection de F sera constitué de la façon suivante :

$F_{\{P, C\}}$ =

notions / réalisations	P	C
$f_{\mathcal{S}}(1)$	MARTIN	3
$f_{\mathcal{S}}(2)$	DUPONT	6
$f_{\mathcal{S}}(3)$	DUPONT	5
$f_{\mathcal{S}}(4)$	DURAND	5
$f_{\mathcal{S}}(5)$	DURAND	3
$f_{\mathcal{S}}(6)$	MARTIN	4

Considérons maintenant l'ensemble de notions $\mathcal{S} = \{C, H\}$ alors $F_{\mathcal{S}}$ sera défini par :

$F_{\{C, H\}}$ =

notions / réalisations	C	H
$f_{\mathcal{S}}(1)$	3	8
$f_{\mathcal{S}}(2)$	6	9
$f_{\mathcal{S}}(3)$	5	11
$f_{\mathcal{S}}(4)$	5	16
$f_{\mathcal{S}}(5)$	6	14
$f_{\mathcal{S}}(6)$	3	10
$f_{\mathcal{S}}(7)$	4	10

Remarques : Si l'on considère l'ensemble de notions $\mathcal{U}$

$\mathcal{U} = \{ A[1] , A[2] , \ldots , A[q] , q \in n \}$

notons que l'opérateur de projection possède les propriétés suivantes :

1 - $F_{\mathcal{E}} = F$

2 - $\Pi_{\mathcal{U}}(F_{\mathcal{S}}) = F_{\mathcal{T}}$ où $\mathcal{T} = \mathcal{U} \cap \mathcal{S}$.

2.2.5. Etude de l'ensemble des notions $\mathcal{N}$

Définition 9 : Etant donnés les ensembles de notions $\mathcal{S}$, $\mathcal{U}$ de $\mathcal{N}$, nous dirons qu'il existe une relation fonctionnelle $m : F_{\mathcal{S}} \rightarrow F_{\mathcal{U}}$ si m satisfait aux conditions suivantes :

1 - m est une fonction surjective de $F_{\mathcal{S}}$ dans $F_{\mathcal{U}}$.

2 - pour tout $f \in F$: $m(f_{\mathcal{S}}) = f_{\mathcal{U}}$.

Exemples : Pour l'emploi du temps, l'utilisateur pourra définir les relations fonctionnelles suivantes :

m_1 : $F_{\{P\}} \longrightarrow F_{\{M\}}$: signifie qu'un professeur n'enseignera dans l'établissement qu'une matière.

m_2 : $F_{\{P,H\}} \rightarrow F_{\{C\}}$: à une heure donnée un professeur n'enseigne qu'à une seule classe.

m_3 : $F_{\{P,H\}} \rightarrow F_{\{S\}}$: à une heure donnée, un professeur n'enseigne que dans une salle.

m_4 : $F_{\{C,H\}} \rightarrow F_{\{P\}}$: à une heure donnée, une classe ne peut suivre les cours que d'un professeur.

m_5 : $F_{\{S,H\}} \rightarrow F_{\{P\}}$: à une heure donnée, une salle n'est occupée que par un professeur.

Ces exemples montrent que tout utilisateur disposant de l'ensemble des notions d'un fichier, définies par l'analyse du contenu des concepts, a la possibilité de caractériser naturellement le fichier de l'emploi du temps. Ces relations fonctionnelles qui permettent de donner une signification aux réalisations d'un fichier sont aussi appelées relations sémantiques par F. PECCOUD [25] .

Définition 10 :

Soient - $\mathcal{P}(\mathcal{R})$ l'ensemble des parties de $\mathcal{R}$

- $\mathcal{U}, \mathcal{V}$ deux éléments de $\mathcal{P}(\mathcal{R})$.

Nous définirons la relation binaire τ sur $\mathcal{P}(\mathcal{E})$ de la façon suivante :

$$\mathcal{U} \tau \mathcal{V} \Longleftrightarrow \exists m : F_{\mathcal{U}} \longrightarrow F_{\mathcal{V}}.$$

Exemple : La relation fonctionnelle m : $F_{\{P,H\}} \longrightarrow F_{\{S\}}$ définie précédemment définit la relation binaire $\{P,H\} \tau \{S\}$

Remarque : L'opérateur de projection $\Pi_{\mathcal{V}} : F_{\mathcal{U}} \longrightarrow F_{\mathcal{V} \cap \mathcal{U}}$ est une relation fonctionnelle. Ceci résulte de la propriété 2 énoncée en 2.2.4.

2.2.5.1. Propriétés de la relation binaire τ :

Nous ne ferons qu'énoncer les résultats, pour les démonstrations on pourra se reporter à C. DELOBEL [24] .

1 - Soient $\mathcal{U}, \mathcal{V}, \mathcal{M}$ une suite d'éléments de $\mathcal{P}(\mathcal{E})$

1 - Si $\mathcal{U} \tau \mathcal{V}$ et $\mathcal{V} \tau \mathcal{M}$ alors $\mathcal{U} \tau \mathcal{M}$

2 - $\mathcal{U} \tau \mathcal{U}$

3 - Si $\mathcal{V} \subset \mathcal{U}$ alors $\mathcal{U} \tau \mathcal{V}$

4 - Si $\mathcal{U} \tau \mathcal{V}$ et $\mathcal{U} \tau \mathcal{M}$ alors $\mathcal{U} \tau (\mathcal{V} \cup \mathcal{M})$

2.2.5.2. Partitions sur F et relation binaire τ

Définition 11 : Soit $\tau_{\mathcal{S}}$ une relation définie sur le fichier F de la manière suivante :

$\mathcal{S} \subset \mathcal{E}$, $\mathcal{S} = \{ A[1], A[2], \ldots, A[p], p \leq n \}$

alors : f (i) $\tau_{\mathcal{S}}$ f (j) $\Leftrightarrow$ a (i,1) = a(j,1), a(i,2) = a (j,2), ..., a (i,m) = a (j,m).

La relation $\tau_{\mathcal{S}}$ définit une partition sur F.

Pour la suite, nous utiliserons les notations suivantes :

- $[F]_{\tau_{\mathcal{S}}}$ = une classe d'équivalence

$F/\tau_{\mathcal{S}}$ = l'ensemble des classes d'équivalence.

Exemple :

La relation $\tau_{\{C\}}$, définie dans l'emploi du temps, détermine une partition sur le fichier F en quatre classes d'équivalence données ci-après.

notions	P	M	C	S	H
réalisations					
f(1)	MARTIN	MATH	3	B2	8
f(6)	DURAND	FRANÇAIS	3	A2	10
f(7)	MARTIN	MATH	4	A3	10
f(3)	DUPONT	INFO	5	A1	11
f(4)	DURAND	FRANÇAIS	5	A2	16
f(5)	DUPONT	INFO	6	A1	14
f(2)	DUPONT	INFO	6	A1	9

Proposition : si $\mathcal{U} \tau \mathcal{V}$ alors $F/\tau_{\mathcal{U}} \equiv F/\tau_{\mathcal{U} \cup \mathcal{V}}$

Par conséquent, si le couple $(\mathcal{U}, \mathcal{V})$ vérifie la relation binaire τ , alors les partitions associées à $\tau_{\mathcal{U}}$ et $\tau_{\mathcal{U} \cup \mathcal{V}}$ sont identiques.

2.2.5.3. Définition d'un préordre sur $\mathcal{P}(\mathcal{E})$ par la relation binaire τ

La relation binaire τ , réflexive et transitive (2.2.5.1) définit un préordre sur $\mathcal{P}(\mathcal{E})$.

Soit τ^* la relation binaire définie comme suit :

$\mathcal{U}, \mathcal{V}$ éléments de $\mathcal{E}$, $\mathcal{U} \tau \mathcal{V} \Leftrightarrow \mathcal{V} \tau \mathcal{U}$

Alors, τ^* définit sur $\mathcal{P}(\mathcal{E})$ une partition en classe d'équivalence.

2.2.6. Relations élémentaires

Définition 11 : Soient $\mathcal{U}, \mathcal{V} \in \mathcal{P}(\mathcal{E})$

Nous dirons que la relation $\mathcal{U} \tau \mathcal{V}$ est une relation élémentaire si:

$\forall J \in \mathcal{P}(\mathcal{E}), J \subset \mathcal{U}$ et $J \neq \mathcal{U}$ alors $J \tau \mathcal{V}$ n'est pas une relation

Exemples : Dans l'emploi du temps, la relation binaire $\{P\} \tau \{M\}$ est une relation élémentaire. Par contre, la relation $\{P,M\} \tau \{M\}$ n'est pas une relation élémentaire car $\{M\} \tau \{M\}$ est une relation (propriété 2 de 2.2.5.1).

Remarques :

1 - Si $\mathcal{V} \subset \mathcal{U}$ alors, $\mathcal{U} \tau \mathcal{V}$ (d'après la propriété 3 de 2.2) peut être une relation élémentaire car $\mathcal{V} \tau \mathcal{V}$ est une relation.

2 - L'opérateur projection définissant une relation fonctionnelle, ne peut donner lieu à une relation élémentaire.

Ainsi, l'ensemble des relations élémentaires que l'utilisateur peut exprimer simplement ne se déduisent pas de l'analyse d'une procédure de traitement mais de l'observation physique de l'application. Dans l'exemple de l'emploi du temps, on sait qu'un enseignant ne peut être à une heure donnée dans plusieurs salles de cours ! Nous considérons que l'ensemble des relations élémentaires met en évidence une propriété intrinsèque d'un fichier.

Exemple : L'ensemble des relations élémentaires définies, dans le cas de l'emploi du temps, par l'utilisateur sera le suivant :

$\ell_1 : \{P\} \tau \{M\}$
$\ell_2 : \{P,H\} \tau \{C\}$
$\ell_3 : \{P,H\} \tau \{S\}$
$\ell_4 : \{H,S\} \tau \{P\}$
$\ell_5 : \{H,S\} \tau \{C\}$
$\ell_6 : \{H,C\} \tau \{P\}$
$\ell_7 : \{H,C\} \tau \{S\}$

Il est alors évident, que l'on peut ainsi exhiber beaucoup de relations élémentaires. Ainsi, cet ensemble peut comporter de nombreuses relations redondantes en ce sens que certaines relations élémentaires peuvent se déduire à partir d'autres. Il importe donc de rendre cohérent cet ensemble. C'est ce que nous allons nous attacher à faire.

Proposition 5 :

Soient deux relations élémentaires : $\mathcal{S} \tau \mathcal{V}$ et $\theta \tau \mathcal{U}$ telles que

1 - $\mathcal{U} \subseteq \mathcal{S}$

2 - $\forall J \in \mathcal{P}(\mathcal{E}), J \in \theta \cup (\mathcal{S} - \mathcal{U}), J \tau \mathcal{U}$ n'est pas une relation

Alors $\theta \cup (\mathcal{S} - \mathcal{U}) \tau \mathcal{V}$ est une relation élémentaire.

(la démonstration est donnée par C. DELOBEL [24]).

Exemples :

Appliquons la proposition 5 aux relations élémentaires suivantes :

1) $\ell_1 : \{P\} \tau \{M\}$,
$\ell_6 : \{H,C\} \tau \{P\}$,

nous en déduisons la relation élémentaire $\{H,C\} \tau \{M\}$ qui n'avait pas été envisagée.

2) $\ell_2 : \{P,H\} \tau \{C\}$,
$\ell_7 : \{H,C\} \tau \{S\}$,

Alors, nous obtenons par application de la proposition 5

$\ell_3 : \{P,H\} \tau \{S\}$, relation déjà exhibée.

Ces deux exemples montrent que l'ensemble des relations élémentaires $\ell_1, \ell_2, \ldots, \ell_7$, pour caractériser intrinsèquement l'emploi du temps ne doit comporter qu'un ensemble de relations élémentaires minimum, non redondantes, à partir duquel, l'on peut générer toutes les autres.

Remarque : Si dans la proposition 5, l'on a $\mathcal{U} = \mathcal{S}$ alors nous retrouvons la propriété de transitivité énoncée en 2.2.5.1.

2.2.7. Définitions de couvertures, couvertures minimales, poids d'une couverture d'un ensemble

soient - un ensemble Φ de relations élémentaires

$\Phi = \{\ell_1, \ell_2, \ldots, \ell_n\}$

- une transformation T qui à toute paire (ℓ_i, ℓ_j), $\ell_i \in \Phi$, et $\ell_j \in \Phi$ fait correspondre $\ell_m \in \Phi$ si

ℓ_m est obtenue par la proposition 5, telle que

$$T((\ell_i, \ell_j)) = \ell_m$$

Alors, pour tout ensemble Φ_0, $\Phi_0 \subset \Phi$, de relations élémentaires, nous aurons

$$T(\Phi_0) = \{ \ell_m / T((\ell_i, \ell_j)) = \ell_m ; \ell_i, \ell_j \in \Phi_0 \}$$

Nous définirons la <u>fermeture de Φ</u> , que l'on notera Φ_*

$$\Phi_* = \bigcup_{n \geq 0} T^n(\Phi) \quad \text{ou} \quad T^0(\Phi) = \emptyset$$

<u>Exemple</u> : Dans le cas de l'emploi du temps, désignons par $\Phi = \{\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6, \ell_7\}$, l'ensemble des relations élémentaires.

Il est alors facile, d'en déduire de la définition de la fermeture de Φ, $\Phi_* = \Phi \cup \{\ell_8, \ell_9\}$

avec $\ell_8 : \{H,C\} \tau \{M\}$

$\ell_9 : \{H,S\} \tau \{M\}$

Nous appellerons <u>couverture d'un ensemble</u> Φ , tout ensemble Φ_1, $\Phi_1 \subset \Phi$ tel que :

$$\Phi = \bigcup_{n \geq 0} T^n(\Phi_1) \text{ et } T^0(\Phi_1) = \Phi_1$$

<u>Exemple</u> : L'ensemble $\Phi = \{\ell_i / 1 \leq i \leq 7\}$ est, d'après la définition de la couverture, une couverture de $\Phi_* = \{\ell_i / 1 \leq i \leq 9\}$ puisque $\Phi_* = \bigcup_{n \geq 0} T^n(\Phi)$, par construction.

Nous appellerons <u>couverture minimale d'un ensemble</u> Φ , tout ensemble Φ_1, $\Phi_1 \subset \Phi$ tel que :

1 - Φ_1 est une couverture de Φ ,

2 - Quelque soit Φ_2, $\Phi_2 \subset \Phi_1$ et $\Phi_2 \neq \Phi_1$

alors, Φ_2 n'est pas une couverture de Φ .

<u>Remarque</u> : nous montrerons ultérieurement que l'ensemble $\Phi = \{\ell_i / 1 \leq i \leq 7\}$ des relations élémentaires définies par l'utilisateur n'est pas une couverture minimale de $\Phi_* = \{\ell_i / 1 \leq i \leq 9\}$.

Nous définirons le <u>poids d'une couverture</u> d'un ensemble $\Phi_1 = \{\ell_{j,1}, \ell_{j,2}, \ldots, \ell_{j,q}\}$ comme une fonction P à valeurs entières de façon que :

$$P(\Phi_1) = \sum_{k=1}^{k=q} P(\ell_{j,k})$$

Pour obtenir la (ou les) couverture (s) minimale (s) de poids minimum, c'est-à-dire le (ou les) ensemble (s) de relations élémentaires possèdant le moins d'éléments et à partir desquels on peut retrouver toutes les relations élémentaires ; il est nécessaire d'exhiber la fermeture Φ_* de l'ensemble de relations élémentaires Φ , de rechercher les couvertures minimales, et enfin parmi celles-ci de ne retenir que celles où la fonction de poids est telle que pour tout ℓ_i de Φ_* , $P(\ell_i) = 1$ est minimale.

2.3. <u>Algorithme de recherche des couvertures minimales de poids minimum</u>

L'algorithme que nous présentons se rattache aux méthodes de séparation et d'évaluation à graphe arborescent et dichotomique de ROY [15] que l'on appelle aussi "branch and bound".

Pour utiliser cette méthode de séparation et d'évaluation, nous devons introduire la définition de graphe biparti ci-après.

Soient :
- Φ un ensemble de relations élémentaires
- $\Phi_* = \{\ell_i / 1 \leq i \leq n$, la fermeture de Φ
- T la transformation définie en 2.2.7.
- $L = \{ (\ell_j, \ell_k) / \exists \ell_m \in \Phi_*$ tel que $T((\ell_j, \ell_k)) = \ell_m \}$

Nous définirons le graphe biparti (ROY [15]), $H = (L, \Phi_*, T)$ à partir de la transformation T par la relation binaire suivante :

$$\forall \ell_m \in \Phi_* \ , (\ell_j, \ell_k) \, T \, \ell_m \iff T((\ell_j, \ell_k)) = \ell_m$$

L'algorithme d'énumérations des couvertures minimales de poids minimun dans le graphe biparti $H = (L, \Phi_*, T)$ se compose d'une succession de phases que nous décrivons maintenant.

2.3.1. Description de l'algorithme

I - Soient - $+\infty$ le poids affecté à l'ensemble Φ_* (nombre supérieur ou égal au nombre d'éléments ℓ_i de Φ_*)

- A_0 = ensemble des ℓ_i de Φ_* tels qu'il n'existe pas de paire d'éléments de Φ_*, (ℓ_j, ℓ_p) où $T((\ell_j, \ell_p)) = \ell_i$

  Notons que les éléments de A_0 seront alors obligatoirement dans les couvertures minimales.

- B_0 = ensemble des ℓ_i de $\Phi_* - A_0$ tels qu'il n'existe pas d'éléments ℓ_j et ℓ_p de Φ_* où $T((\ell_i, \ell_j)) = \ell_p$.

  Notons qu'ici, les éléments de B_0 n'appartiendront à aucune des couvertures minimales.

- $L_0 = \Phi_* - A_0 - B_0$.

Si $L_0 = \emptyset$, A_0 sera la seule couverture minimale de poids minimum. Dans ce cas, nous calculons son poids et nous l'éditons.

Si $L_0 \neq \emptyset$, nous selectionnons le premier élément de L_0 dans l'ordre lexicographique, soit $\ell_{j,0}$.

Nous initialisons une pile notée IPILE avec le quadruplet suivant :

- hauteur de la pile associée à A_0
- hauteur de la pile associée à B_0
- élément $\ell_{j,0}$
- L_0

on passe ensuite à la phase II.

II - Considérons la quadruplet défini au cours de la phase 1 (hauteur de la pile associée à A_p, hauteur de la pile associée à B_p, $\ell_{j,p}, L_p$), lorsque nous arrivons à la phase II.

Posons : $A_{p+1} = A_p \cup \{\ell_{j,p}\}$

$$B_{p+1} = B_p \cup \{\ell_i \in \Phi_* - A_{p+1} - B_p \mid \exists\, \ell_j \in A_{p+1},\ \ell_p \in A_{p+1},\ T((\ell_j, \ell_p)) = \ell_i\}$$

aller en III.

III - Considérons le couplet (A_{p+1}, B_{p+1}) et posons :

$$L_{p+1} = \Phi_* - A_{p+1} - B_{p+1}$$

Alors :

- Si $L_{p+1} = \emptyset$, aller en IV
- Si $L_{p+1} = \emptyset$, nous selectionnons le premier élément $\ell_{j,p+1}$ dans l'ordre lexicographique des éléments de L_{p+1}, on empile le quadruplet obtenu (hauteur de la pile associée à A_{p+1}, hauteur de la pile associée à B_{p+1}, $\ell_{j,p+1}$, L_{p+1}) sur IPILE et aller en II.

IV - Soit A_{p+1}, l'ensemble construit lorsqu'on arrive à la phase IV. Calculons son poids.

IV.1 Si le poids de A_{p+1} est strictement supérieur au poids minimum obtenu jusque ici, nous en déduisons que : A_{p+1} n'est pas une couverture minimale de poids minimum. Nous l'abandonnons et nous passons en IV.4.

IV.2 Si le poids de A_{p+1} est égal au poids minimum obtenu à ce stade de la procédure, alors A_{p+1} est une couverture minimale de poids minimum que nous gardons.
aller en IV.4.

IV.3 Si le poids de A_{p+1} est strictement inférieur au poids minimum obtenu à ce stade ; nous éliminons les couvertures minimales gardées jusque là et nous faisons : poids de A_{p+1} = nouveau poids minimum.
Nous conservons A_{p+1} et nous passons en IV.4.

IV.4 Si la pile IPILE est vide, alors toutes les couvertures minimales de poids minimum de Φ_* ont été obtenues.
aller à VI.
Sinon, nous devons prendre le quadruplet se trouvant sur la pile (hauteur de la pile associée à A_q, hauteur de la pile associée à B_q, ℓ_j, q, L_q lorsque nous en arrivons en IV.4.

Nous envisagerons deux cas :

- soit $\ell_{j,q}$ a déjà été considéré dans A_q ; alors il est marqué en utilisant une notation négative de $\ell_{j,q}$. Il nous faut alors envisager l'autre possibilité. Aller en V.

- Soit $\ell_{j,q}$ n'est pas marqué ; nous allons en conséquence envisager d'une part $\ell_{j,q}$ dans A_q, puis $\ell_{j,q}$ dans B_q

Pour cela, nous reprenons un autre quadruplet dans la pile IPILE si elle n'est pas vide. Aller en IV.4.

V - Ajoutons sur IPILE le quadruplet suivant :
hauteur de la pile associée à A_q, hauteur de la pile associée à B_q, $\ell_{j,q}$ marqué, L_q.

Posons :

$$B'_q = B_q \cup \{\ell_{j,q}\}$$

$$A'_{q+1} = A_q \cup \{\ell_i \in \Phi_* - B'_q - A_q\}$$ tels que quels que soient ℓ et ℓ' éléments de Φ_* alors
$T((\ell_{j,q}, \ell)) = \ell_i$ et $T((\ell_i, \ell')) = \ell_{j,q}$
soient les deux seules relations avec ℓ_i et $\ell_{j,q}$ comme deuxième membre.

$$B_{q+1} = B'_q \cup \{\ell_i \in \Phi_* - A_{q+1} - B'_q\}$$ tels que il existe
$\ell_j \in A_{q+1}$
$\ell_p \in A_{q+1}$ tels que $T((\ell_j, \ell_p)) = \ell_i$

Aller en III.

VI - Fin de l'algorithme.

Remarques

ROY [15] montre que l'on peut associer à cette exploration par séparation et évaluation, une arborescence. Cette méthode heuristique a aussi été développée dans un cadre plus général par HERVE [16]. Nous n'avons fait qu'adapter une méthode à notre problème, c'est pourquoi on pourra se reporter pour les démonstrations à ces auteurs.

2.3.2 Procédure de résolution

Pour la clarté de l'exposé nous donnons le programme en [annexe 1]. Cependant, nous montrerons sur l'exemple de l'emploi du temps, la détermination des couvertures minimales de poids minimum.

Exemple : De l'ensemble des relations élémentaires mis en évidence par l'utilisateur $\Phi = \{\ell_i / 1 \leq i \leq 7\}$, nous en déduisons la fermeture $\Phi_* = \{\ell_i / 1 \leq i \leq 9\}$. D'après la définition de la transformation T, nous en déduisons :

$$L = \{(\ell_3, \ell_5), (\ell_2, \ell_7), (\ell_5, \ell_6), (\ell_2, \ell_1), (\ell_4, \ell_7), (\ell_3, \ell_6), (\ell_1, \ell_6), (\ell_1, \ell_4)\}$$

Le graphe H = (L, Φ_*, T) défini par la relation binaire suivante :

$(\ell_i, \ell_j) \in L, \ell_m \in \Phi_* \quad (\ell_i, \ell_j)\ T\ \ell_m \Leftrightarrow \ell_m = T((\ell_i, \ell_j))$

est représenté par la figure 1.

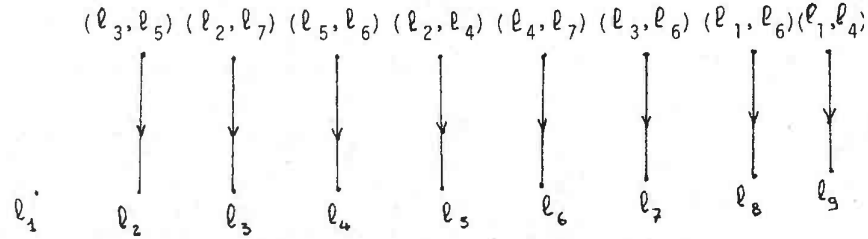


Figure 1

Prenons pour poids de l'ensemble Φ_*, cardinal (Φ_*) c'est-à-dire 9. Après la détermination des points de Φ_* appartenant à $A_0 = \{\ell_1\}$ et à $B_0 = \{\ell_8, \ell_9\}$, le graphe H se réduit au sous-graphe représenté sur la figure 2.

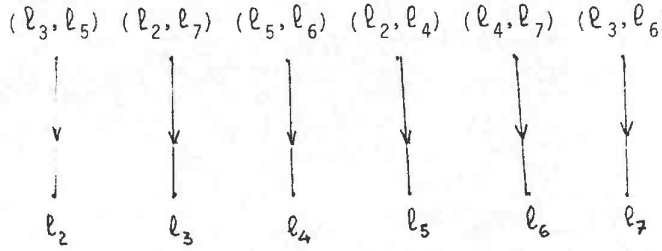
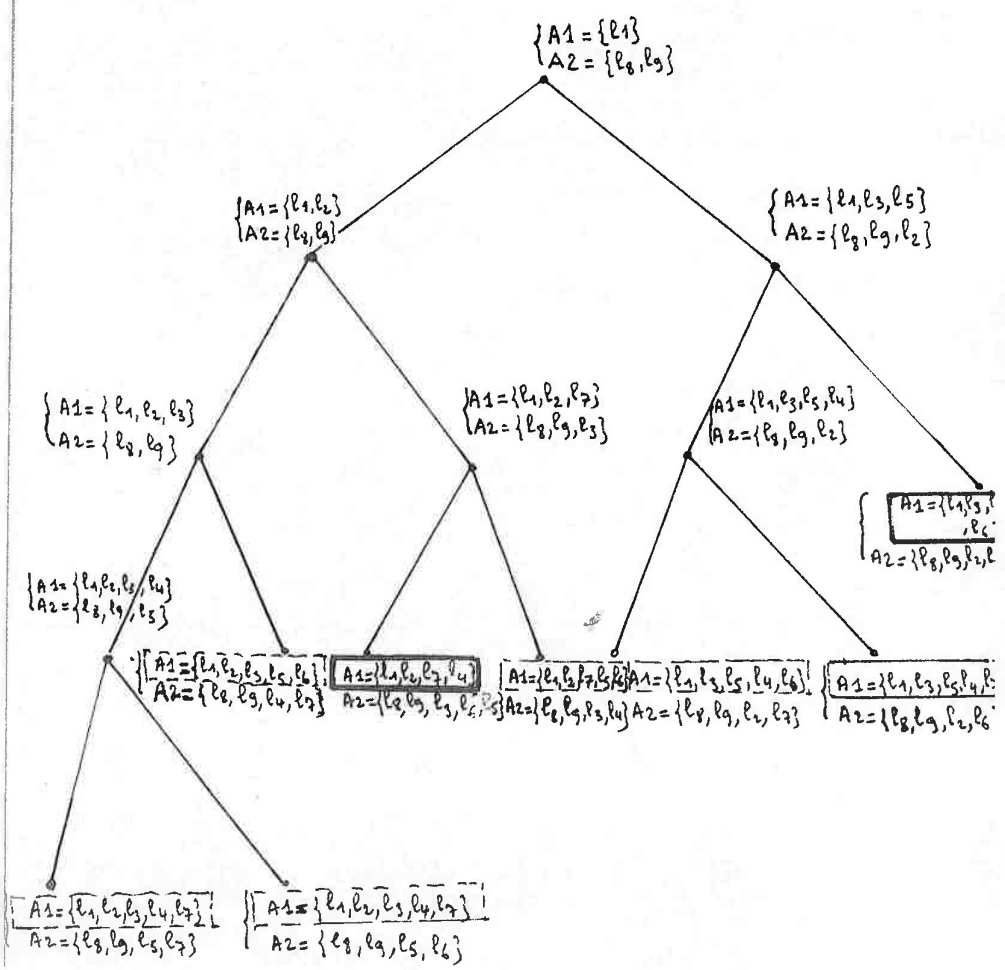


Figure 2

L'arborescence associée à l'algorithme est la suivante :


A1 = {l1}
A2 = {l8, l9}
A1 = {l1, l2}
A2 = {l3, l6}
A1 = {l1, l3, l5}
A2 = {l8, l9, l2}
A1 = {l1, l2, l3}
A2 = {l8, l9}
A1 = {l1, l2, l7}
A2 = {l8, l9, l3}
A1 = {l1, l3, l5, l4}
A2 = {l8, l9, l2}
A1 = {l1, l2, l3, l4}
A2 = {l8, l9, l5}
A1 = {l1, l2, l3, l5, l6}
A2 = {l8, l9, l4, l7}
A1 = {l1, l2, l7, l4}
A2 = {l8, l9, l3, l5, l6}
A1 = {l1, l3, l5, l4, l6}
A2 = {l8, l9, l2, l7}
A1 = {l1, l2, l3, l4, l7}
A2 = {l8, l9, l5, l7}
A1 = {l1, l2, l3, l4, l7}
A2 = {l8, l9, l5, l6}

Nous vérifions facilement qu'il y a deux couvertures minimales de poids minimum égal à 4 (encadrées en trait plein sur l'arborescence).

$$\begin{bmatrix} \{\ell_1, \ell_2, \ell_4, \ell_7\} \\ \{\ell_1, \ell_3, \ell_5, \ell_4\} \end{bmatrix}$$

Les ensembles encadrés en traits pointillés sont des couvertures minimales, mais pas de poids minimum.

Remarques

1 - Le programme d'énumération que nous avons écrit ne donne que les couvertures minimales de poids minimum. Il serait facile, en modifiant les tests sur les poids d'énumérer toutes les couvertures minimales de $\Phi_{\mathcal{R}}$.

2 - C. DELOBEL [24] montre aussi que l'ensemble des couvertures minimales de poids minimum conduit à une même représentation du préordre τ.

2.4. Aide à l'analyse et à la description des fichiers externes CIVA

Il est à remarquer que l'ensemble des relations élémentaires, de couvertures minimales de poids minimum définissent des propriétés intrinsèques du fichier ; aussi, est-il possible de penser qu'elles caractérisent la structure de l'information.

D'autre part, cette formalisation est introduite indépendamment des supports et de la présentation des données du fichier sur ceux-ci. C'est un aspect important, car nous donnons alors la possibilité à l'utilisateur de définir simplement les informations en effectuant une analyse cohérente des relations logiques des notions des fichiers, sans que cette analyse soit remise en cause à la programmation.

En effet, à partir d'un ensemble de relations élémentaires appartenant à une couverture minimale il est possible de construire un grand nombre d'organisations logiques.

Il est aussi intéressant de noter que pour tout support de fichier externe : paquet de cartes perforées, feuilles d'imprimante, l'utilisateur doit prévoir pour décrire une organisation logique de fichier, disposer d'un système d'adressage physique, (ces adresses devant être un ensemble totalement ordonné).

Exemples :

Supports	notion : Adresse
Carte perforée	"Numéro de colonne"
paquet de cartes perforées	"Numéro de carte, numéro de colonne"
feuille d'imprimante	"Numéro de ligne, numéro de colonne dans la ligne"
feuille d'imprimantes	"Numéro de page, numéro de la ligne, numéro de la colonne dans la ligne".

Ainsi, pour chaque support ordonné, l'adresse sera une notion généralement structurée dont chaque réalisation repérera une partie du support.

2.4.1. Groupe logique de notions

Considérant que les couvertures minimales de poids minimum caractérisent la structure d'information d'un fichier, il nous a paru naturel que l'utilisateur puisse donner une description du fichier, indépendamment d'une organisation logique, en décrivant simplement les relations élémentaires.

Considérons les ensembles $\mathcal{U}$ et $\mathcal{V}$ de notions de $\mathfrak{G}$ tels que

$$\mathcal{U} = \{U[1], U[2] - U[p]\}, \quad \mathcal{V} = \{V[1], V[2], -, V[n]\}$$

Une relation élémentaire τ : $\mathcal{U}\ \tau\ \mathcal{V}$

Un groupe logique de notions attaché à une relation élémentaire est défini par la notion :

$$U[1] \times U[2] \times \ldots \times U[p] \times V[1] \times V[2] \times \ldots \times V[n]$$

Exemple : Considérons la relation élémentaire $\ell_2 : \{P,H\}\ \tau\ \{C\}$ définie dans l'emploi du temps, nous lui associerons le groupe logique P x H x C.

Il est cependant important de noter que lors de la description des groupes logiques, il sera parfois indispensables soit de sélectionner une notion présente dans tous les groupes, soit alors d'en introduire une particulière de telle façon que chaque réalisation de cette notion permette d'identifier sans ambiguité tout groupe logique.

Du fait de la prépondérance des cartes perforées et de l'imprimante comme supports de fichier externe, ces notions particulières identifierons généralement des codes de cartes, lignes/colonnes correspondants à un ou plusieurs groupes logiques.

2.4.2. Article de fichier

Généralement, dans les applications d'informatique de gestion, l'utilisateur est conduit à manipuler des unités logiques associant, plusieurs groupes logiques, (certains pouvant être répétitifs). Ainsi, pour définir l'organisation logique du fichier ; c'est-à-dire une correspondance entre l'ensemble des groupes logiques et l'ensemble des positions du support, l'utilisateur doit alors déterminer ce qu'il désire considérer comme unités logiques. L'unité logique est un ensemble de groupes logiques dont la caractérisation est rattachée à une ou plusieurs notion (s) appelée (s) indicatif (s).

Remarque : Une unité logique peut posséder plusieurs indicatifs. Dans ce cas, il est nécessaire de spécifier leur hiérarchie.

Exemple : Dans un fichier d'assurés d'une compagnie d'assurances, supposons que l'on ait déterminé les groupes logiques suivants :

groupe 1 : NUMERO x NOM x ADRESSE ;
groupe 2 : NUMERO x NUMERO DE POLICE x NUMERO DE VEHICULE ;
groupe 3 : NUMERO x NUMERO DE SINISTRE x DATE x COUT-REPAR ;

Il est intéressant de remarquer que chaque groupe logique a une signification !

groupe 1 : identification d'un assuré.
groupe 2 : assurance d'un véhicule
groupe 3 : accident d'un véhicule.

Sur cet exemple, un utilisateur va pouvoir déterminer plusieurs unités logiques. Par exemple :

- l'unité logique caractérisant l'assurance d'un ou plusieurs véhicules L'indicatif serait NUMERO ou NOM du propriétaire, et l'unité logique serait composée de : groupe 1, groupe 2 ou (groupe 1, groupe 2) répété un certain nombre de fois, ou encore groupe 1, suivi d'un certain nombre de fois groupe 2.
- Si l'on considère deux indicatifs : par exemple NOM et NUMERO DE POLICE ; pour une réalisation de NOM nous pouvons avoir plusieurs réalisations de NUMERO DE POLICE.

Un article A est une notion produit des groupes logiques appartenant à une même unité logique.

Exemple : Reprenons l'exemple précédent, l'utilisateur pourra déterminer entre autres l'article assurance de véhicules par un individu :

Article 1 : groupe 1 x (groupe 2)* ($*$ signifiant un nombre quelconque de répétitions)

Ou encore l'article accident de véhicule :

Article 2 : groupe 1 x (groupe 2 x groupe 3)* .

2.4.3 Fichier

Pour préciser la définition formelle que nous avons donnée de la notion de fichier (2.2.3.), nous disons qu'un fichier est une notion produit d'ordre n d'une réunion d'articles.

Exemple : Dans l'exemple du fichier des assurés nous pourrions définir un fichier F par

$F = (\text{Article 1} \cup \text{Article 2})^{*}$ ($*$ signifie $0 \leq n$).

2.5. Aide à l'Analyse de l'Acquisition et de l'Edition de fichiers externes

Lorsque le choix du support externe et de l'organisation logique du fichier externe sont effectués et les articles définis, l'utilisateur doit pouvoir décrire de façon modulaire l'analyse des contrôles de validité à l'Acquisition, des différents groupes logiques à l'Edition.

2.5.1. Analyse des contrôles de validité à l'Acquisition

Rappelons, que les contrôles de validité envisagés seront entrepris sans consultation de fichiers permanents destinés à être mis à jour. Les contrôles de correspondance entre mouvements, et articles permanents, les mises à jour feront l'objet de traitements internes.

Comme nous le verrons au chapitre 4, le découpage des articles en enregistrements physiques, selon les types de supports.

- cartes perforées ou bandes magnétiques -

le type de supports, n'a qu'une importance relative pour l'Acquisition, si ce n'est le contrôle de séquence des enregistrements physiques que cela implique.

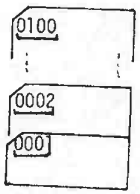
2.5.1.1. Contrôles de séquence

Nous avons déterminé trois classes de contrôles de séquence sur des fichiers supposés triés :

a - soit l'utilisateur effectue un choix de code argument intrinsèque à un support (cartes perforées, bandes magnétiques) c'est-à-dire :

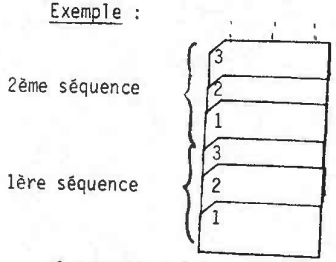
a1 - réserve une zone, de longueur fixe, de présence obligatoire au début de chaque enregistrement physique, que nous pouvons appeler argument de séquence (ou clé) d'enregistrement de telle sorte que la suite des codes réalisations de cette clé détermine une séquence soit croissante, soit décroissante.

Exemple :



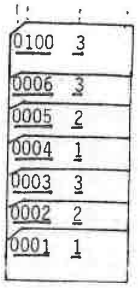
a2 - réserve une zone, de longueur fixe, de présence obligatoire au début de chaque enregistrement physique de manière à déterminer les séquences de groupes consécutifs d'enregistrements dans un fichier.

Exemple :



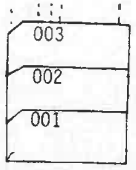
a3 - réserve une zone, toujours de longueur fixe et de présence obligatoire au début de chaque enregistrement physique, constituée de deux clés : une de type a_1, l'autre de type a_2.

Exemple :



b - Soit l'utilisateur détermine une notion de l'article, redondante de longueur fixe et présente obligatoirement dans la même zône de chaque enregistrement physique.

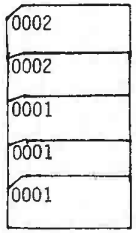
Exemple :



Remarque 1) Notons que ce choix implique deux contraintes importantes :
- chaque groupe d'entrée doit posséder cette notion argument
- chaque réalisation de groupe d'entrée doit être incluse dans l'enregistrement physique.

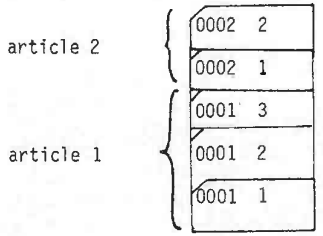
2) Dans certains cas, où la notion de l'article choisi comme clé n'est pas présente dans tous les groupes d'entrée, l'utilisateur peut imposer la présence obligatoire en début de chaque enregistrement physique.

Exemple :

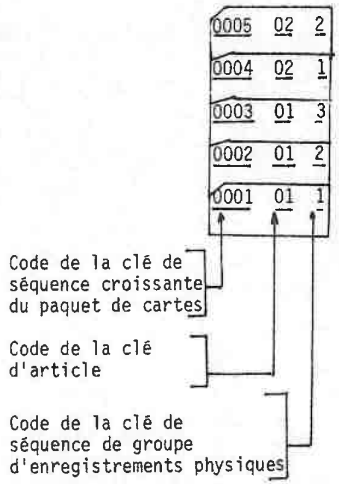


c) Soit l'utilisateur effectue le choix d'un argument mixte, composé d'une clé du type a) associée à une clé du type b).

Exemple 1 :



Exemple 2 :



Remarque : Dans le cas le plus général, où les groupes d'entrée peuvent nécessiter plusieurs enregistrements physiques, les codes de groupes d'entrée étant déterminés, nous pourrons alors ne considérer que le code clé de séquence croissante ou décroissante de la suite des enregistrements physiques.

Nous n'avons pas parlé dans le cadre de ce contrôle de séquence, du contrôle sur un indicatif intrinsèque d'articles, c'est-à-dire autre que code carte, car nous l'étudierons en 4.6.1.

L'organisation de l'Acquisition a été conçue de manière à ce que si l'on ne détecte pas par un contrôle de séquence la présence d'enregistrements successifs en doubles, tout se déroule comme si l'on exécutait les contrôles de validité d'un article ou de plusieurs articles de même indicatif dans le buffer utilisateur.

Un article sera déclaré valide quand :

* pour chaque réalisation des groupes d'entrée d'un article présenté dans le buffer on peut :

  - identifier le groupe d'entrée par le contrôle d'identification

  - vérifier que pour chaque réalisation de notion (de chaque groupe d'entrée) la classe est correcte par le contrôle de classe.
  - Vérifier que chaque réalisation de notion est vraisemblable : contrôle de vraisemblance.
  - Vérifier que chaque réalisation de notion a une réalisation compatible avec d'autres réalisations par contrôles de compatibilité individuelle.

* pour la suite des réalisations des groupes d'entrée constituant l'article on peut :

  - Vérifier que la suite des groupes d'entrée est vraisemblable contrôle de structure de l'article.
  - Vérifier que certaines réalisations de notions présentes dans différents groupes d'entrée de l'article sont compatibles : contrôle de compatibilités mutuelles sur les notions de l'article.

Vérifier la séquence des enregistrements - contrôle de séquence sur clé de code carte.

Vérifier la séquence de l'article : contrôle de séquence sur clé d'article logique.

Nous obtenons ainsi une classification générale des contrôles que l'utilisateur pourra mettre en oeuvre dans une Acquisition que nous étudierons en détail au chapitre V.

2.5.2 Analyse de l'Edition des groupes de sortie

Rappelons que notre objectif à l'Edition est d'effectuer le changement de format (interne - externe) et de support des fichiers internes CIVA.

L'analyse des état de sortie sur imprimante des applications d'informatique de gestion met en évidence différentes sortes de lignes ou groupes de lignes d'édition que nous nommerons groupes de sortie. Citons, les

en-têtes d'états, la ligne de détail, les totaux, les fins d'états, les reports de page. Il est alors important de remarquer qu'une fois déterminés, ces groupes de sortie sont imprimés selon un conditionnement unique. Par exemple, dans une facture on n'imprimera la première ligne détail qu'après avoir imprimé l'entête, le total - hors taxe qu'après la dernière ligne détail, etc...

Ainsi, l'analyse par l'utilisateur pourra s'effectuer de la manière suivante :

- Recenser les lignes de l'état qui demanderons le même traitement à l'Edition (édition et écriture) : nous dirons qu'elles sont alors de même type.

- Regrouper dans un même groupe de sortie la suite des lignes qui s'éditent sous une même condition (2.4.4.2.2.).

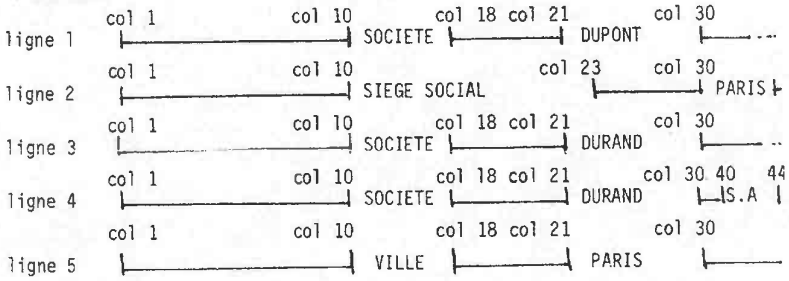
2.5.2.1. Type d'une ligne

Une ligne d'état sera caractérisée par :

- son découpage en zônes
- l'origine des informations dans les différentes zônes.

Nous dirons que deux lignes sont de même type si elles ont le même découpage en zônes et même origine des informations dans chaque zône respective. Nous parlerons alors de type d'une ligne.

Exemple :



Alors : les lignes 1 et 2 sont de types différents car elles n'ont pas le même découpage e: zônes.

- Les lignes 1 et 3 sont de même types puisque possèdant le même découpage en zônes et les informations de même origine.
- Remarquons que les lignes 1 et 4 ne sont pas de même type.
- Les lignes 1 et 5 possèdent le même découpage en zône mais avec des informations d'origine différentes.

2.5.2.2. Groupe de sortie

Un groupe de sortie est un ensemble ordonné et indissociable de types de lignes éditées sous une même condition.

Nous distinguons les conditions suivantes :

- celles dûes aux contraintes technologiques :
par exemple, le nombre de lignes maximum que l'on peut imrpimer sur une page,

- celles inhérentes à l'application traitée :
par exemple, changement de valeur d'un indicatif, fin de fichier.

Remarque : Pour exprimer les contraintes technologiques nous utiliserons des variables qui seront soit des compteurs, soit des sauts de lignes ou de pages.

Nous étudierons plus en détail au cours du chapitre IV la logique générale de l'Edition, disons seulement que l'utilisateur doit exprimer sur l'état de sortie un ensemble de relations simples.

Si nous appelons GSI et GSJ des groupes de sortie d'un état à éditer, et COND un conditionnement alors les relations à exprimer seront de la forme

(GSI et COND) entrainent (GSI) ou
(GSI) entraine (GSJ)

Nous interprétons ces relations en disant qu'après l'édition du groupe I si nous sommes dans les conditions COND alors nous devons de manière unique éditer le groupe de sortie J. De plus, pour chaque type de ligne de chaque groupe de sortie, il sera nécessaire de définir des sauts de lignes ou de pages avant l'écriture ; Enfin, il faudra déterminer quels cumuls éditer, lors de ruptures de valeurs d'indicatifs internes ou lors d'un

dépassement de capacité de page, dans un report en fin et / ou en début de page.

2.6. Conclusion

Les techniques d'analyse des fichiers externes que nous venons de présenter doivent permettre à l'utilisateur de définir de manière simple et très précise les données externes.

Dans les chapitres suivants nous allons définir des langages de description pour l'Acquisition et l'Edition qui décrivent les résultats de cette analyse.

CHAPITRE 3

Les Informations CIVA.

Langage de description des fichiers externes à l'Acquisition

Introduction

L'étude présentée au chapitre 2 a montré l'intérêt de dissocier les informations externes, des informations internes. En effet, il est alors possible de déterminer une organisation logique et une codification des informations internes de manière à rendre les traitements le plus efficace possible, quasiment indépendamment de l'organisation logique externe.

3.1. Informations internes en CIVA

Nous avons distingué quatre sortes d'informations internes [1] :

- l'information élémentaire, plus petite unité d'information

Exemple : CODE-PRODUIT entier

- l'information structurée composée d'informations élémentaires ou structurées

Exemple : le numéro de Sécurité Sociale.

NUM-SS structure (sexe entier, année-mois entier, mois-nais entier, cod-départ entier, numero-com entier, num-registre entier)

- l'information file.

3.2. Types des informations

Le type d'une information est un ensemble de valeur (ou données). Nous dirons alors qu'une information appelée A est d'un type B si l'ensemble des données de A appartient à l'ensemble des données de B. Il est fréquent qu'un même type d'information soit présent plusieurs fois dans une même description de fichiers internes. Il sera alors commode de ne donner qu'une fois la description du type dans une classe, en donnant la possibilité de compléter cette définition par renvoi d'une classe à l'autre. A la compilation, toutefois, toute définition devra être complète. De plus, il sera nécessaire de préciser le niveau de définition du type : classe de l'Application, ou système [1].

3.3. Description explicite des types [1]

3.3.1. Informations élémentaires:

- Définitions standard : nous utilisons les types courants : caractère, entier, booléen, réel, décimal, (forme interne), complexe, réel double précision, date.

Un identificateur A possédera un de ces types si on déclare par exemple A entier.

- Définition par restriction : nous associons au nom du type une condition restrictive sur l'ensemble des données. Par exemple, A type entier < 5000.

- Définition explicite par l'ensemble des valeurs

Par exemple SITUATION-FAM code (marié, célibataire) ; situation sera une suite de caractères pouvant prendre les valeurs marié ou célibataire.

3.3.2. Files

Nous distinguerons trois types de files : les files de taille fixe, de taille bornée, et de taille variable.

Exemples

```
TAB1 file (15) car ;
TAB2 file (max = 15) car ;
TAB3 file car ;
```

3.3.3. Structure :

c'est une composition d'objets de type simple, file ou structure.

Exemple :

```
A structure (B entier, C réel, D file (5) car)
T structure (X entier, Y structure (V réel, W réel))
```

Remarque 1) Il sera possible de définir des analogies par "comme", en déclarant un nouvel identificateur du même type qu'un autre.

2) D'autres exemples sont donnés dans [13] avec des spécifications plus détaillées ainsi que les règles de syntaxe.

3.4 Description logique d'un fichier, répartition des objets internes sur le support [3] [1]

3.4.1. La description logique d'un fichier interne est une description de structure, donnée dans une classe.

Exemple :

```
Classe A ;
  FICHINT file ARTICLE,
finclasse ;
Classe B ; utilise classe A ;
  ARTICLE structure (nom file (10) car, date-naissance date) ;
finclasse ;
```

3.4.2. La répartition des objets sur le support, selon un découpage physique et une répartition en bloc est une contrainte que l'utilisateur doit ignorer : c'est le système qui se préoccupe de la gestion des buffers des fichiers internes.

Il n'y a pas de limite théorique à la taille d'un enregistrement logique. Pratiquement, un enregistrement logique nécessite un buffer de sa taille.

3.5. Informations externes en CIVA

Si l'on se refère aux définitions (A3) et (A1) de IFIP-ICC Vocabulary [48], une information peut être interprétée comme la représentation utilisant certaines conventions connues d'une notion ; une donnée étant la réprésentation codifée d'une réalisation de notion. Nous parlerons donc dans la suite de données externes, d'informations externes. Le type d'une information externe sera interprêté comme un ensemble de données externes ce que nous appellerons ainsi type externe d'une information.

Les informations externes seront représentées par les valeurs des variables externes. Une variable externe étant définie par une description

externe qui lui associe un identificateur externe et un type externe.

Dans les descriptions de fichiers externes, nous imposerons que tous les identificateurs externes soient distincts. Cette contrainte est introduite pour supprimer les ambiguités possibles dans les transferts à l'Acquisition.

3.6. Langage de Description des fichiers externes CIVA à l'acquisition.

Le langage que nous allons présenter doit permettre de décrire facilement les résultats de l'analyse, que nous avons donné au chapitre 2, en groupes d'entrée et en définissant des types externes d'informations auxquels nous assignons une signification.

3.6.1. Structure générale d'une description de fichier externe en entrée :

La description d'un fichier externe en entrée commencera par % FICHIER suivi d'un identificateur externe dénotant le nom du fichier en entrée. Cette description est composée d'une section % DGE obligatoire et de trois autres sections % DCI, % DCM, % DSY facultatives.

Syntaxe :

⟨description fichier externe en entrée⟩ ::= %FICHIER ⟨identificateur fichier ext⟩ , ⟨suite section entrée⟩ ;

(1) ⟨suite section entrée⟩ ::= %DGE ⟨suite ge⟩ | %DGE ⟨suite ge⟩ ; %DSY ⟨suite delsym⟩ | %DGE ⟨suite ge⟩ ; ⟨suite sectcontrsyn⟩

(2) ⟨suite sectcontrsyn⟩ ::= %DCI ⟨suite contrind⟩ | %DCI ⟨suite contrind⟩ ; ⟨suite sect contr⟩

(3) ⟨suite sect contr⟩ ::= %DSY ⟨suite del syn⟩ | % DCM ⟨suite contrmut⟩ | % DCM ⟨suite contrmut⟩ ; %DSY ⟨suite decl syn⟩

Sémantique :

(1) la section %DGE permet de décrire les groupes d'entrée du fichier externe à l'Acquisition. Tous les contrôles d'identification de groupe d'entrée de séquence sur code carte (rep. indicatif) ainsi que les

contrôles individuels de classe, de vraisemblance , de compatibilité sont déclarés dans cette section.

(2) Dans la section %DCI, nous pouvons définir simplement des déclarations de contrôles individuels dans %DGE. Si l'on utilise un appel de méta fonction , celle-ci sera alors définie dans % DCI.

(3) La section %DCM est une section qui permet de définir les controles mutuels sur les variables externes des différents groupes d'entrée du fichier externe.

(4) La section % DSY permet de définir, pour chaque article de fichier interne utilisé à l'Acquisition et ayant des identificateurs de feuille de même nom, de définir les correspondances entre les identificateurs externes tous distincts et de tel identificateurs internes.

3.6.1.1. Remarques

- Les articles logiques externes composés des groupes d'entrée peuvent être de formats variables, certains groupes d'entrée sont répétitifs ou absents. Toutefois un groupe d'entrée aura un format fixe.

- Dans le cas général un article logique pourra nécessiter plusieurs enregistrements physiques (souvent cartes).

3.6.2 Identificateurs externes :

Syntaxe

(1) ⟨Identificateur fichier externe⟩ ::= ⟨identificateur externe⟩

(2) ⟨Identificateur externe⟩ ::= ⟨lettre⟩ | ⟨lettre⟩ (⟨letchif⟩)* ⟨lettre⟩ | ⟨lettre⟩ (⟨letchif⟩)* ⟨chiffre⟩

⟨let chif⟩ ::= - ⟨lettre⟩ | ⟨chiffre⟩ | ⟨lettre⟩ ⟨let chif⟩ | - ⟨let chif⟩ | ⟨chiffre⟩ ⟨let chif⟩

Remarque : Rappelons que la notation '*' utilisée dans (2) signifie

$$(\langle \text{let chif} \rangle)^* = \bigcup_{n \geq 0} (\langle \text{let chif} \rangle)^n \text{ avec,}$$

$(\langle \text{let chif} \rangle)^0$ représentant la suite vide.

Exemples : A, CIVA, CODE 1, PRIX-UNITAIRE .

Sémantique :

Un identificateur de fichier externe (1) est un identificateur externe qui permet de faire la liaison avec le système (par un D.C.B.).

Pour les identificateurs seuls les huit premiers caractères seront significatifs.

Dans une description de fichier externe tous les identificateurs externes utilisés doivent être différents.

3.6.3. Données externes

- Syntaxe

⟨Donnée externe⟩ ::= ⟨donnée num⟩ / ⟨donnée alphab⟩ / ⟨donnée alphan⟩

⟨donnée num⟩ ::= ⟨occur⟩ / ⟨signe⟩ ⟨occur⟩ / ⟨occur⟩ . ⟨occur⟩
⟨signe⟩ ⟨occur⟩ . ⟨occur⟩

⟨occur⟩ ::= 0 / 1 / 2 / ····· 9 / ⟨occur⟩ (⟨occur⟩)

⟨signe⟩ ::= + / -

⟨donnée alphab⟩ ::= ⟨suite lettre b⟩

⟨suite lettre b⟩ ::= A / B / ····· / Z / ␣ / ⟨suite lettre b⟩ (⟨suite lettre b⟩)

⟨donnée alphan⟩ ::= ⟨occur⟩ / ⟨suite lettre b⟩ / ⟨scarebcdic⟩
⟨donnée alphan⟩ (⟨donnée alphan⟩)

⟨scarebcdic⟩ ::= caractères EBCDIC de 10070 / ⟨scarebcdic⟩ (⟨scarebcdic⟩

Exemples : 10
10.52
DATE,ENFANT
PU1, NUMERO2

Sémantique :

Les données externes utilisées sont les nombres entiers, les nombres décimaux, les chaines de caractères ne comportant que des lettres ou

les chaînes de caractères composées de caractères EBCDIC.

3.6.4. Section % DGE

Cette section permet de définir les groupes d'entrée des fichiers externes résultants de l'analyse étudiée au chapitre 2. Chaque définition de groupe d'entrée pourra comporter des déclarations de contrôles : contrôles de séquence sur code carte, sur indicatif d'article logique externe, contrôles de classe, de vraisemblance, de compatibilité individuelle.

Syntaxe de % DGE

⟨suite de groupe d'entrée⟩ ::= ⟨identificateur externe⟩ :
⟨descriptge⟩ / ⟨suite de groupe d'entrée⟩ (, ⟨suite de groupe d'entrée⟩)*

⟨descript ge⟩ ::= ⟨descript simple⟩ / ⟨ident ge⟩ /
⟨descript simple⟩ / ⟨delseq⟩ ⟨ident ge⟩ ⟨descript simple⟩

⟨ident ge⟩ ::= ⟨identificateur externe⟩ ⟨formext⟩ IGE
⟨code externe⟩

⟨code externe⟩ ::= ⟨donnée num⟩ / ' ⟨donnée alphab⟩ ' /
' ⟨donnée alphan⟩ '

⟨del seq⟩ ::= CLT ⟨formext⟩ / CLA ⟨formext⟩ / CLG ⟨formext⟩ /
CLT ⟨formext⟩ CLA ⟨formext⟩ / CLT ⟨formext⟩ CLG ⟨formext⟩ /
CLA ⟨formext⟩ CLG ⟨formext⟩ /
CLT ⟨formext⟩ CLA ⟨formext⟩ CLG ⟨formext⟩

⟨descript simple⟩ ::= ⟨identificateur externe⟩ ⟨form ext⟩

⟨identificateur externe⟩ ⟨form ext⟩ , CLE ⟨identificateur externe⟩
⟨form ext⟩ ; ⟨contrind⟩ / ⟨descript simple⟩ (⟨descript simple⟩)*

⟨contr ind⟩ ::= ⟨contr class⟩ / ⟨contr vrais⟩ / ⟨contr cump ind⟩ /
$ ⟨méta dclind⟩ / ⟨contr class⟩ , ⟨contr vrais⟩ / ⟨contr class⟩ ,
⟨contr compind⟩

⟨contr vrais⟩ , ⟨contr compind⟩
⟨contr class⟩ , ⟨contr vrais⟩ , ⟨contr compind⟩

⟨contr class⟩ ::= C, [⟨cond class⟩]

⟨contr vrais⟩ ::= V, ⟨cond vrais⟩

⟨cond vrais⟩ ::= ⟨cond vrais 1⟩ | ⟨cond vrais 2⟩

⟨cond vrais 1⟩ ::= [⟨cond gen⟩]

⟨cond gen⟩ ::= ⟨opérateur⟩ ⟨donnée num⟩ | ⟨opérateur⟩ ⟨liste gen⟩ /
⟨cond gen⟩ OU ⟨cond gen⟩ / ⟨cond gen⟩ ET ⟨cond gen⟩

⟨liste gen⟩ ::= (⟨list élém⟩) / ⟨list gen⟩ (, ⟨liste gen⟩)*

⟨list élém⟩ ::= ⟨list élém exp⟩ / ⟨list élém imp⟩

⟨list élém exp⟩ ::= ⟨donnée externe⟩ / ⟨donnée externe⟩
(, ⟨donnée externe⟩)*

⟨list élem imp⟩ ::= ⟨born inf⟩ , ⟨born sup⟩ / ⟨born inf⟩ ,
⟨born sup⟩ , ⟨pas⟩

⟨born inf⟩ ::= ⟨donnée num⟩

⟨born sup⟩ ::= ⟨donnée num⟩

⟨pas⟩ ::= ⟨donnée num⟩

⟨cond vrais 2⟩ ::= module ⟨identificateur module⟩ ⟨paramètres⟩ /
$ ⟨méta dclind⟩

⟨contr compind⟩ ::= P, ⟨cond compind p⟩

⟨cond comp ind p⟩ ::= [⟨cond compind⟩]

⟨cond compind⟩ ::= ⟨identificateur externe⟩ ⟨opérateur⟩
⟨donnée externe⟩

⟨identificateur externe⟩ = ⟨liste élém⟩ /
⟨cond comp ind⟩ ET ⟨cond comp ind⟩ /
⟨cond comp ind⟩ OU ⟨cond comp ind⟩

⟨méta dclind⟩ ::= (méta variable CIVA)

⟨cond class⟩ ::= NUMERIC/ ALPHAB / ALPHAN

⟨form ext⟩ ::= ⟨format ext num⟩ / ⟨format ext alphab⟩ /
⟨format ext alphab⟩ / ⟨date⟩

⟨format ext num⟩ ::= C / C (⟨format ext num⟩) / C (⟨occur⟩)

⟨format ext alphab⟩ ::= A / A (⟨format ext alphab⟩) / A (⟨occur⟩)

⟨format ext alphan⟩ ::= X / X (⟨format ext alphan⟩) / X (⟨occur⟩)

⟨date⟩ ::= D1 / D2 / D3

⟨opérateur⟩ ::= = / ≠ / > / >= / < / <=

Exemple :

```
% DGE
    E1 : CLT C (2)
         CLA C (2)
         CLG C (3)
         GE1 X (4) IGE 'GA10'
         NOM X (20), 'C', [ ALPHAB ]
         PRENOM X (10)
         SEXE C (2), 'V', [ = (1,2)]
         SIT-FAM, X (11), 'V', [=(MARIE,VEUF,CELIBATAIRE,DIVORCE)]
         AGE , C (3), 'V', [ ≤ 100 ]
    E2 : GE2 X (4) IGE 'GA11'
         NOM-JF, X (20), 'P', [ SEXE=2 ET SIT-FAM = 'MARIE' ] ;
```

Sémantique :

La section % DGE permet de définir les groupes d'entrée et pour chaque groupe d'entrée les contrôles possibles lors de l'Acquisition (chapitre 4). Notons que nous avons introduit trois types externes date D1, D2, D3 de formes suivantes :

D1 : 1972 05 21
D2 : 21.05.72
D3 : 21 MAI 72

3.6.5. Section % DCI

Cette section permet de définir des méta-fonctions de contrôles individuels lorsque les définitions sont trop 'lourdes' pour figurer dans la section % DGE ou encore lorsqu'un type de contrôle est commun a plusieurs variables externes.

Nous pourrions aussi, dans cette section déclarer les définitions de méta-fonctions donnant le texte de modules de calcul d'un argument dans certains contrôles de vraisemblance.

3.6.6. Section % DCM

Syntaxe

⟨suite contr mut⟩ ::= ⟨identificateur externe⟩ (⟨cond mut ⟩)
⟨suite contr mut⟩ (suite contr mut)

⟨cond mut ⟩ ::= ⟨expression arthmétique simple⟩ ⟨opérateur⟩
⟨donnée externe⟩ / ⟨donnée externe⟩ ⟨opérateur⟩
⟨expression arithmétique⟩

⟨expression arithmétique simple⟩ ::= ⟨identificateur externe ⟩
⟨opérateur binaire⟩ ⟨identificateur externe⟩ / ⟨donnée externe⟩
⟨opérateur binaire⟩ ⟨identificateur externe⟩
⟨identificateur externe⟩ ⟨opérateur binaire⟩ ⟨donnée externe⟩

⟨opérateur binaire⟩ ::= + / - / * / / / **

Exemple :

PREST-FAM (ALLOC 1 + ALLOC 2 > 1000)

Sémantique :

Dans les expressions arithmétiques nous utiliserons uniquement des variables externes définies dans % DGE.
Les expressions arithmétiques seront simples.

3.6.7. Section % DSY

Syntaxe :

⟨suite decl syn⟩ ::= ⟨identificateur externe⟩ SYN
⟨suite identificateur interne⟩ / ⟨suite decl syn⟩
(⟨suite decl syn ⟩)

⟨suite identificateur interne ⟩ ::= ⟨identificateur interne ⟩
⟨qualif ⟩ ⟨suite identificateur interne ⟩
(, suite identificateur interne)

⟨qualif ⟩ ::= de ⟨identificateur interne⟩ | ⟨qualif⟩ (⟨qualif⟩).

Exemples :

DATE 1 SYN DATE de COMMANDE ;
DATE 2 SYN DATE de EXPEDITION ;

Remarques : Certaines notions internes n'ont pas été spécifiées dans cette grammaire ; pour plus de précisions se reporter à [1] [6] [4] .

3.7. Conclusion

Le langage de description des fichiers externes à l'Acquisition permet de définir simplement d'une part les contrôles physiques (contrôles de séquence essentiellement) d'autre part les contrôles logiques résultant d'une analyse fixe des notions contenues dans le fichier externe.

CHAPITRE 4.

SERVICES D'ACQUISITION DE FICHIERS EXTERNES

4.1. Introduction.

Nous avons présenté au chapitre 1 (1.3.), au chapitre 2 (2.5.) les différents objectifs à atteindre à l'élaboration de Services d' Acquisition de fichiers externes.

Rappelons que, selon l'importance attachée à la fiabilité des informations externes, du service attendu pour l'édition des informations erronées (et du coût de traitement en ordinateur), il est possible de définir une multitude de services d'Acquisition de fichiers. Le nombre beaucoup trop important de conventions possibles rend illusoire la définition d'une instruction unique comme nous allons l'illustrer ci-après.

Notons encore que pour les instructions d'affectation [6], nous avons rencontré les mêmes difficultés.

4.1.1. Exemples d'Acquisition de fichiers externes.

Dans chacun des exemples simples que nous allons exposer, nous voulons surtout mettre en évidence les différentes conventions qu'il est possible d'utiliser.

...

4.1.1.1. Changement de support et de format.

Notation.

- le fichier externe, noté Y est constitué d'un seul article composé des groupes d'entrée GE1 , GE2, dont les réalisations sont obligatoirement présentes et non répétitives. L'article est supposé inclus dans un enregistrement physique -carte perforée par exemple.

- le fichier interne, noté X , ne sera constitué que d'un seul article noté Z de type structure, et ne comportant pas de champ de type file de structure.

Description des fichiers X et Y :

FICHIER Y

DGE

E 1 : E11 X(10)
E12 C(5). C(2)
E13 A(3),
E 2 : E21 X(2)
E22 X(6) ;

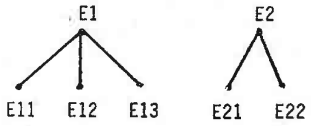
FICHIER X

Z structure (E11 file (10) car, C1 structure (E 21 file (2) car, E22 file (6) car),
C2 structure (E13 décimal 9 (5) v 9 (2),
E12 file (6) car)) ;

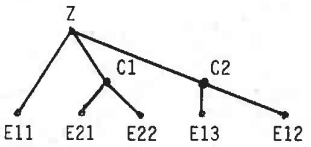
...

Les arborescences associées aux descriptions de fichiers sont :

- Fichier Y



- Fichier Z



Les fichiers X, Y ainsi décrits, envisageons l'Acquisition la plus élémentaire qu'il soit. Nous noterons E ij une feuille "courante" traitée.

Pour chaque Z de X faire

- Lirele fichier externe Y (si fin de fichier : faire le traitement fin de fichier) ;

Pour chaque E ij de Z faire

. recherche de E ij de Y correspondant à E ij de Z ;

. si le type de E ij de Z est compatible avec le type de E ij de Y alors effectuer la conversion de format externe en format interne ;

...

ranger le résultat dans Z fsi ;

- fpc ;
- écrire l'article Z ;

fpc ;

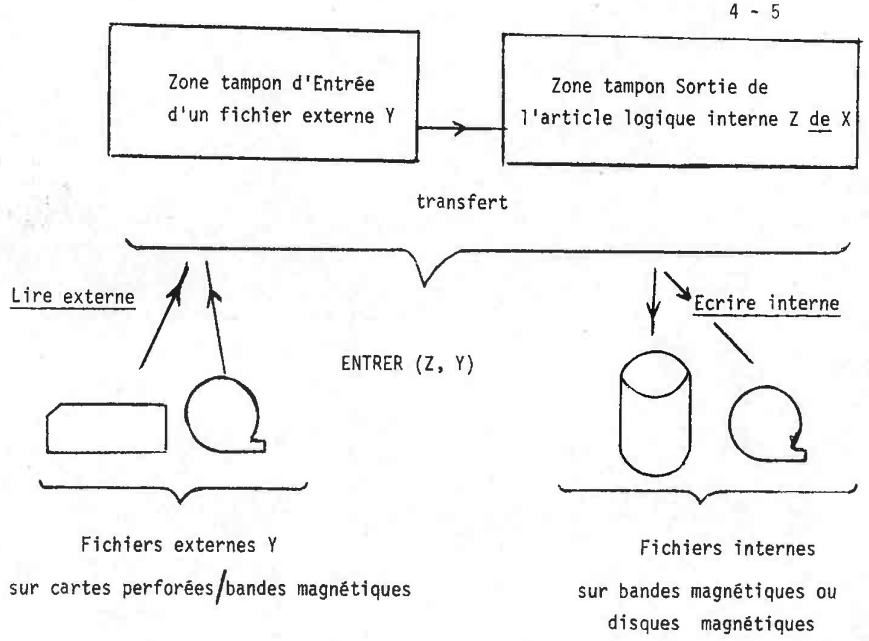
Les fonctions mises en évidence dans cette description de traitement sont :

1. La lecture et la fin du fichier externe Y ;
2. L'analyse des arborescences associées aux groupes d'entrées pour la recherche de la feuille externe de même nom que la feuille interne traitée.
3. L'analyse des compatibilités de types externes et de types internes.
4. La conversion d'une donnée de format externe en format interne.
5. Le rangement du résultat -donnée en format interne- dans la zone tampon affectée au fichier interne X.
6. L'écriture de l'article interne Z situé dans la zone tampon de X.

Remarques.

1. Ultérieurement, lorsque nous parlerons d'"entrer" un article Z de X de format externe Y, ce que nous noterons ENTRER (Z, Y), il s'agira de la création en format interne d'un article logique Z de format externe décrit par Y.

...



2. Si l'on envisage en outre, des traitements de contrôles de séquence de cartes, soit sur l'ensemble du fichier externe, soit sur des séquences de cartes d'article, nous n'aurons qu'à insérer une fonction spécifique de contrôle de séquence entre (1) et (2) sur un argument code de carte.

3. L'analyse des informations externes étant indépendante de l'organisation logique de ces informations sur un support (chapitre 2), il est important de dire que pour une organisation logique externe et une organisation logique interne l'Acquisition établira automatiquement l'interface. Dans l'exemple 4.1.1.1., notamment, nous pouvons noter que l'ordre de présentation des "rubriques" (ou feuilles) E12 et E 13 dans le fichier interne n'est pas le même que sur le fichier externe dans le groupe d'entrée E 1.

...

4.1.1.2. Acquisition utilisant un contrôle de séquence ; avec contrôle de structure, de vraisemblance et de compatibilité individuelle.

Considérons les descriptions de fichier X et Y suivantes :

```
% FICHIER Y

  % DGE

    E 1 : CLT   C(5)
          GE1   C(1) IGE [1]
          E11   X(10)
          E12   C(5) . C(2)
          E13   A(3),

    E 2 : GE2   C(1)  IGE[2]
          E21   X(2)
          E22   X(6) ;
```

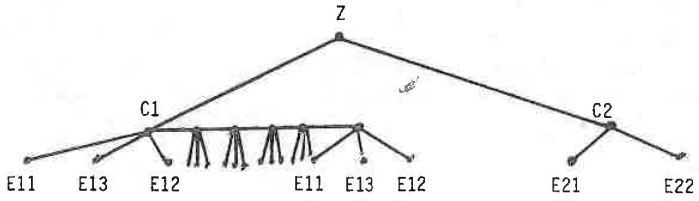
```
FICHIER X

Z structure (C1 file (6) structure (E11 file (10) car,
E13 file (3) car, E12 décimal 9(5) V 9(2)),
C2 structure (E21 file (2) car, E 22 file (6) car)) ;
```

...

Les arborescences associées à ces descriptions de fichiers sont données ci-dessous :

Fichier Y

Fichier X



Il est alors nécessaire de donner la structure de présentation des groupes d'entrée dans l'article logique externe à l'Acquisition.

Considérons alors une acquisition qui effectue l'"entrer" de l'article interne Z selon un format Y avec le contrôle de séquence de cartes de fichier externe, les contrôles de structure de l'article logique, de vraisemblance et de compatibilité individuelle.

...

Pour chaque Z de X faire :

Lire le fichier externe Y (si fin du fichier, traitement fin fichier Y) ;

Contrôle de séquence de l'article logique sur CLT ;
Contrôle de structure de l'article logique - sur IGE -
s'il n'y a pas d'erreur de séquence détectée ;

Pour chaque élément de rang I de la file C1 faire:

- recherche de E1i de Y correspondant à C1 (I,i) de Z
- si contrôle de vraisemblance sur E1i de E1 alors traiter le contrôle de vraisemblance ; sinon si contrôle de compatibilité et pas d'erreur de vraisemblance alors traiter le contrôle de compatibilité fsi ; fsi ;
- si le type de E1i de Y est compatible avec le type de C1 (I,i) alors effectuer la conversion du format externe en interne ; ranger le résultat dans Z fsi ;

fpc ;

Pour chaque E 2j de C2 faire

- recherche de E 2j de Y correspondant à E2j de C2;
- si contrôle de vraisemblance sur E 2j de Y alors traiter le contrôle de vraisemblance fsi ;

  si le contrôle de compatibilité et pas d'erreur de vraisemblance alors traiter le contrôle de compatibilité fsi ;
- si le type de E 2j de Y est compatible avec le type de E 2j de Z alors effectuer la conversion, ranger le résultat dans Z fsi ;

fpc ;

- écrire l'article Z ;

fpc ;

...

Par rapport aux fonctions énumérées dans l'exemple (4.1.1.1.), nous remarquons des fonctions identiques. C'est le cas des fonctions de lecture de fichier externe Y, d'écriture d'article logique interne Z. Ces fonctions seront des instructions de base de CIVA.

La fonction d'analyse des arborescences pour la correspondance des noms des feuilles que l'on retrouve dans cet exemple, fait intervenir trop de cas particuliers pour que l'on en fasse une instruction unique de CIVA.

Quant aux fonctions de contrôles de séquences, de structure, de vraisemblance, de compatibilité individuelle, de compatibilité mutuelle, elles seront étudiées plus en détail ultérieurement. Nous dirons simplement qu'aucune d'elles, du fait du nombre de possibilités dans le traitement des erreurs, ne peut faire l'objet d'une instruction de base de CIVA.

4.2. Instructions d'Entrée-Sortie de CIVA à l'Acquisition de fichiers externes.

(pour les instructions de lecture de fichier interne et d'écriture de fichier externe : se reporter à 5.2.)

Il existe trois opérations d'entrée-sortie à l'Acquisition :

- la lecture de fichiers externes,
- l'écriture d'un article de fichier interne,
- l'édition simple d'une variable, d'une liste de variables ou de constantes en format externe (resp. interne).

4.2.1. Lecture de fichiers externes, fin de fichier.

Syntaxe.

⟨instruction de lecture de fichier externe⟩ = LIREXT à ⟨identificateur de fichier externe⟩;

...

Sémantique.

La lecture de fichier externe assure l'accès aux articles logiques sur le support externe et les enregistre dans une zone tampon (ou buffer) en mémoire.

Nous réalisons la lecture LIREXT en utilisant un appel moniteur

CAL 1, 1 adresse d'une table FPT

La table FPT (Function Parameter Table) sera de même forme que celle créee par M :READ sous le système SIRIS 7 / SIRIS 8 sur 10 070 CII en mode DEV [10].

La gestion automatique de la zone tampon (buffer) sera assurée par des procédures écrites directement en assembleur. Ainsi à chaque demande de lecture LIREXT, un article logique externe doit être présent au début de la zone tampon.

Les ordes d'ouverture et de fermeture de fichiers seront assurés par le système.

Exemple.

LIREXT àY ;

Lorsqu'une commande de fin de fichier est rencontrée à la lecture, un code (05) est rangé dans l'octet 0 du registre SR3.

Nous introduisons une fonction de test Boolen de fin de fichier appelée FFICH qui prendra les valeurs 0 ou 1.

⟨fin de fichier⟩ :: = FFICH (à ⟨identificateur de fichier externe⟩) ; |
FFICH (⟨identificateur de fichier interne⟩) ;

Exemple.

LIREXT à Y ; si FFICH (àY) alors traitement de fin de fichier de Y ;

4.2.2. Ecriture d'article de fichier interne.

Syntaxe

⟨instruction d'écriture de fichier interne⟩ :: = ECRIRE (1) ⟨identificateur de l'article⟩ ;

...

Sémantique :

L'instruction (1) provoque l'écriture d'un enregistrement à partir d'une zone tampon en mémoire. Le type de l'unité de sortie ne pourra être que bande ou disque magnétique.

L'écriture sera réalisée en utilisant un appel moniteur :
CAL1, 1 adresse d'une table FPT.

La table FPT aura la même forme que celle créée par la procédure M : WRITE du système SIRIS 7 sur 10 070 en mode FIL.

4.3.2. Editions simples d'une variable, d'une liste de variable, de constantes en format externe (resp. interne).

Lors des traitements de données externes erronées, il importe de pouvoir lister simplement sur imprimante une variable, une liste de variables ou de constantes (libellés d'erreurs). Ce problème n'est pas nouveau ; il est d'ailleurs résolu dans la plupart des langages de programmation (instruction INPUT de Fortran IV étendu, DISPLAY de COBOL, ...)

A la mise au point d'une application [4] de telles instructions peuvent être utiles pour éditer suivant un format défini à l'implémentation une variable, une liste de variables ou une constante, en types internes.

Syntaxe :

⟨édition simple⟩ :: = IMPRIMER1 (1) ⟨liste de sortie externe⟩ ; /
IMPRIMER2 (2) ⟨liste de sortie interne⟩ ;

...

⟨liste de sortie externe⟩ :: = â ⟨constante externe⟩ | â ⟨variable externe⟩ |
⟨liste de sortie externe⟩ (, ⟨liste de sortie externe⟩)*

⟨liste de sortie externe⟩ :: = ⟨constante interne⟩ | ⟨variable interne⟩ |
⟨liste de sortie interne⟩ (, ⟨liste de sortie interne⟩)*

Remarque : la notation (⟨A⟩)* représente une suite finie d'éléments de A éventuellement vide.

Sémantique :

L'instruction (1) IMPRIMER1 imprime au début de ligne de l'imprimante la liste de sortie externe, soit avec le format spécifié dans la description de fichier externe pour les variables externes, soit sous forme d'une chaîne de caractères EBCDIC pour les constantes externes.

Exemple :

Reprenons la description externe de fichier de 4.1.1.2.

IMPRIMER1 (âE1, âE21, 'libellé d'erreur') ;

la variable âE1 désigne la valeur externe suivante :

' _ _ 11 NOM _ _ _ _ _ _ _ _ _ _ 1411 ABC'

la variable âE21 désigne : 'P1'

L'instruction aura pour effet d'imprimer en début de ligne :

_ _ 11 NOM _ _ _ _ _ _ _ _ _ _ 1411 ABC P1 LIBELLE _ D _ ERREUR

L'instruction (2) IMPRIMER2 imprime en début de ligne d'imprimante la liste de sortie interne suivant un format externe défini lors de l'implémentation

booleen : VRAI ou FAUX

caractère : un caractère ou une étoile si le code du caractère n'appartient pas au code EBCDIC.

...

entier : valeur cadrée à droite sur dix caractères.

décimal : sa valeur pour lequel le point décimal virtuel sera imprimé.

réel ou double précision : sa représentation suivant le format de type E ou F de Fortran IV.

complexe : il sera représenté par deux réels séparées par ','

file de caractère : la chaîne de caractère (sans séparation entre les caractères.)

File d'éléments simples : chaque élément de la file sera édité suivant le format défini ci-dessus et ses éléments seront séparés par des espaces.

structure : chaque champ sera édité selon son type.

Exemple :

B file (5) car ;

A file (3) entier ;

C réel ;

Si l'on considère : B = 'ABCDF' ;

A (1) = 11 ;

A (2) = 14 ;

A (3) = 8 ;

C = 1/3 ;

Alors :

IMPRIMER2 (A, B, C, 'A,B,C') provoquera l'impression de la ligne suivante :

11 14 8 ABCDF 0. 333333 A,B,C

...

4.3. Instruction d'affectation d'objets externes vers des objets internes de CIVA.

Pour transférer des données externes vers des objets internes lors de l'Acquisition de fichier externe, il est nécessaire de pouvoir :

- d'une part, désigner des emplacements de mémoires externes et des mémoires internes ; l'identificateur externe utilisé à cet effet sera distingué par le symbole à de l'identificateur interne puisqu'en général ils portent le même nom,

- d'autre part, accèder aux données externes ou internes.

C'est le rôle de l'affectation que nous voulons définir et que nous qualifierons par la suite d'affectation externe - interne. Bien que cette affectation présente beaucoup de similitude avec l'affectation d'objets internes [6], les conventions utilisées n'étant pas les mêmes, nous sommes conduits à la définition d'une instruction particulière de CIVA qui ne recouvrira d'ailleurs que quelques cas simples.

Syntaxe.

⟨Affectation externe-interne⟩ :: =⟨identificateur interne⟩: =
à ⟨donnée externe⟩ / à ⟨identificateur externe⟩

Exemple :

I : = à 14 ; A : = à E21 ;

Sémantique :

Les identificateurs externes (resp. internes) désignent des variables de type externe (resp. de type interne).

...

Comme dans [6], le nombre de cas d'affectations externe-interne est beaucoup trop important pour définir une seule instruction recouvrant toutes les possibilités. Nous avons ici aussi préféré n'envisager que les possibilités courantes pour la définition de l'instruction d'affectation externe-interne. L'utilisateur pourra toujours définir des instructions de transfert plus complexes dans le méta-langage de CIVA en utilisant l'instruction de base.

Nous allons donc énumérer ci-dessous les différentes possibilités d'affectations simples d'objets externes vers des objets internes CIVA ; en donnant les conventions utilisées dans chaque cas.

4.3.1. Affectations élémentaires.

La variable interne est de type simple ou de type file d'éléments simples. Rappelons qu'un type simple dénote soit une constante, soit une variable simple ou indicée [6][1].

La variable externe est de type élémentaire.

4.3.1.1. La variable interne est de type simple, la variable externe élémentaire.

La valeur de la variable externe est alors convertie dans le type interne et ensuite affectée à la variable interne.

4.3.1.2. La variable interne est de type file, la variable externe est élémentaire répétitive.

Quel que soit le type de file (maximum, fixe ou variable) nous prendrons la convention d'affecter après conversion successivement aux éléments de file de la variable interne les valeurs de la variable externe dans l'ordre de leur emplacement en mémoire. Nous arrêtons le transfert des valeurs externes converties dès que la file de la variable interne sera "remplie" ou lorsque toutes les valeurs externes auront été "épuisées".

...

Si des valeurs externes n'ont pas été affectées, elles seront perdues. Après l'affectation, les tailles actuelles des files variables ou bornées réceptrices sont égales à la taille de la suite des éléments transférés dans la file.

4.3.2. Affectations non élémentaires.

L'une au moins des variables est de type structure : dans ce cas, toutes les feuilles des structures associées aux variables internes sont simples, la variable externe est non répétitive.

4.3.2.1. La variable externe est seule structurée.

4.3.2.1.1. Variable interne de type simple.

L'affectation opérera le transfert après conversion de la première feuille de la structure externe vers l'élément interne.

4.3.2.1.2. Variable interne de type file [9][6]

Les valeurs des feuilles de la structure externe seront affectées après conversion successivement aux éléments de la file dans l'ordre d'accès aux feuilles dans l'analyse de la structure.

Les règles d'arrêts de transfert et de tailles actuelles de la file interne sont les mêmes que pour 4.3.1.2.

4.3.2.2. La variable interne est seule structurée.

4.3.2.2.1. Variable externe élémentaire.

Après conversion, la valeur de la variable externe est affectée à la feuille de même nom de la variable interne.

...

4.3.2.3. Les variables externes et internes sont de type structure.

L'affectation est alors une suite d'affectations après conversion entre les feuilles de même nom des deux variables, dans l'ordre d'accès aux feuilles de la structure, de la variable interne.

Remarque.

1. Si l'utilisateur veut définir des affectations plus complexes que celles citées précédemment ou en modifier les conventions, il pourra le faire au moyen des méta-modules CIVA.

4.4. Traduction de l'affectation externe - interne

4.4.1. Rappels sur les principes de compilation dans CIVA.

Un texte source écrit dans le méta-langage CIVA, le codifieur produit une chaîne codée, qui sert d'entrée au compilateur CIVA.

L'affectation externe - interne sera traitée de la manière suivante :

- A la codification, la rencontre d'une telle affectation produit les commandes de chaîne codée appropriées. (voir annexe 2 : format d'une chaîne codée).

- A la compilation de la chaîne codée, on générera le code objet associé.

4.4.2. Table de conversions de type externe en interne dans CIVA.

Avant de générer le code objet, le compilateur procède à des tests de compatibilité de types entre la variable externe et la variable interne, et génère alors les ordres de conversions par appels de sous-programmes fermés généralement, mise à part les types-dates, date 1, date 2, date 3, les autres conversions sont celles rencontrées dans les langages de programmation classiques.

...

La table des conversions possibles est donnée ci-dessous avec les notations :

oui pour conversion autorisée
non dans le cas contraire.

type externe / type interne	ch. numérique	ch. alphabétique	ch. alphanumérique	date1	date 2	date 3
Entier	OUI	NON	OUI	NON	NON	NON
Réel	OUI	NON	OUI	NON	NON	NON
Booléen	OUI	OUI	OUI	NON	NON	NON
Décimal	OUI	NON	OUI	NON	NON	NON
Caractère	OUI	OUI	OUI	OUI	OUI	OUI
Complexe	OUI	NON	NON	NON	NON	NON
Date	OUI	NON	OUI	OUI	OUI	OUI

Lorsque une conversion ne pourra être effectuée, un message d'erreur sera systématiquement édité sur imprimante.

Lorsqu'il sera nécessaire, soit de modifier les conditions de transfert, soit de définir des affectations plus complexes, notamment avec des variables internes du type structure de file ou structure conditionnelle, l'utilisateur écrira les méta-modules appropriés.

...

4.5. Analyse des arborescences [9]

Rappelons qu'à toute structure en CIVA, nous associons une information arborescente. Dans l'instruction d 'affectation (externe - interne) nous avons utilisé la convention suivante, sur les arborescences externes et internes :

. Les objets externes et internes intervenant dans un transfert ont les mêmes noms ou des noms synonymes (réf. Chapitre 3).

Pour reconnaître les objets externes de même nom (ou synonyme) que le nom d'une feuille de l'arborescence interne, il faut procéder en deux étapes :

- explorer l'arborescence pour accèder à un champ ou à une feuille.

- explorer les arborescences externes souvent à plusieurs reprises pour accèder aux feuilles satisfaisant à une condition.

Avant d'étudier plus en détail chacun de ces deux points, rappelons que l'exploration d'une arborescence dépend de sa représentation en mémoire. Toutefois, cela consiste à passer d'un mot à un K-uple de valeurs. Le mot représente l'emplacement repérant le nom de la racine de l'arborescence, les valeurs étant celles des feuilles de l'arborescence.

L'information sera donc définie par les fonctions d'accès élémentaires : lien vertical, lien horizontal, racine représentée en mémoire par un double chaînage. [3]

4.5.1. Exploration d'une arborescence. [9]

Nous utiliserons une pile attachée à l'information arborescente construite de la manière suivante :

- au départ, la pile est vide et on y fait entrer la première racine r_1 ;

...

- après entrée d'un point x, on fait entrer $r_1(x) = LV(x)$ s'il existe, sinon on fait sortir x de la pile ;

- après sortie d'un point x en tête de la pile, on fait entrer LH(x) s'il existe, sinon on fait une sortie de la pile ;

- après sortie d'un point x qui est la racine ri, on fait entrer LH(x) s'il existe,sinon on arrête l'analyse.

L'exploration consistera, pour une information interne, à vérifier si le type du champ en cours est de type file de structure, structure ou simple.

1. S'il est du type simple,plusieurs éventualités pourront se présenter :

a. soit il n'existe pas de section d'identificateurs synonymes externes-internes, alors nous le sortons de la pile ; pour la recherche de l' identificateur correspondant externe.

b. soit il existe une section d'identificateurs synonymes externes-internes alors l'identificateur n'a pas de synonyme, nous sommes dans le cas (a), sinon nous analyserons la liste des indices de la pile pour repérer l'identificateur possédant un synonyme externe unique ;

2. S'il est du type structure, l'exploration continue ;

3. S'il est du type file de structure, alors nous imposerons que les feuilles de la structure n'aient pas de synonymes internes, la structure correspond obligatoirement à un groupe d'entrée. Nous sortirons alors le type de la file et nous créerons une seconde pile associée pour l'exploration de la structure pour laquelle nous nous plaçons dans le cas 2. Lorsque l'exploration seconde pile associée est terminée, nous reprenons au point où nous en étions.

...

4.5.2. Exploration des arborescences externes.

L'analyse de l'arborescence s'effectuera par l'algorithme cité en 4.5.1. Pour chaque feuille, nous devrons tester si l'identificateur externe associé est le même que l'identificateur interne ; si oui on arrête l'exploration, sinon on continue l'exploration.

Nous donnerons au paragraphe 4.7. des exemples de méta-modules permettant d'effectuer quelques affectations externes-internes non simples.

4.6. Méta-modules de traitement des contrôles d'erreurs de fichiers externes.

Dans les chapitres précédents, notamment au chapitre 2, nous avons énoncé les différentes sortes de contrôles que nous envisagerons de traiter au cours de l'Acquisition : contrôles de séquence, de structure de l'article externe, de vraisemblance, de compatibilité individuelle ou mutuelle.

Toutefois, nous rappellerons que nous ne définissons pas d'instruction unique du langage CIVA, le nombre de cas possibles étant beaucoup trop important; citons les nombreuses possibilités d'analyse des erreurs, de rectifications automatiques de certaines erreurs détectées, de calculs statistiques.

Nous avons préféré plutôt que d'imposer des contraintes pour définir des instructions, ou de proposer de nombreuses instructions pour un grand nombre de cas particuliers, donner la possibilité à l'utilisateur de définir ses propres modules de contrôles (les instructions de CIVA et du langage étendu de CIVA et les aides à l'analyse des contrôles). Nous donnerons quelques méta-fonctions et méta-modules de base qui seront joints au méta-processeur ou disponibles dans des classes (voir instruction UTILISE CIVA).

4.6.1. Méta-module de contrôle de séquence.

L'étude faite au 2.5.1., caractérise les différents contrôles possibles pour un fichier externe sur cartes perforées. Cependant il peut être intéressant pour détecter les articles logiques en double (ou absents)

...

de pouvoir effectuer des contrôles de séquences croissantes ou décroissantes sur les réalisations d'une variable externe que nous qualifierons par la suite de clé de l'article. Nous introduisons à ce propos, une distinction entre ce que l'on appelle un indicatif et une clé : le critère de choix d'une clé est une caractéristique de la codification des réalisations externes de la variable par rapport à un tri mais pas de la signification attachée intrinséquement à l'information.

4.6.1.1. Contrôles de séquences sur clé de code carte. (réf. 2.5.1.)

Avant d'appeler un méta-module de contrôle de séquence, il importe de pouvoir tester si les paramètres effectifs lors de l'appel sont compatibles avec les clés déclarées dans la description externe.

4.6.1.1.1. Méta-fonction de test de contrôle de séquences sur code-carte

Syntaxe.

$ TEST SEQ1 (Q)

Sémantique :

le paramètre Q désigne :

soit 'T' pour le contrôle de la clé CLT de séquence du paquet de cartes

soit 'A' pour le contrôle de la clé CLA de séquence de l'article externe

soit 'G' pour le contrôle de la clé CLG de séquence d'enregistrements

La méta-fonction $ TEST SEQ1 teste si la clé déclarée dans la description de fichier externe est compatible avec la valeur du paramètre effectif de l'appel du métamodule de contrôle.

Exemple :

$ TEST SEQ1 (A) = VRAI si la clé est CLT sinon $ TEST SEQ1 = FAUX ; si la clé est compatible , alors elle prend la valeur VRAI, sinon elle prend la valeur FAUX.

...

4.6.1.1.2. Méta-module de contrôle de séquence sur clé de code carte.

Syntaxe. $ TC SEQ 1 (Q, P, C, VALI, VALF, PAS)

où Q : désigne soit 'T', soit 'A', soit 'G' comme décrit en 4.6.1.1.1.

P : désigne la valeur précédente de la clé contrôlée.

C : désigne la valeur 'courante' de la clé contrôlée.

VALI : désigne la valeur initiale de la clé contrôlée.

VALF : désigne la valeur finale de la clé contrôlée

PAS : le pas de séquence, valeur numérique positive ou négative.

SI PAS = 0 alors nous aurons :

- si VALF - VALI > 0 une séquence qui n'est pas strictement croissante
- si VALF - VALI < 0 une séquence qui n'est pas strictement décroissante.

La définition du méta-module est donnée en CIVA et META-CIVA.

Exemple : méta-module de contrôle de séquence

```
$ DEF MOD   TCONTRSEQ1  (Q,P,C, VALI, VALF, PAS)
    $ SI  $TESTSEQ1 (Q)  $ALORS ;  $TCSEQ1 (Q,P,C, VALI, VALF, PAS)
    $ FIN SI ;
$ FIN MOD ;
```

Nous allons donner un exemple de méta-module TC SEQ1 que l'utilisateur pourrait écrire.

```
$DEF MOD  TC SEQ1 (Q,P,C, VALI, VALF, PAS)
    $ LOCAL L1, L2 ;
    $ L1 = 'T', 'A', 'G' ;
    $ SI L1(1)  =  Q  $ ALORS;
    $ SI L2 < PAS   $ ALORS
```

...

Décision

VALF ≥ P	V	V	F	F	F	Sinon ;
P > C	V	F	F	V	F	
C ≤ VALI	V	F	F	F	V	

Actions

IMPRIMER1 ('ERREUR1')	1	1	1	1	1	fintable ;

$ ALLERA $ SORT ; $ FINSI ;

$ SI L2 > PAS $ ALORS ;

Décision

VALF ≤ P	V	V	F	F	F	sinon ;
P < C	V	F	F	V	F	
C ≤ VALI	F	F	F	F	V	

Actions

IMPRIMER1 ('ERREUR2')	1	1	1	1	1	fintable ;

$ ALLERA $ SORT ; $ FINSI ;

$ SI VALF > VALI $ ALORS ;

Décision

VALF ≥ P	V	V	F	F	F	sinon ;
P ≥ C	V	F	F	V	F	
C ≥ VALI	F	F	F	F	V	

Actions

IMPRIMER1 ('ERREUR2')	1	1	1	1	1	fintable ;

$ ALLERA $ SORT ; $ FINSI ;

...

Décision

```
VALF ⩽ P      |V|V|F|F|F|  sinon ;
  P ⩽ C       |V|F|F|V|F ;
  C ⩽ VALI    |F|F|F|F|V ;
```

Actions

```
IMPRIMER1 ('ERREUR')|1| 1| 1| 1| 1   fintable ;
    $ FINSI ;
$ SI  L (2) = Q  $ ALORS , .....  $ SORT : $ FIN MOD ;
```

Supposons que CLT C(4) soit une clé donnée dans la description externe.

Alors à l'appel du méta-module :

$ T CONTR SEQ1 ('T', PRE, SUIV, 1, 1 000, 2) (où le type interne PRE et SUIV seront déclarés avant l'appel) produirait le texte suivant :

décision

```
1000 ⩾ PRE    |V|V|F|F|F|  sinon ;
 PRE ⩾ SUIV   |V|F|F|V|F ;
 SUIV ⩾ 1     |F|F|F|F|V ;
```

Actions

```
IMPRIMER('ERREUR')|1| 1| 1| 1| 1   fintable ;
```

Remarque : pour effectuer une suite de contrôles sur (T, A, G) ou (T,G), (T,A), (A,G), (G), (A), (T), il suffirait de faire une suite d'appels de $ TCSEQUENC1.

4.6.1.2. Contrôles de séquence sur clé d'article logique externe.

Les contrôles précédents (4.6.1.1.) permettent un contrôle très fin de la séquence des enregistements physiques du fichier externe. Toutefois pour détecter des articles en double (ou absents) il peut être utile d'effectuer des contrôles de séquence sur une variable externe

...

autre qu'un argument code carte et servant de clé.

4.6.1.2.1. Méta-fonction de test de contrôle de séquence sur clé d'article.

Syntaxe : $ TEST SEQ2 (Q) ;

Sémantique

le paramètre Q désigne :

soit 'NSC' pour le contrôle numérique de séquence strictement croissante

'NSD' pour le contrôle numérique strictement décroissant

' NC' pour le contrôle numérique croissant

' ND' pour le contrôle numérique décroissant

'ASC' pour le contrôle alphabétique strictement croissant

'ASD' pour le contrôle alphabétique strictement décroissant

' AC' pour le contrôle alphabétique croissant

' AD' pour le contrôle alphabétique décroissant

La méta-fonction $ TEST SEQ2 teste si la classe de la clé du fichier externe est compatible avec le type du contrôle, soit numérique, soit alphabétique.

Si il y a incompatibilité $ TEST SEQ2 prend la valeur FAUX, sinon la valeur VRAI.

4.6.1.2.2. Méta-module de contrôle de séquence d'article logique.

Syntaxe : $ TC SEQ2 (Q, P, C,) ;

Q : désigne soit 'NSC', 'NSD', '-NC', '-ND', 'ASC', 'ASD', ' AC', ' AD'

P : désigne la valeur précédente de la clé en interne

C : désigne la valeur courante de la clé en interne.

Le texte de définition du méta-module $ TC SEQ2 est donné en CIVA.

...

Exemple d'utilisation du méta-module de contrôle de séquence

```
$ DEF MOD  T CONTROLSEQ2 (Q, P, C) ;
$ SI  $ TC SEO2 (Q)  $  ALORS ;
$ TC SEQ2  (Q,P,C) ;
$ FIN SI ;
$ FIN MOD ;
```

Un appel $ TCONTROL SEQ2 ('NSC', U, V) ;

(U et V supposés déclarés précédemment) produit le texte en CIVA du contrôle de séquence d'article logique.

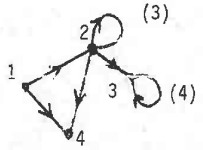
4.6.2. Méta-module de contrôle de structure de l'article.

Lorsqu'un fichier externe possède des groupes d'entrée répétitifs (ou absents) nous affectons à chaque groupe d'entrée une clé de groupe d'entrée IGE (chapitre 3). Cette clé permet l'identification d'un groupe d'entrée dans l'article logique.

Nous pouvons ainsi manipuler des structures d'information externes très générales : graphes de groupes d'entrée, que nous décrivons par une table de sélection particulière.

Pour illustrer le type de table de sélection utilisée, nous allons prendre un exemple.

Supposons que le graphe des groupes d'entrée du fichier externe dont les valeurs des clés sont pour GE1, GE2, GE3, GE4, respectivement 1,2, 3, 4, soit le suivant :



Les valeurs données entre parenthèses représentent les facteurs de répétition des groupes d'entrée 3 et 2.

...

Désignons par :

ZE = l'identificateur de groupe d'entrée courant.

IPR = le numéro de la règle précédemment exécutée en entrant dans la table de sélection, pour la première fois IPR = 0.

REP = facteur de répétition d'un groupe d'entrée.

Rétable (IPR) = est une instruction de retour dans la table qui affecte auparavant à IPR le numéro de la règle qui vient d'être exécutée.

Sélection :

	IPR = 0 et ZE = 1 et REP = 1	IPR= 1v2 et ZE = 2 et REP ⩽ 3	IPR = 2 et ZE = 2 et REP > 3	IPR = 2v4 et ZE = 3 et REP ⩽ 4	IPR = 4 et ZE = 3 et REP > 4	IPR = 1 v 2 et ZE = 4 et REP = 1	IPR = 3 v 4
REP=REP+1 ;		1		1			
Retable (IPR) ;	1	2	1	2	1	1	

fintable

Le traitement du contrôle de structure de l'article logique externe supposera que l'article soit entièrement enregistré dans la zone tampon assignée au fichier externe. Un pointeur d'accès aux zones de clés sera utilisé.

Exemple :

$ DEF MOD CONTROLS TRUCT (ZE, REP , C1, C2, C3, C4, C5, C6, C7)

Sélection	C1	C2	C3	C4	C5	C6	C7	sinon
REP=REP+1;		1		1				
nouveau ZE ;	1	2	1	2	1	1		
retable (IPR);	2	3	2	3	2	2		
$ TERR-STRUCT;								1
$ TERR-REP ;							1	

fintable

$ FIN MOD ;

...

avec : $ DEFMOD TERR - STRUCT ;

IMPRIMER1 ('ERREUR DE STRUCTURE', à ZE)

$ FIN MOD ;

$ DEFMOD TERR-REP ;

IMPRIMER1 ('ERREUR DE REPETITION', à ZE) ;

$ FIN MOD ;

Ce que nous appelons "nouveau" ZE représente l'accès par pointeur à un autre groupe d'entrée.

Remarques :

1. Le transfert dans le cas général d'un article externe en interne, utilisera un méta-module table de sélection de cette forme.

2. Lorsque l'on a exécuté une règle avant de faire l'instruction Retable, il est préférable de traiter les contrôles individuels de format, de vraisemblance, de compatibilité.

4.6.3. Méta-modules de contrôles individuels.

Les contrôles individuels doivent être exécutés dans l'ordre : classe, vraisemblance, compatibilité.

4.6.3.1. Méta-fonction de test de contrôle individuel.

Syntaxe : $ TESTMUTF (āX, Q)

Sémantique : Q désigne 'C' (classe) ou 'V' (vraisemblance) ou 'P' (compatibilité).

āX représente l'identificateur de la variable externe contrôlée.

La méta-fonction $ TESTMUTF teste si le contrôle individuel est compatible avec la déclaration du fichier externe défini.

$ TESTMUTF prend la valeur VRAI si il y a compatibilité, FAUX sinon.

...

Exemple :

$ TESTMUTF (āX, 'V') prend la valeur VRAI si la description de la variable X comporte une condition 'V' sinon FAUX.

4.6.3.2. Méta-module de contrôle de classe.

Syntaxe : $ T C C (āX)

Sémantique :

X désigne la variable externe contrôlée.

$ T C C (āX) vérifie que la classe de la suite de caractères de la valeur de la variable X est conforme avec la déclaration.

La définition du méta-module $ TCC pourra être écrite en CIVA.

Exemple :

$ DEF MOD CONTROL CLASSE (āX) ;

$ SI TEST MUTF (āX , 'C') $ ALORS $ TCC (āX) ;

$ FINSI ;

$ FIN MOD ;

4.6.3.3. Méta-modules de contrôles de vraisemblance.

Nous avons introduit deux sortes de contrôles de vraisemblance :

- par listes de valeurs,
- par module de calcul d'un argument.

4.6.3.3.1. Méta-module de contrôle de vraisemblance par listes de valeurs.

Syntaxe : $ TC V1 (āX)

X désigne la variable externe contrôlée en interne.

...

Sémantique :

$ TC V1 (ãX) vérifie que la valeur de la variable contrôlée ; X appartient à la liste.

Ce module sera écrit en CIVA.

4.6.3.3.2. Exemple de méta-module de contrôle de vraisemblance par liste.

Considérons la variable à contrôler suivante :

Z C (2) , 'V', [= (1,99)]

alors si: $DEF MOD CONTROLVL (ãX)

SI TEST MUTF (ãX,'V') $ ALORS

$ TCV1 (ãX) $ FINSI ;

$ FIN MOD ;

un appel du méta-module serait $ CONTROVL (ãZ).

4.6.3.3.3. Méta-module de contrôle de vraisemblance par "module de calcul d'un argument".

Syntaxe : $ TCV2 (ã X, U, S, W)

ãX : désigne la variable externe contrôlée.

U : désigne la position du premier caractère gauche de l'argument dans la zone où la valeur de ãX est cadrée à gauche.

S : désigne la taille de l'argument contrôlé.

W : paramètre désignant l'appel du module présent dans la zone condition du descripteur du fichier externe.

Sémantique :

$ T CV2 (ã X,U,S,W) calcule l'argument en appelant le module et le compare à la valeur externe.

...

Exemple :

$DEF MOD CONTROL VM (ãX, U, S, W)

$ SI$ SITESTMUTF (ãX,'V') $ ALORS;

$ TCV2 (ãX, U,S,W);

$ FIN MOD ;

Un appel du méta-module serait :

$ CONTROLVM (ãZ, 9,1, CALCUL);

4.6.3.4. Méta-module de contrôles de compatibilité individuelle.

Syntaxe :

$ T C P (ãX)

ãX désigne la variable externe contrôlée.

Sémantique :

$ T C P (ãX) vérifie que la condition associée à ãX est satisfaite.

Exemple :

NOM JF, 'P', [SEXE = 2 ET SIT FAM = 'MARIE'],

alors un appel de $ T C P (ã NOM JF) vérifie que la condition SEXE = 2 ET SIT FAM = 'MARIE' est satisfaite.

4.6.4. Méta-module de contrôles de compatibilités mutuelles.

Les déclarations de compatibilités mutuelles sont données dans la section % DCM . Ils ne seront traités que si tous les autres contrôles individuels n'ont pas détecté d'erreurs.

Syntaxe :

$ T C M (ãX)

ãX : désigne identificateur de relation mutuelle (défini en % DCM) à contrôler.

...

Sémantique :

un appel $ T C M (ãX) vérifie que la condition identifiée ãX est satisfaite.

Exemple de relation mutuelle :

PREST-FAMIL (INDEMNITE1 + INDEMNITE2 < 1 500)

Un appel du méta-module serait :

$ T C M (ãPRES-FAMIL).

Remarques:

Autres méta-modules utilisés à l'Acquisition de fichierx externes.

L'utilisateur devra dans le cas général écrire d'autres méta-modules, citons :

- les méta-modules d'initialisation de variables de travail : $ INIT.

Exemple :

$ DEF MOD INIT ;

Z = 0 ; TRAV = Y ;

$ FIN MOD ;

Un appel du méta-module $ INIT produirait le texte CIVA ;
Z = 0 ; TRAV = 4 ;

- les méta-modules de recherche de synonymes de variables internes déclarés alors dans la section % DSY, que nous appelerons $ SYN.

Cette recherche doit être exceptionnelle car elle s'avérera toujours très longue. En effet, pour chaque identificateur interne, il est nécessaire d'explorer la table des synonymes lors de l'exploration de l'arborescence interne. (réf. 4.5)

- les méta-modules de contrôles de dépassement de capacité dans les zones tampons, soit lors de la construction de l'article logique externe en mémoire, soit de l'article interne : $ ZTAMP.

...

4.7. Méta-modules d'Acquisition de fichiers externes.

Après avoir effectué l'analyse des données externes, la description du fichier externe, l'utilisateur a la possibilité d'écrire les méta-modules d'Acquisition de fichiers externes en utilisant les instructions de base de CIVA, les instructions du méta-langage de base CIVA 6 , et les méta-fonctions que nous avons définies précédemment.

4.7.1. Exemples d'Acquisition de fichiers externes avec et sans contrôle.

Considérons les descriptions de fichiers suivantes données dans les classes D1, D2

- Description du fichier externe

```
classe D1 ;
  % FICHIER EXTERNE,
    % DGE
E1 : NOM X (20)
     PRENOM X (10)
     SIT FAMILIALE X (11), 'V', [=(CELIBATAIRE, VEUF, MARIE, DIVORCE)]
     NBRE - ENFANT C (2), 'V', [≤10] ; $
     fin classe ;
```

- Description du fichier interne

```
classe D2 ;
  FICHIER PRODUIT
  ARTICLE (NOM file (20) car, PRENOM file (10) car,
           NBRE - ENFANT entier ;
           SIT-FAMIALE file (max = 11) ;
  fin classe ;
```

...

4.7.1.1. Acquisition sans contrôle.

L'utilisateur ne désire pas entreprendre de contrôle malgré -l'analyse des données externes effectuées-, mais uniquement faire un changement de support et de format. Il suffit d'écrire le module.

```
module  ACQ1 ;  sans contrôle sur EXTERNE
   utilise classe  D1 , D2 ;
$ ACQUISITION1  (PRODUIT, à EXTERNE) ;
fin module ;
```

Le méta-module $ ACQUISITION1 sera défini de la manière suivante :

```
$ DEF MOD   ACQUISITION1  (X, àY) ;
    $ LOCAL Z ;
    $ Z = $ NOMCHAMP (X, 1) ;
Pour chaque  Z de X faire
    $ ENTRER1 (Z, àY) ;
fpc ;
$ FIN MOD ;
```

Nous définirons le méta-module $ ENTRER 1 :

```
$ DEF MOD  ENTRER1 (Z, àY) ;
    $ LOCAL I, J, U, V ;
   LIREXT  àY ;
si  FFICH (àY) alors aller à TFIN  fin si ;
$V  = $ NOM CHAMP (à Y,1) ;
$ FAIRE I  =  1, $ NBRE FEUIL  (Z) ;
$ U  =  $ NOM FEUIL (Z, I) ;
```

...


```
    $ FAIRE  J = 1,  $  NBRE FEUIL (V) ;
      $ SI U  =  $ NOM FEUIL (V, J)  $ ALORS ;
          U  =  $ NOM FEUIL (V,J)  $ ALLERA  $ FSI ;
      $ SINON   $ J  =  $ J + 1 ;
    $ FSI : ;  $ FINSI ;  $ FIN FAIRE ;
    $ I  =  $ I + 1 ;
  $ FIN FAIRE ;
TFIN : ;  ECRIRE  Z  ;
  $ FIN MOD ;
```

Alors, l'appel de $ ACQUISITION1 dans le module ACQ1 : produirait le texte d'Acquisition CIVA suivant :

```
module   ACQ1 ;
  utilise classe D1, D2 ;
Pour chaque  ARTICLE de  PRODUIT faire
    LIREXT à EXTERNE ;
    si  FFICH (àEXTERNE) alors  allera TFIN finsi ;
        NOM = à NOM ;
        PRENOM = àPRENOM ;
        NBRE ENFANT = à NBRE ENFANT ;
        SIT - FAMILIALE = àSIT - FAMILIALE ;
    ECRIRE  ARTICLE ;
  fpc ;
TFIN : ;  fin module ;
```

Le texte du module produit ne comportant que des instructions élémentaires de CIVA.

...

4.7.1.2. Acquisition avec contrôles élémentaires.

Dans ce cas, l'utilisateur entreprend tous les contrôles définis dans la description du fichier EXTERNE,

```
module ACQ2 ; X contrôles de vraisemblance X ;
    utilisé classe D1 , D2 ;
        $ ACQUISITION2 (PRODUIT, à EXTERNE, 'V') ;
fin module ;
```

le méta-module $ ACQUISITION2 sera défini par :

```
$ DEF MOD  ACQUISITION2 (X, àY, CD1) ;
    $ LOCAL Z ;
    $ Z = $ NOMCHAMP ( X, 1) ;
        Pour chaque Z de X faire
            $ ENTRER2 (Z, àY, CD1) ;
    fpc ;
$ FIN MOD ;
```

Le méta-module $ ENTRER2 s'écrira :

```
$ DEF MOD ENTRER2 (Z, àY, CD1) ;
    $ LOCAL I, K ,V, U;
      LIREXT à Y ;
    SI  FFICH (àY) alors allera TFIN fin si ;
    $V = $NOMCHAMP (àY, 1) ;
    $ FAIRE  I = 1,  $NBRE FEUIL (V) ;
    $U  =  $ NOM FEUIL  (V,I) ;
    $ SI   $ TEST MUTF (U, CD1)  $ ALORS ;
        $ TCV1 (U) ;
```

...


```
        $ FINSI ;
            $ I =  $ I + 1 ;
        $ FIN  FAIRE;
        $ FAIRE  I = 1, $ NBRE FEUIL (Z) ;
          $U = $ NOM FEUIL  (Z, I) ;
        $ FAIRE  K = 1,  $ NBRE FEUIL  (V) ;
          $ SI  U  =  $ NOM FEUIL  (V, K)  $ ALORS ;
                U  =  $ NOM FEUIL  (V, K)  $ ALLERA  $ FSI ;
          $ SINON  $ K = $  K + 1 ;
          $ FSI  : ;  $ FINSI ;  $ FIN  FAIRE ;
          $ I  =  $ I + 1 ;
          $ FIN  FAIRE ;
    TFIN  :   ECRIRE Z ;
$ FIN  MOD ;
```

4.7.2. Formats généraux des différentes Acquisitions de fichiers externes.

Les exemples traités en 4.7.1. montrent qu'au niveau de l'analyse générale des applications d'informatique de gestion, l'utilisateur pourra déclarer des Acquisitions, sans avoir à connaître les descriptions de fichiers.

Exemples : module ACQ1 ; module ACQ2.

Le passage de l'analyse générale à la programmation s'effectuera ensuite de manière progressive et modulaire dans un langage unique CIVA (exemple 4.7.1.).

...

Alors, au niveau de l'analyse et de la programmation détaillée, il sera possible de distinguer trois classes d'élaboration d'Acquisition d'une part en fonction de l'analyse effectuée sur les fichiers externes, d'autre part en fonction de l'importance attachée à la fiabilité des informations.

4.7.2.1. Acquisitions élémentaires -sans contrôle-
qui consistent simplement en un changement de support et de format.

4.7.2.2. Acquisitions avec contrôles individuels seulement :
de classe, de vraisemblance, de compatibilité.

4.7.2.3. Acquisitions générales
où tous les contrôles sont traités : individuels et mutuels.

D'autres contrôles dépendant des particularités des applications pourront être traités dans les trois classes données ci-dessus :

- contrôles de séquence de cartes.
- contrôles de structure de l'article logique externe (identification et répétition des groupes d'entrée).

4.7.2.1. Format général d'une Acquisition sans contrôle.

Syntaxe : $ ACQUISITION (X, âY)

X désigne l'identificateur de fichier interne.

âY désigne l'identificateur de fichier externe.

4.7.2.2. Format général d'une Acquisition avec contrôles individuels.

Syntaxe : $ ACQUISITION (X, âY, I)

X : désigne l'identificateur de fichier interne.

âY : désigne l'identificateur de fichier externe.

I : désigne un des types de contrôles individuels suivants :

...

'C' : traitement des contrôles de classes seulement pour les variables externes possédant une déclaration de contrôle 'C'.

'V' : traitement des contrôles de vraisemblance.

Deux cas peuvent se présenter pour une variable externe :

. soit un seul type 'V' est déclaré, et le traitement de contrôle de vraisemblance est effectué.

. soit un type 'C' est déclaré avec un type 'V', alors le traitement de contrôle de vraisemblance est effectué seulement si le contrôle 'C' n'a pas détecté d'erreur de classe.

'P' : traitement des contrôles de compatibilités individuelles.

Trois cas peuvent se présenter pour une variable externe :

. soit un seul type 'P' est déclaré pour une variable et le traitement de contrôle de compatibilité individuel est effectué.

. soit un type 'V' est déclaré avec un type 'P' ; alors le contrôle 'P' n'est entrepris que si le contrôle 'V' n'a pas détecté d'erreur de vraisemblance.

. soit encore un type 'C' est déclaré avec un type 'P' ; alors le contrôle 'P' n'est traité que si le contrôle 'C' et éventuellement le contrôle 'V',si un type 'V' est déclaré,n'ont pas détecté d'erreurs.

4.7.2.3. Format général d'une Acquisition avec contrôles mutuels :

Syntaxe : $ ACQUISITION (X, âY, Q, CD1, CD2, ——, CDN)

X, âY et Q ont la même signification que dans 4.7.2.2.

CD1, CD2, ——, CDN désignent des identificateurs de conditions déclarés dans la section % DCM.

Les contrôles mutuels associés à CD1, CD2,——, CDN ne seront traités que si aucune erreur n'est détectée lors des contrôles individuels.

...

4.7.3. Aides à l'analyse et la programmation de méta-modules d'Acquisition de fichiers externes.

Avant de décrire une technique d'analyse et de programmation détaillées et modulaires dans le cas général d'Acquisition, nous résumons ci-dessous le processus d'analyse.

1. Définition des notions externes en entrée à partir des notions externes en sortie (chapitre 1).

2. Analyse détaillée des notions mises en évidence dans 1.et définition fonctionnelle, c'est à dire indépendante des contraintes physiques des groupes logiques d'entrée (chapitre 2).

3. Analyse de la structure du fichier interne, détermination de la structure de présentation des articles logiques externes et éventuellement définition des synonymes.

4. Analyse des contraintes dûes aux supports de fichiers externes en entrée et définition des clés de contrôles de séquence dans les groupes logiques d'entrée.

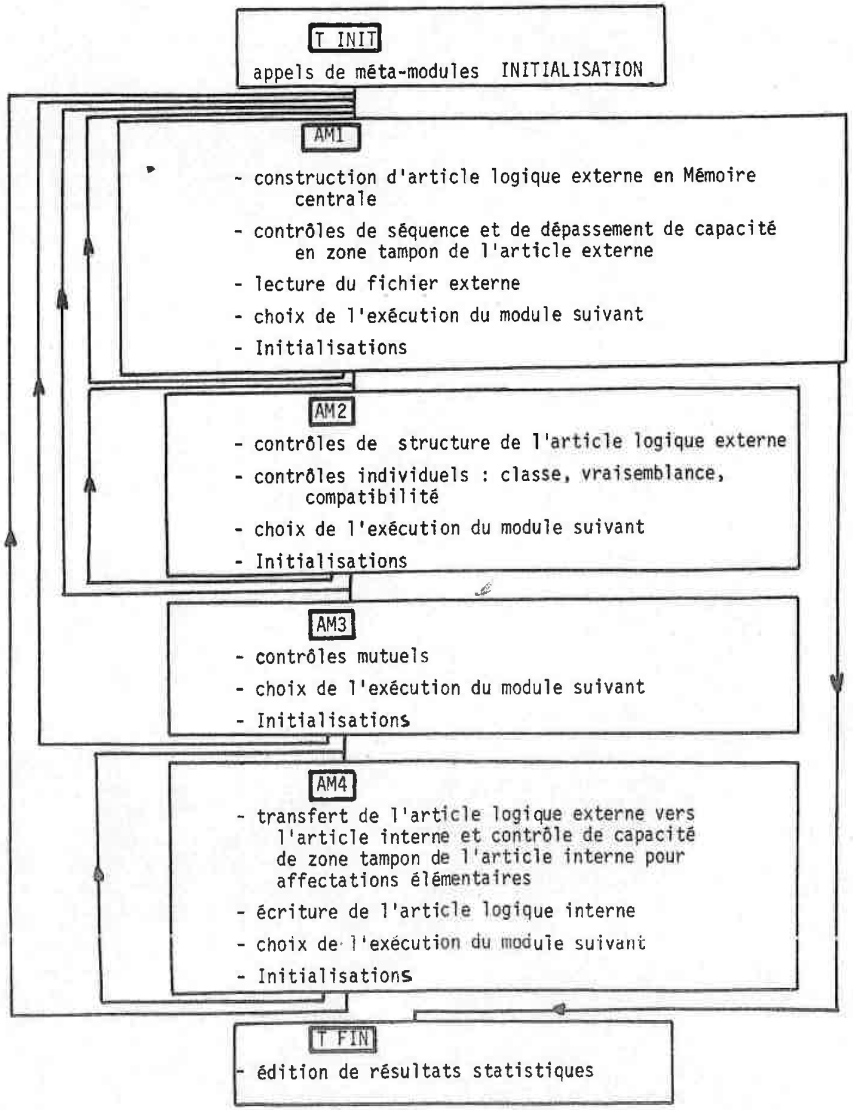
5. Analyse et définition des contrôles individuels.

6. Analyse et définition des contrôles mutuels.

7. Analyse et définition des procédures statistiques simples utjlisées pour l'estimation des différents contrôles traités.

...

L'analyse et la programmation modulaires de l'Acquisition dans le cas le plus général peuvent être décrits par le schéma suivant :

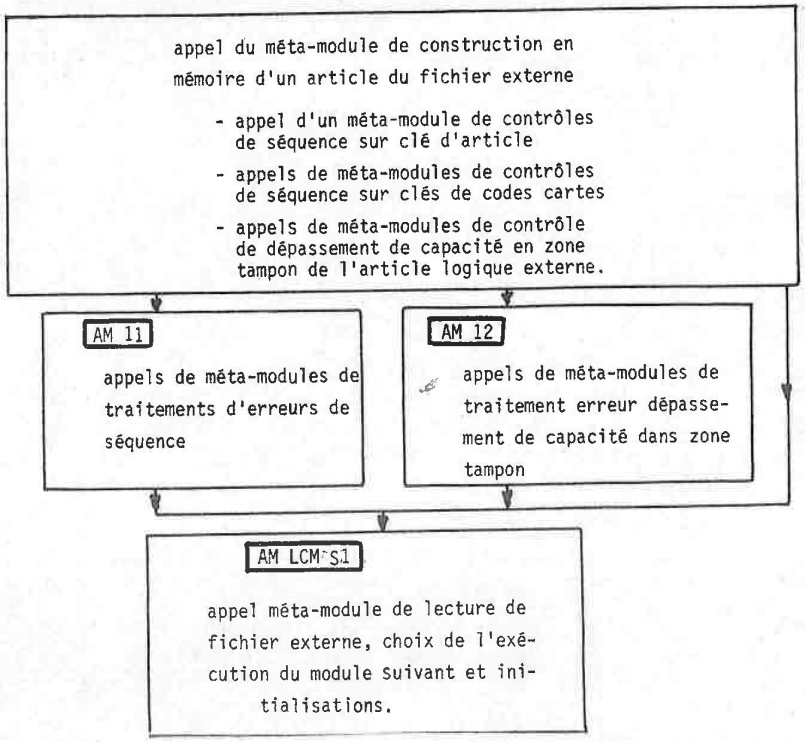


...

Chaque "phase" de traitement peut être détaillée :

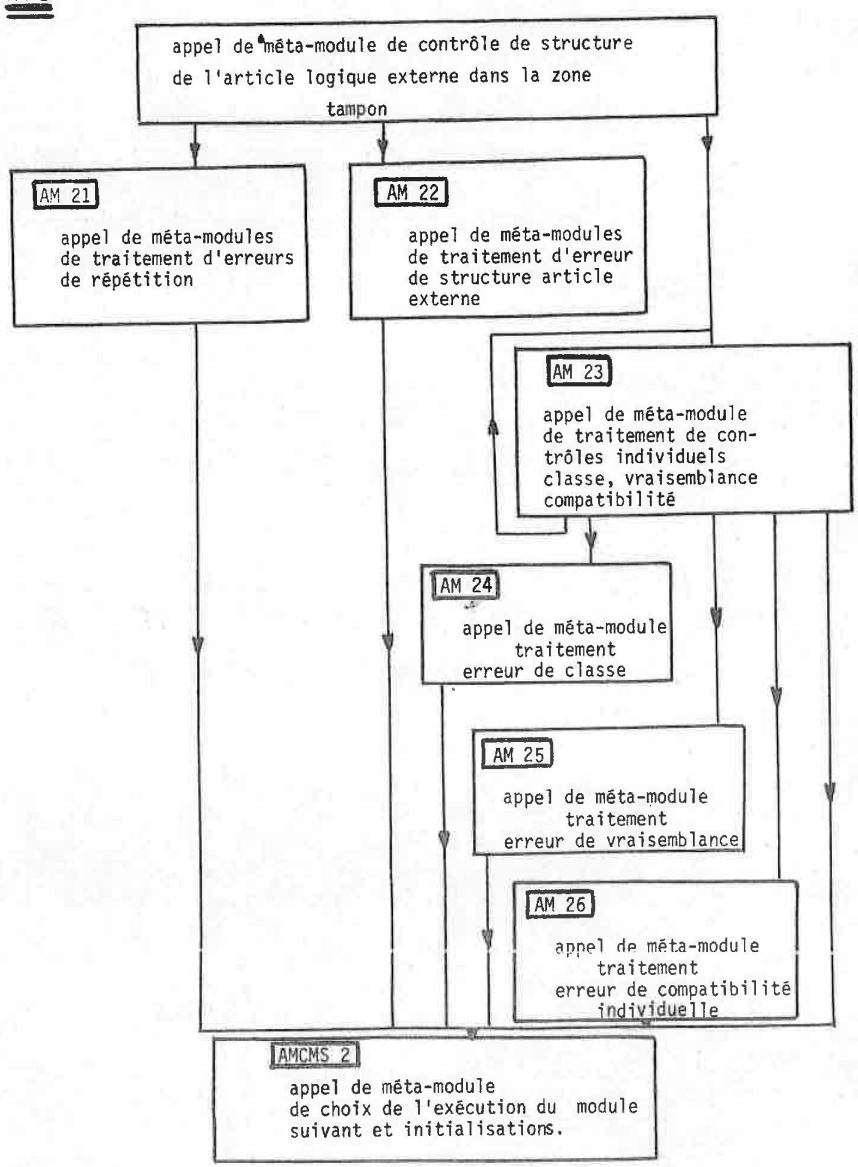
TINIT : nous effectuons dans cette phase les initialisations de variables de travail et la lecture du fichier externe.

AM1 :

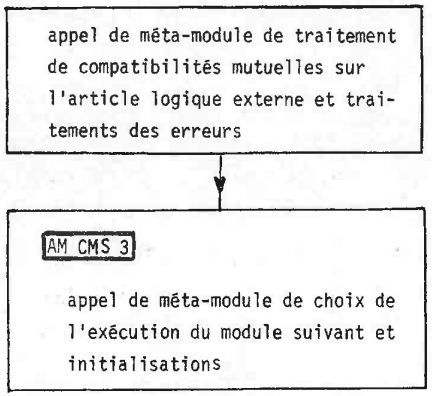


...

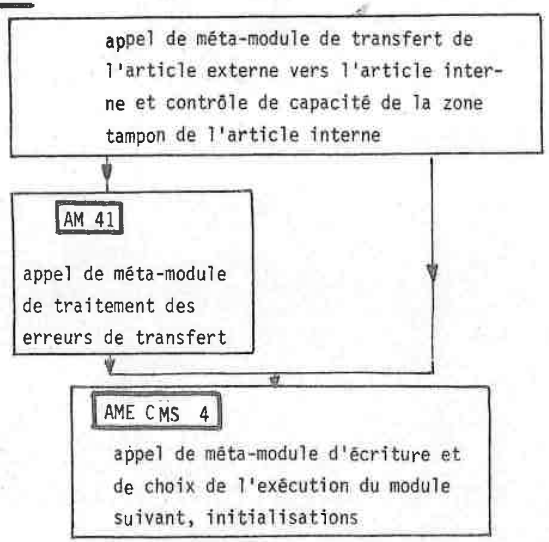
AM 2


appel de méta-module de contrôle de structure de l'article logique externe dans la zone tampon
AM 21
appel de méta-modules de traitement d'erreurs de répétition
AM 22
appel de méta-modules de traitement d'erreur de structure article externe
AM 23
appel de méta-module de traitement de contrôles individuels classe, vraisemblance compatibilité
AM 24
appel de méta-module traitement erreur de classe
AM 25
appel de méta-module traitement erreur de vraisemblance
AM 26
appel de méta-module traitement erreur de compatibilité individuelle
AMCMS 2
appel de méta-module de choix de l'exécution du module suivant et initialisations.

AM 3 :


appel de méta-module de traitement de compatibilités mutuelles sur l'article logique externe et traitements des erreurs
AM CMS 3
appel de méta-module de choix de l'exécution du module suivant et initialisations

AM 4 :


appel de méta-module de transfert de l'article externe vers l'article interne et contrôle de capacité de la zone tampon de l'article interne
AM 41
appel de méta-module de traitement des erreurs de transfert
AME CMS 4
appel de méta-module d'écriture et de choix de l'exécution du module suivant, initialisations

...

Nous ne détaillerons pas plus chacun des méta-modules donnés dans les schémas précédents car leur contenu dépend de l'application traitée. Toutefois, nous donnerons quelques précisions utiles sur deux méta-modules particuliers :

AMLCM S 1 et TFIN

AMLCM S1 : est un méta-module de lecture, de choix de l'exécution du module suivant avec les initialisations nécessaires.

Ce méta-module (comme les méta-modules AMCMS2, AMCMS3, AMECMS4) comportera une table de décision qui permettra de tester :

- la détection d'une erreur de séquence de cartes d'une erreur de dépassement de capacité ou encore de la fin de la constitution normale de l'article externe dans la zone tampon.

Dans chaque cas sont envisagés la lecture d'un enregistrement physique, (si la fin de fichier est détectée, le branchement à TFIN) et les initialisations nécessaires pour l'exécution du module suivant, soit AM1, soit AM2.

TFIN : c'est un méta-module d'édition qui ne comportera que des instructions d'écriture de résultats de calculs statistiques sur le nombre de cartes traitées, le nombre d'articles internes acquis, le nombre d'erreurs des différents types.

Remarques :

La description modulaire des traitements à l'Acquisition doit en outre donner la possibilité à l'utilisateur de mieux adapter les contrôles d'erreurs en fonction de l'importance attachée à la fiabilité des données et du coût des traitements.

En effet, les tests par jeux d'essais de chaque phase de contrôles AM1, AM2, AM3, AM4, paraissent pouvoir être mis en oeuvre assez facilement du fait de la modularité.

...

CHAPITRE 5. SERVICES D'EDITION DE FICHIERS EXTERNES

5.1. Introduction.

En CIVA, chaque fois qu'un fichier interne doit être "sorti" de l'ordinateur selon un format externe, l'utilisateur ne pourra le faire qu'avec un service d'Edition de fichiers externes.

Il importe alors de signaler que les fichiers internes seront traités séquentiellement et que tous les totaux partiels ou généraux devront être calculés en interne.

Le rôle d'un Service d'Edition est de donner la possibilité à l'utilisateur de définir facilement l'Edition et l'impression d'états de sortie simples comme des changements de supports et de formats ou plus élaborés comme nous l'étudierons en 5.6.

Aussi avons-nous rencontré, pour la conception de tels services, les mêmes sortes de problèmes que pour l'Acquisition de fichiers externes, notamment dans les transferts d'informations de format interne selon un format externe.

Il existe un trop grand nombre de conventions possibles pour définir, ici aussi, une seule instruction de transfert.

Nous définirons donc une instruction d'affectation d'objets internes selon un format externe en utilisant les conventions de transferts "courantes". L'utilisateur doit ici encore avoir la possibilité de définir des instructions plus complexes avec le méta-langage CIVA.

Nous définirons dans ce chapitre les instructions d'écriture de fichier externe nécessaire à la conception des Services d'Edition.

Les possibilités de tels services, comme nous l'avons déjà dit sont très vastes : changement de supports et de format, états de sortie avec impression de totaux partiels ou généraux, édition de tableaux statistiques, édition automatique d'un texte de documentation.

...

Dans ce chapitre, nous ne présenterons que quelques aspects possibles des Services d'Edition CIVA. Notre démarche a surtout pour but de fournir à l'utilisateur un moyen simple pour décrire les résultats de l'analyse exposée en 2.5.2. La description des fichiers externes utilisera d'ailleurs de nombreux éléments de l'Editeur COBOL [14].

5.2. Instruction d'écriture de fichier externe CIVA.

Syntaxe :

⟨instruction d'écriture de fichier externe⟩:: = ECRIREXT ⟨identificateur de groupe de sortie⟩;

Sémantique :

Cette instruction provoque l'écriture d'une ou plusieurs lignes d'imprimante composant le groupe de sortie.

Nous donnerons un exemple en 5.6.

...

5.3. Instruction d'Affectation d'objets internes à des objets externes de CIVA.

Syntaxe :

<instruction d'Affectation interne-externe>:: = à <identificateur externe>:: = <identificateur interne>;

Sémantique :

L'identificateur externe pourra désigner une variable externe éditée -avec un format d'édition- ou une variable externe non éditée.

Les variables externes en sortie seront de types élémentaires ou structures - Les variables internes seront de types simples, structures ou files.

Toutefois, dans la définition de l'affectation, nous n'envisagerons que les cas énumérés ci-dessous avec la convention suivante : si un identificateur externe est présent dans plusieurs groupes de sortie, il pourra être qualifié.

La variable externe en sortie sera toujours considérée de type externe élémentaire.

5.3.1. Variable interne de type simple ou file d'éléments simples.

- variable interne de type simple :

La valeur interne de type simple est affectée après conversion et édition à la variable externe de même nom. Les conversions autorisées seront celles couramment utilisées dans PL1 [46].

...

- variable interne de type file d'éléments simples.

Quelle que soit la nature des files (maximum, fixe ou variable), nous prendrons la convention d'affecter la file de valeurs après conversion et édition dans l'ordre de leur emplacement en mémoire à la variable externe de même nom. Le transfert s'arrête dès que le format de la variable externe est "rempli" ou dès que tous les éléments de la file interne auront été"utilisés".

Si des éléments internes n'ont pas été affectés, ils seront perdus.

5.3.2.2. Variable interne de type structure.

Nous conviendrons d'affecter la suite des valeurs des feuilles de la variable interne dans l'ordre d'accès lors de l'analyse de la structure, à la variable externe de même nom.

Remarques :

Nous n'acceptons qu'un nombre très limité d'affectations correspondant aux restrictions essentielles suivantes:les files internes ont toutes des éléments de types simples ; les structures ont toutes des feuilles de types simples.

Si l'utilisateur veut redéfinir les affectations précédentes, modifier les conventions de transfert, ou encore définir des affectations plus complexes dans le cadre d'Editions plus élaborées, il pourra le faire au moyen du méta-langage de CIVA.

5.4. Langage de description des fichiers externes CIVA en sortie.

Le langage que nous présentons doit permettre de décrire facilement les groupes de sortie résultats de l'analyse que nous avons donnés en 2.5.2.

...

Cette description, comme pour les fichiers externes de l'Acquisition commencera par % FICHIER suivi d'un identificateur externe dénotant le nom du fichier externe en sortie. Cette description est composée de deux sections :

% DPR : description de mise en page et de contrôle de variables de rupture internes

% DGS : description des groupes de sortie.

Seule la section % DGS est obligatoire, cependant si la section % DPR est utilisée, elle devra précéder % DGS.

Syntaxe :

<description fichier externe en sortie> :: = % FICHIER <identificateur fichier externe>,<suite section>

<suite section> :: = % DGS <suite groupe sortie> ;

% DPR <suite contrainte> ; % DGS <suite groupe sortie> ;

<suite contrainte> :: = <contrainte page> /

<contrainte page> , <contrainte rupture>

<contrainte page> :: = LIMIT PAGE <occur> LIGNES /

PRLIGNE <occur> DRLIGNE <occur> /

LIMITPAGE <occur> LIGNES, PRLIGNE <occur> /

LIMITPAGE <occur> LIGNES, DRLIGNE <occur> /

LIMITPAGE <occur> LIGNE , PRLIGNE <occur> , DRLINGE <occur>

<contrainte rupture> :: = <rupture logique> / <rupture page> /

<rupture logique> , <rupture page>

...

<rupture logique> :: = RUPTURE SOURCE < suite d'identificateur interne >

<suite d'identificateur interne> :: = <identificateur variable interne> /

<identificateur variable interne> (, <suite d'identificateur variable interne>)*

<rupture page> :: = RUPTURE PAGE <suite d'identificateur interne >

<suite groupe sortie> :: = <identificateur externe> : <suite groupe ligne> /

<identificateur> : <suite groupe ligne> (, suite groupe sortie)*

<suite groupe ligne> :: = <groupe sortie simple> / <groupe sortie à saut > /

<groupe sortie simple> (, <suite groupe ligne >)* /

<groupe sortie à saut> (, <suite groupe ligne >)*

<groupe sortie à saut> :: = <description saut > <groupe sortie simple> /

<identificateur externe> <description saut> <groupe sortie simple>

<groupe sortie simple> :: = <description rubrique> /

<identificateur externe> <description rubrique > /

<description rubrique> (<groupe sortie simple >) /

< identificateur externe> <description rubrique> <groupe sortie simple>

<description rubrique> :: = COL < occur1> <format> < origine>

< origine > :: = <origine interne> / <origine libellé >

< origine interne > :: = SOURCE < identificateur interne >

< origine externe > :: = VALEUR < constante externe >

< constante externe > :: = à < identificateur externe > / ' < libellé > '

< libellé > :: = < chaîne de caractères >

< chaîne de caractère > :: = < caractère EBCDIC > / < caractère EBCDIC > < chaîne caractères > (sauf caract. (') et ('))

...

<description saut> :: = SAUT <identificateur canal> / SAUT <occur>

<identificateur canal> :: = C <occur>

<occur> :: = 0 / 1 / 2 / ... / 9 / <occur> <occur>

<occur 1> :: = 0 / 1 / 2 / ... / 160

<occur 2> :: = <signe> <occur>

<signe> :: = + / -

Remarque :

Nous n'avons pas spécifié <format> car nous pouvons utiliser les spécifications de format élémentaire de PL1 avec toutes les possibilités d'édition - symboles d'insertion, de suppression et de remplacement.

5.5. Utilisation des services d'Edition.

Le but recherché est de donner la possibilité à l'utilisateur de définir dans un langage unique de manière progressive et modulaire de l'analyse générale à la programmation des services d'Edition.

Au niveau de l'analyse générale, il suffira de déclarer un module d'édition, par exemple : module EDIT ; ce module pouvant alors être présenté ultérieurement :

```
module EDIT ;
  utilise classe D1, D2, D3 ;
    $ EDITER (FICHINT, FICHEDIT) ;
fin module ;
```

...

Les classes D1, D2 contiendront les descriptions, respectivement du fichier interne FICHINT, du fichier externe FICHEDIT.

Le méta-module $ EDITER peut alors être défini dans une classe D3 de la manière suivante :

```
$ DEF MOD EDITER (X, Y) ;
  $ LOCAL Z ;
  $ Z = $NOMCHAMP (X,1) *si le fichier interne ne possède qu'un
                          seul article *
Pour chaque Z de X faire
    $ SORTIR (Z, FICHEDIT) ;
fpc ;
$ FIN MOD ;
```

Le méta-module $ SORTIR consistera alors selon les conventions retenues à traiter des transferts qui seront généralement des transferts élémentaires d'informations internes vers des groupes de sortie.

5.6. Exemple d'analyse et de description d'un fichier externe en sortie.

Considérons le problème suivant, somme toute assez caractéristique des éditions courantes dans les applications d'informatique de gestion.

Une société de distribution en gros désire obtenir un état statistique trimestriel des ventes effectuées aux différentes entreprises clientes de manière à disposer pour chaque entreprise du chiffre d'affaires : pour le trimestre, pour chaque mois, pour chaque jour et pour chaque article.

...

Nous pourrons avoir l'état de sortie ci-dessous :

ENTREPRISE xxxxx

ETAT STATISTIQUE TRIMESTRIEL DES VENTES

DATE	ENTREPOT	CODE-ARTICLE	QUANTITE	TOTAL TT
1 Octobre	003	1 421	10	158,15
	004	1 421	12	173
	007	1 421	14	3 052
		TOTAL ARTICLE	36	3 383,15
	003	1 422	8	35,12
	006	1 422	10	1 800
		TOTAL ARTICLE	18	1 835,12
		TOTAL JOUR		5 218,27
2 Octobre	005	1 423	10	53,00
8 Octobre	002	1 421	10	158,15
		TOTAL PAGE		21 315,12

Date	ENTREPOT	CODE-ARTICLE	QUANTITE	TOTAL TT
8 Octobre	003	1 421	14	173,96
		TOTAL MOIS		
		TOTAL GENERAL		

En utilisant l'aide à l'analyse décrite en 2.5.2., nous déterminons facilement sur cet état de sortie : 9 groupes de sortie que nous avons repéré par les numéros 1, 2, ... 9 .

La description du fichier externe EDITION associé à cet état de sortie serait alors :

```
% FICHIER EDITION,
    % DPR    LIMITPAGE   60 lignes,
             PRLIGNE 10,  DRLIGNE 50,
             RUPTURE SOURCE ENTR, MOIS, JOUR, ART,
             RUPTURE PAGE LR ;   * de lignes restant à écrire dans
                                                la page *
    % DGS
      ST1  :  L11  SAUT C1,
              col 18  X (10)  VALEUR  'ENTREPRISE',
              col 30  X (7)   SOURCE ENTR,
              L12  SAUT  + 1,
              col 10  X (39) VALEUR 'ETAT STATISTIQUE TRIMESTRIEL DES
                                                VENTES ';

      ST2  :  L21  SAUT + 2,
              col 1   X (4)  VALEUR 'DATE',
              L22  SAUT + 1 ,
              col 7   X (8)  VALEUR 'ENTREPOT',
              col 16  X (12)  VALEUR 'CODE ARTICLE',
              col 28  X (8)  VALEUR 'QUANTITE',
              col 40  X (10)  VALEUR' TOTAL  TT  ',
```

...

```
ST3  :  L31  SAUT + 2,
        col 1  Z9   SOURCE  JOUR,
        col 4  X(8) SOURCE  MOIS,

ST4  :  L41  SAUT + 1,
        col 7  X(8)  SOURCE ENTREPOT,
        col 16  X(12)  SOURCE ART,
        col 28 Z(3)9 SOURCE QUANTITE,
        col 40  Z(8) 9. V99  SOURCE TOTAL TT,

ST5  :  L 51  SAUT + 2,
        col 16  X(12)   VALEUR 'TOTAL ARTICLE',
        col 28  Z (5)9  SOURCE  TOT-QUANTITE,
        col 40  Z (9) 9. V99  SOURCE  TOT-ARTICLE,

ST6  :  L61   SAUT + 2,
        col 28  X(10)  VALEUR 'TOTAL JOUR',
        col 40  Z (9) 9. V99   SOURCE  TOTAL-JOUR,

ST7  :  L71  SAUT + 1,
        col 28  X (10)  VALEUR 'TOTAL PAGE',
        col 40  Z(9) 9 V.99  SOURCE  TOTAL-PAGE,

ST8  :  L81  SAUT + 2,
        col 38  X (40)  VALEUR 'TOTAL MOIS',
        col 50 Z(10) 9. V99  SOURCE TOTAL-MOIS,
```

...


```
ST9  :  L91  SAUT + 4,
        col 40  X(13)  VALEUR 'TOTAL GENERAL',
        col 55  Z(10) 9. V99   SOURCE TOTAL-GEN,

ST10 :  L10  SAUT  C1,
             SAUT  + 10 ;
```

Dans cette description figurent des variables de totaux partiels SOURCE de deux natures :

1. TOT-QUANTITE, TOT-ARTICLE, TOTAL-JOUR, TOTAL MOIS, TOTAL-GEN.

2. TOTAL-PAGE.

Il importe de bien préciser que le service d'Edition n'effectuera qu'un type de total sortie partielle : le TOTAL-PAGE sur l'indicatif de plus haut niveau est spécifié dans RUPTURE SOURCE, c'est à dire dans notre exemple ENTR.

En effet, ce total partiel de type interne incrémenté à chaque total partiel dépend uniquement de la mise en page sur l'imprimante. Un compteur de lignes (qui sera en fait un registre) restant à écrire dans la page identifiée ici par LR sera utilisé par le service d'édition. Lorsque LR deviendra égal à 5, alors le TOTAL-PAGE sera converti, édité en format externe puis imprimé. Lorsque la valeur de l'indicatif de plus haut change, nous disons alors qu'il y a une rupture sur l'indicatif ENTR le compteur TOTAL-PAGE est remis à zéro.

Les autres totaux partiels (1) sont des données du fichier interne correspondant à des ruptures d'indicatifs[1].

...

5.7. Utilisation d'une table de Décision comme aide à l'analyse et à la description de l'impression des groupes de sortie.

Le service d'édition devra analyser d'une part, pour chaque article du fichier interne les indicatifs spécifiés dans la description externe du fichier en sortie par la contrainte RUPTURE SOURCE ; d'autre part la rupture page spécifiée par RUPTURE PAGE.

En effet, en fonction des résultats des tests effectués sur les changements de valeur : (entr, mais, jour, art et lr; dans le cas de l'exemple 5.5.) qui prendront la valeur 0 s'il n'y a pas de rupture sinon 1.

Il nous a paru alors intéressant pour déterminer les impressions à entreprendre, en fonction des expressions logiques déterminées par les conditions précisées ci-dessus, d'utiliser une table de Décision, que nous illustrons sur l'exemple donné en 5.5.

...

Exemple :

PRECEDENT = 1

Décision

PRECEDENT =	0	1	10	10	10	10	10	3	3	4	4	4	4	5	5	5	6	6	6	6	8	8	8	Sinon ;
lr	-	V	F	F	F	F	F	V	F	V	F	V	F	V	V	F	V	V	V	F	V	F	F	
entr	-	-	V	V	V	V	F	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	F	
mois	-	-	V	V	V	F	-	V	V	V	V	V	V	V	V	V	V	V	F	V	F	F	-	
jour	-	-	V	V	F	-	-	V	V	V	V	V	V	V	F	V	V	F	-	F	-	-	-	
art	-	-	V	F	-	-	-	V	V	V	V	F	F	F	-	F	V	-	-	-	-	-	-	
Actions																								
ECRIREXT ST1 ;	1																							
PRECEDENT = 1;	2																							
ECRIREXT ST2 ;		1	1	1	1	1	1																	
PRECEDENT = 2 ;																								
ECRIREXT ST3 ;		2	2														1				1			
PRECEDENT = 3 ;		3															2				2			
ECRIREXT ST4 ;			3					1	1	1	1			1				1						
PRECEDENT = 4 ;			4					2		2				2				2						
ECRIREXT ST5 ;				2	2	2	2				1	1												
PRECEDENT = 5 ;				3							2													
ECRIREXT ST6 ;					3	3	3							1										
PRECEDENT = 6 ;					4									2										
ECRIREXT ST7 ;								2		2		2			1				1			1	1	
PRECEDENT = 7 ;								3																
ECRIREXT ST8 ;						4	4											1						
PRECEDENT = 8 ;						5												2						
ECRIREXT ST9 ;							5																	
PRECEDENT = 9 ;							6																	
ECRIREXT ST10 ;								3		3		3			2				2			2	2	
bi = 0 ;			5	4	5	6	7																	
PRECEDENT = 10 ;								4		4		4			3				3			3	3	
Retable ;	3	4	6	5	6	7	8	3	5	3	5	3	5	3	3	4	3	3	3	4	3	4	4	fintable

Cette table de décision pourrait être simplifiée.

Toutefois, sous cette forme elle permet de mieux voir la description du résultat de l'analyse détaillée en 2.5.2.

La variable PRECEDENT, qui doit être initialisée à 0 avant la première entrée dans la table de Décision, permet de 'retenir' dans la suite d'impression des groupes de sortie, le dernier imprimé.

La règle d'impression utilisée dans cette table, si l'on convient de noter I le ième groupe de sortie imprimé, GSK1, GSK2, ... GSKN, la suite des groupes de sortie à imprimer peut être :

SI PRECEDENT = I et CONDITION alors imprimer GSK1, imprimer GSK2, ... , imprimer GSKN.

Exemple :

dans le cas de la règle 2 de la table de Décision :

Si PRECEDENT = 1 et Ir alors imprimer ST2, imprimer ST3 ;

Remarque :

Nous n'avons pas spécifié les décrémentations du compteur de lignes restant à imprimer dans la table de Décision, précisons seulement qu'il est initialisé au nombre de lignes maximum de la page, moins le nombre de lignes réservé à l'écriture du total page.

Dans notre exemple, le compteur ligne restante LR, serait initialisé à 45.

5.8. Conclusion.

Le Service d'Edition que nous proposons n'est qu'une approche du type de Service que l'utilisateur pourra mettre en oeuvre dans CIVA.

Toutefois, les techniques d'aide à l'analyse détaillée et la programmation modulaire de l'Edition permettent de décrire dans un langage unique un Service d'Edition. Ceci est important car nous devrions obtenir une amélioration importante dans la réalisation des programmes. En outre, nous apportons une amélioration de la documentation des dossiers d'analyse, puisque nous utilisons une seule formulation.

Nous avons fait allusion en 5.1. à d'autres possibilités des Services d'Edition. Citons, l'Edition automatique de texte [45], l' Edition automatique de tableaux statistiques [42].

CONCLUSION

Le travail que nous venons de présenter n'est qu'une étape dans l'étude des problèmes d'analyse des données, d'Acquisition et d'Edition des fichiers externes.

Notre but, dans cette étude, était de montrer la faisabilité des outils proposés. Pour cela, il nous a semblé qu'il était logique de réaliser effectivement ces outils dans les cas d'Acquisition et d'Edition.

Néanmoins, plusieurs points nous paraissent particulièrement intéressants à développer dans le sens d'une plus importante formalisation, ensuite d'une généralisation des méthodes proposées et d'une plus grande automatisation des phases d'analyses.

Notamment, il serait intéressant de pouvoir entreprendre l'Acquisition d'un fichier externe logique composé de plusieurs fichiers externes.

...

Quant aux problèmes d'édition, plusieurs voies semblent intéressantes pour automatiser un peu plus l'analyse des états de sortie, l'Edition proprement dite.

En effet, c'est parce que nous avons pu assez facilement reconnaître la structure de présentation de l'état de sortie en 5.5. que nous pouvons espérer développer des heuristiques permettant d'une part de faciliter la conception de maquettes d'état de sortie, d'autre part de reconnaître une maquette à partir d'une syntaxe de présentation des objets externes (textes, tableaux, ...) en vue de l'Edition. Citons quelques systèmes d'Edition existants FORMAT [43], SCRIPT [44], [45].

ANNEXES

ANNEXE 1.

Procédure de résolution.

Les modalités pratiques nécessaires pour décrire la procédure de séparation et d'Evaluation sont les suivantes :

- Les piles d'éléments de A_p et B_p peuvent respectivement être représentées par des tableaux à une dimension IAP, IBP ; en effet,leur hauteur respective IHAUTAP et IHAUTBP est majoré par n nombre d'éléments de

- Le tableau ILPreprésentant l'ensemble des éléments qui appartiennent à L_p ou à A_pUB_p est défini de la façon suivante :

$$ILP(I)\begin{cases} = 0 & \text{si } I \in A_pUB_p \\ \neq 0 & \text{si } I \in L_p \end{cases}$$

- Pour garder la pile des informations nécessaires lors de la sélection en chaque point de l'arborescence, nous avons introduit le tableau IPILE dont la hauteur est donnée par IHAUTPILE.

- La pile des couvertures gardées est représentée par le tableau ICOUV dont la hauteur est donnée par IHAUTCOUV.

- Le poids des couvertures gardées IPOIDS sera initialisé à une valeur maximale ; à la fin de la procédure, IPOIDS représentera le poids minimum des couvertures minimales.

- Pour représenter les relations élémentaires de Φ_* nous introduisons la matrice IG (M, N) où N est le nombre d'éléments de Φ_*, M le nombre de relations de manière que, si la Ième relation est

$T((\ell_i, \ell_j)) = \ell_k$, alors IG (I, K) = 1.

- Afin de savoir si l'on analyse la branche de gauche ou de droite de l'arborescence, nous utilisons un booléen L_2 égal à 0 ou à 1.

- Le dernier élément $\ell_{j,p}$ de $\Phi_* - A_p - B_p$ choisi sera gardé dans ILJP. Dans ce cas, nous serons amené à considérer successivement $\ell_{j,p} \in A_p$ et $\ell_{j,p} \in B_p$.

```
      INTEGER P
      LOGICAL B1,B2
      DIMENSION IG(30,30),IAP(20),IBP(20),ILP(20)
      DIMENSION IPILE(100),ICOUV(100),IA6(50),IA8(5),IA7(5)
1000  FORMAT(9I2)
9000  FORMAT(36HCOUVERTURE MINIMALE DE POIDS MINIMUM)
5000  FORMAT(20I4)
4000  FORMAT(7HPILE BP,I4)
3000  FORMAT(7HPILE AP,I4)
      IHAUTAP=1
      IHAUTBP=1
      IHAUTPIL=1
      IPOIDS=100
      I6=24
      L2=1
      M=8
      N=9
      DO 1 I=1,N
    1 ILP(I)=I
      READ(105,1000)((IG(I,J),J=1,N),I=1,M)
      READ(105,1000)(IA6(K),K=1,I6)
      DO 2 I=1,N
      IS=0
      DO 3 P=1,M
    3 IS=IS+IG(P,I)
      IF(IS.EQ.0)IAP(IHAUTAP)=I;IHAUTAP=IHAUTAP+1;ILP(I)=0;
     1WRITE(108,3000)I
    2 CONTINUE
      DO 22 I=1,N
      CALL APPART(IAP,I,IHAUTAP,B1)
      IF(B1)GO TO 22
      DO 23 K=1,I6
      IF(I.EQ.IA6(K))GO TO 22
   23 CONTINUE
      IBP(IHAUTBP)=I
      IHAUTBP=IHAUTBP+1
      ILP(I)=0
      WRITE(108,4000)I
   22 CONTINUE
      ILJP=1
   20 IF(ILP(ILJP).NE.0)GO TO 21
      IF(ILJP.EQ.N)GO TO 50
      ILJP=ILJP+1
      GO TO 20
   21 DO 8 K=1,N
      IPILE(IHAUTPIL)=ILP(K)
    8 IHAUTPIL=IHAJTPIL+1
      IPILE(IHAUTPIL)=IHAUTAP
      IHAUTPIL=IHAJTPIL+1
      IPILE(IHAUTPIL)=IHAUTBP
      IHAUTPIL=IHAJTPIL+1
```

```
      IPILE(IHAUTPIL)=ILJP
      WRITE(108,5000)(IHAUTAP,IHAUTBP,ILJP,(ILP(K),K=1,N))
      IF(L2.NE.1)IPILE(IHAUTPIL)=-IPILE(IHAUTPIL)
      IHAUTPIL=IHAUTPIL+1
      IF(L2.EQ.0)L2=1;GO TO 80
      IAP(IHAUTAP)=ILJP
      IHAUTAP=IHAUTAP+1
      ILP(ILJP)=0
      WRITE(108,3000)ILJP
      GO TO 18
   80 DO 61 J1=1,M
      IF(IG(J1,ILJP).EQ.0)GO TO 61
      CALL A(J1,IA6,IA7)
      M1=1
   66 N1=IA7(M1)
      IF(N1.EQ.0)GO TO 61
      DO 63 J2=1,M
      IF(IG(J2,N1).EQ.0)GO TO 63
      CALL A(J2,IA6,IA8)
      M2=1
   65 N2=IA8(M2)
      IF(N2.EQ.0)GO TO 64
      IF(N2.EQ.ILJP)GO TO 63
      M2=M2+1
      GO TO 65
   63 CONTINUE
      CALL APPART(IAP,N1,IHAUTAP,B1)
      IF(B1)GO TO 64
      IAP(IHAUTAP)=N1
      IHAUTAP=IHAUTAP+1
      ILP(N1)=0
      WRITE(108,3000)N1
   64 M1=M1+1
      GO TO 66
   61 CONTINUE
   18 DO 74 II=1,M
      CALL A(II,IA6,IA8)
      JI=1
   73 IF(IA8(JI).EQ.0)GO TO 71
      CALL APPART(IAP,IA8(JI),IHAUTAP,B1)
      IF(B1)GO TO 72
      GO TO 74
   72 JI=JI+1
      GO TO 73
   71 DO 75 IP=1,N
      IF(IG(II,IP).EQ.0)GO TO 75
      CALL APPART(IAP,IP,IHAUTAP,B1)
      IF(B1)GO TO 75
      CALL APPART(IBP,IP,IHAUTBP,B2)
      IF(B2)GO TO 75
      IBP(IHAUTBP)=IP
```

```
103:          IHAUTBP=IHAUTBP+1
104:          ILP(IP)=0
105:          WRITE(108,4000)IP
106:       75 CONTINUE
107:       74 CONTINUE
108:       50 IS=0
109:          DO 9 K=1,N
110:        9 IS=IS+ILP(K)
111:          IF(IS.EQ.0)GO TO 30
112:          ILJP=ILJP+1
113:          GO TO 20
114:       30 IF(IHAUTAP-1.GT.IP0IDS)GO TO 15
115:          IF(IHAUTAP-1.EQ.IP0IDS)GO TO 16
116:          IHAUTC0U=1
117:          IP0IDS=IHAUTAP-1
118:       16 DO 17 K=1,IHAUTAP-1
119:          IC0UV(IHAUTC0U)=IAP(K)
120:       17 IHAUTC0U=IHAUTC0U+1
121:       15 IF(IHAUTPIL.EQ.1)GO TO 12
122:          IHAUTPIL=IHAUTPIL-1
123:          ILJP=IPILE(IHAUTPIL)
124:          IHAUTPIL=IHAUTPIL-1
125:          IHAUTBP=IPILE(IHAUTPIL)
126:          IHAUTPIL=IHAUTPIL-1
127:          IHAUTAP=IPILE(IHAUTPIL)
128:          DO 11 K=N,1,-1
129:          IHAUTPIL=IHAUTPIL-1
130:          ILP(K)=IPILE(IHAUTPIL)
131:       11 CONTINUE
132:          WRITE(108,5000)(IHAUTAP,IHAUTBP,ILJP,(ILP(K),K=1,N))
133:          IF(ILJP.LT.0)GO TO 15
134:          IAP(IHAUTBP)=ILJP
135:          IHAUTBP=IHAUTBP+1
136:          ILP(ILJP)=0
137:          WRITE(108,4000)ILJP
138:          L2=0
139:          GO TO 21
140:       12 J=1
141:       13 IF(J.GT.IHAUTC0U-1)GO TO 19
142:          WRITE(108,9000)
143:          WRITE(108,5000)(IC0UV(K),K=J,J+IP0IDS-1)
144:          J=J+IP0IDS
145:          GO TO 13
146:       19 STOP
147:          END
```

```
      1:                SUBROUTINE APPART(IA,L,ISPA,B)
      2:                DIMENSION IA(20)
      3:                LOGICAL B
      4:                DO 13 I0=1,ISPA-1
      5:                IF(L.NE.IA(I9))GO TO 13
      6:                B=.TRUE.
      7:                RETURN
      8:             13 CONTINUE
      9:                B=.FALSE.
     10:                RETURN
     11:                END
```

```
      1:                SUBROUTINE A(IJ,IA,IAA)
      2:                DIMENSION IA(50),IAA(5)
      3:                I9=0
      4:                M9=1
      5:                M92=1
      6:             10 IF(I9.EQ.IJ+1)GO TO 12
      7:             11 N9=IA(M9)
      8:                M9=M9+1
      9:                IF(N9.EQ.0)I9=I9+1;GO TO 10
     10:                GO TO 11
     11:             12 N9=IA(M9)
     12:                IAA(M92)=N9
     13:                IF(N9.EQ.0)GO TO 13
     14:                M9=M9+1
     15:                M92=M92+1
     16:                GO TO 12
     17:             13 RETURN
     18:                END
```

```
PILE BP   6
    5   5   6   0   0   0   0   0   0   7   0   0
PILE AP   7
    5   5  -6   0   0   0   0   0   0   7   0   0
    4   4   4   0   0   0   4   0   6   7   0   0
PILE BP   4
    4   5   4   0   0   0   0   0   6   7   0   0
PILE AP   6
PILE BP   7
    4   5  -4   0   0   0   0   0   6   7   0   0
    2   4  -2   0   0   3   4   5   6   7   0   0
COUVERTURE MINIMALE DE POIDS MINIMUM
    1   2   7   4
COUVERTURE MINIMALE DE POIDS MINIMUM
    1   3   5   6
 *STOP* 0
JOB STEP 03 TERMINATED AFTER 0000.01MIN
*       CORE USED  001%     DISC USED  0009    WAIT TIME 0000.01
*****************************************************************
     TIME          TIME*CORE       CORE-USE       TIME*DISC      DISC-USE
    00.28            11.64            90%            03.09           50%

  I/O-BYTES      I/O-CALLS      CARDS-READ     CARDS-PCHD      LP-PAGES
   169632            205            195             00            14
JOB TERMINATED MAPAIR20-CM1 IUCA   14.14.27.
```

ANNEXE 2.

FORMAT D'UNE CHAINE CODEE ASSOCIEE A UNE AFFECTATION DE VARIABLE EXTERNE VERS UNE VARIABLE INTERNE.

La chaîne codée produite par le codifieur sera composée d'un code suivi de la taille de la variable externe et de la liste des descripteurs des variables internes puis externes.

Le descripteur d'une variable externe (resp. interne)est un mot contenant les renseignements sur le type, la classe ou le module auxquels elle appartient, sa valeur, son adresse ou servant de pariteur vers des tables contenant les renseignements.

Exemple de chaîne codée :

| 04 | 5 | descripteur de X | = | descripteur de E11 | ; |

BIBLIOGRAPHIE

Publications sur CIVA.

[1] DERNIAME J.C.

Le Projet CIVA
Thèse d'Etat, Nancy , 1974 (à paraître)

[2] AUBRY B.

Traduction des Tables de Décision
Thèse de 3ème cycle, Nancy 1973.

[3] PERROT D.

Contribution à la Compilation dans le Projet CIVA.
Thèse de 3ème cycle, Nancy 1974 (à paraître).

[4] DUCLOY J.

Compilation dans le Projet CIVA
Thèse de Docteur Ingénieur, Nancy 1973

[5] DENDIEN J.

Gestion statique de mémoire dans un système de programmation modulaire.
Thèse de Docteur Ingénieur, Nancy 1973.

[6] BENAMGHAR L.

Instruction d'Affectation et Définition d'un méta-langage dans le Projet CIVA Nancy.
Thèse de Docteur Ingénieur, Nancy 1973.

[7] Mlle LION

Implantation des files dans le projet CIVA.
(Thèse à paraître)

[8] SCHULTZ

Tradution des "Pour chaque" dans le Projet CIVA (DEA 1971-72)

...

[9] Cours de Compilation de Monsieur PAIR, Faculté des Sciences de Nancy, présenté à l'Ecole d'Eté d'Alès.

Références techniques.

[10] C.I.I.
Système de gestion de fichiers sous SIRIS 7/ SIRIS 8

[11] CONTROL DATA - INFOL
Information oriented language - Manuel d'utilisation.

[12] ICL : FIND 2
Consultation de Fichiers et Elaboration de Rapports.
Version Multi interrogation. (Manuel d'utilisation)

[13] C.I.I.
Manuel d'utilisation COBOL SIRIS 7 / SIRIS 8

[14] C.I.I.
Manuel d'utilisation Editeur COBOL sous SIRIS 7/ SIRIS 8

Théorie des graphes.

[15] ROY B.
Algèbre moderne et Théorie des graphes
(tome 1 et tome 2) DUNOD (1970)

[16] HERVE P.
Les procèdures arborescentes d'optimisation
RIRO n° 14 (1968).

...

[17] PICARD C.F.
Graphes et questionnaires
(tome 1 et tome 2) GAUTHIER-VILLARS (1973)

[18] PAIR C.
Cours D.E.A. sur les graphes, Nancy (1968)

Analyse :

[19] REIX
L'analyse en Informatique de gestion (2 Tomes)
DUNOD (1971)

[20] MALLET
La méthode informatique
Hermann (1971)

[21] WARNIER
Construction de programmes.

Méthodes d'Analyse et de programmation.

[22] C.O.R.I.G. C.G.I.

[23] CANTOR I.C.L.

Structure d'Information.

[24] DELOBEL C.
Aspects théoriques sur la structure de l'information dans une Banque de Données
R.I.R.O. (1971)

[25] PECCOUD F.
Modsin A.F.I.R.O. (1967)

[26] LEFKOVITZ D.
File structures for on line SYSTEMS
SPARTAN BOOKS (1969)

...

[27] SALTON
Automatic Information
Organisation and Retrieval
Mc. GRAW-HILL (1968)

[28] MARTELLA G. SAMI M.G.
A proposal for organisation of large Files in
hierarchical Structure

[29] CODD E.F.
A Relational Model of Data for large Shared Data Bank
A.C.M. (1970)

[30] THIEL L.M. et HEAPS M.S.
Programm Design for Retrospective
Searches on Large Data Base
Pergamon Press (1972)

[31] JARDINE and VAN RIJSBERGEN
The use of hierarchic clustering in Information Retrieval
Pergamon Press (1971)

[32] EARLY J.
Toward an Understanding of Data Structure
Communication A.C.M. (Octobre 1971)

[33] BACHMAN C.W. et WILLIAMS S.B.
A general purpose programming System for Random Access Memories
I.D.S. General Electric Co.

Références sur les techniques de contrôle de validité.

[34] Informatique de gestion et comptabilité
Etude présentée à l'occasion du XXVIe Congrès National (1971)

...

[35] RELIN Ph.
Générateurs de programme de contrôle
C.G.I. Afcet (1970)

[36] C.O.R.I.G.
Contrôles et mise à jour d'Information

[37] P.A.C. 500
Programmation automatique CORIG
C.G.I. (1971)

[38] LETANG C.
Utilisation des vecteurs Booléens dans les programmes de contrôle
Groupe Maurice Vidal - Afcet

Références sur les bases de Données.

[39] T.D.M.S.
Time sharing Data Management System I.B.M.

[40] MIISFIT
SOCRATE
Journées Banques de Données AFCET - IRIA (1971)

Références sur d'Edition.

[41] LETANG
Utilisation de vecteurs Booléens dans les programmes d'Edition
Groupe Maurice Vidal AFCET (1972)

[42] BEKKERS Y.
Générateurs de tableaux
Université de RENNES (1973)

[43] BERNS, G.
"Description of Format, a test-processing programm"
CACM, vol. 12, March 1969

...

[44] I.B.M. Data Processing DIVISION C.P. 67/ C.M.S.
User's Guide, Cambridge Scientific Center

[45] VERGES- ESCUIN J.C. et J.P. VERJUS
Reconnaissance Automatique des Structures des Textes en vue de l'Edition
Université de RENNES (1973)

Autres références.

[46] PL1 Manuel d'utilisation. Pont à Mousson

[47] FORTRAN IV CII étendu
Manuel d'utilisation sous SIRIS 7/ SIRIS 8

[48] IFIP - ICC Vocabulary

NOM DE L'ETUDIANT : CHABRIER Jean Jacques

NATURE DE LA THESE : Doctorat de Spécialité en INFORMATIQUE

VU, APPROUVE

& PERMIS D'IMPRIMER

NANCY, le 10 Décembre 1973

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

J.R. HELLUY