"Approche méthodologique pour la production du logiciel dans l'environnement des SGBD sous Windows à partir des spécifications des traitements de la méthode MERISE"

Proposé par LARTEY MBIBOY Franklin, Analyste-programmeur, élève Ingénieur à l'IAI
Ange NAMBILA, Docteur Ingénieur en Informatique, Professeur à l'IAI
Institut Africain d'Informatique (IAI) BP 2263 Libreville - GABON

Résumé : *Du fait que la méthode Merise ne propose rien pour le passage du niveau logique au niveau physique des traitements, vue l'accroissement du nombre de SGBD sous Windows et face aux difficultés que rencontrent les développeurs dans l'environnement Windows, nous proposons ici une méthodologie permettant, à partir des spécifications des traitements de Merise, d'aboutir à la réalisation de l'application en utilisant les objets manipulés par les SGBD et les outils qu'offre l'environnement Windows.*

Mots clés : Système d'information (SI), SI informatisé, SGBD sous Windows, niveau logique, niveau physique, spécification des traitements, méthode Merise, production du logiciel, outils de développement sous Windows.

Summary : *From the fact that the Merise design method presents nothing as for what concerns the passage from the logical to the physical processing level of a system design, given the increase in number of database management systems running under Windows, and from the diffuculties system designers under Windows are faced to, we are here presenting a method that will permit the implementation of the application from the process specifications, using objects manipulated by DBMS and programming tools existing under Windows' environement.*

Key words : Information System, computerized IS, DBMS under Windows, logical level, physical level, specifications of data processing, Merise Method, software production, programming tools under Windows.

Plan de l'article

1 - Introduction

La méthode Merise reste pour l'heure la méthode de développement des SI la plus utilisée dans les pays d'Afrique francophone au sud du Sahara et ce pour plusieurs raisons :

- dans les écoles d'informatiques de ces pays elle est la plus enseignée (ex : à l'IAI),
- la majorité des sociétés des prestations des services informatiques ou les services informatiques des administrations de ces pays l'utilisent comme méthode de conception
- beaucoup d'outils d'aide à la conception ont été conçus autour de cette méthode
- etc...

Lorsqu'on parcourt la littérature consacrée à la méthode Merise, le constat reste le même à savoir, la méthode reste muette pour le passage du niveau logique au niveau physique des traitements. La

principale raison évoquée étant que la diversité des plates formes de développement à ce niveau ne permettrait pas de proposer des règles constantes. Cet aspect est laissé au choix du développeur selon l'environnement qu'il utilisera pour la production de son logiciel.
Il y a aussi que le monde de la micro-informatique est dominé à l'heure actuelle par l'interface Windows qui semble désormais être le standard dans ce domaine. D'ailleurs les SGBD actuels, souvent de type relationnel, proposent tous des outils qui sont adaptés à cet environnement.
La question qui se pose est comment faire la jonction entre les spécifications des traitements de la méthode Merise au niveau logique et la production du logiciel dans l'environnement Windows à partir des outils offerts par les SGBD : voilà ce à quoi tente de répondre cet article.

2 - Système d'information (SI) et développement du SI

Un système d'information est un ensemble cohérent d'éléments matériels ou immatériels en interaction en vue d'atteindre un objectif précis [MELESE 79]. Il est composé d'éléments (homme, machines, méthodes, etc..) chargés de stocker et de traiter les informations ; c'est la mémoire de l'organisation.
Un SI informatisé est un SI dans lequel les fonctions de mémorisation, communication et traitement ont été renforcées par l'utilisation de l'outil informatique. Son développement comporte trois parties : l'analyse, la conception et la réalisation.

- l'analyse consiste à rassembler et à interpréter les faits, à diagnostiquer les problèmes et à utiliser tous les éléments recueillis pour comprendre l'ancien système et déterminer comment l'ordinateur peut être utilisé avec le plus d'efficacité,
- la conception consiste à imaginer un nouveau système en complément ou en remplacement de l'ancien,
- la réalisation ou production consiste à générer le logiciel adapté au nouveau système.

3 - Merise comme méthode de conception et de développement des SI

Issue de l'approche systémique, la méthode Merise a pour rôle la construction des SI. Comme toute méthode de conception des SI, elle comprend trois cycles, à savoir : le cycle d'abstraction, le cycle de vie, le cycle de décision [TARD 83].

3-1 Le cycle d'abstraction.

Merise découpe la conception des SI en quatre niveaux d'invariance : conceptuel, organisationnel, logique et physique ; ce qui permet d'aller du général au particulier.
Dans chaque niveau apparaissent deux modèles : celui des données et celui des traitements. Les données sont définies indépendamment des traitements qui leur seront appliqués. Les modèles sont résumés dans le tableau ci-après :

Niveau d'abstraction	Données	Traitements
CONCEPTUEL : quoi?	MCD	MCT
ORGANISATIONNEL : qui, quand, où?	MOD	MOT
LOGIQUE	MLD	MLT
PHYSIQUE : comment?	MPD	MPT

Légende : Modèle, Conceptuel, Données, Traitements, Organisationnel, Logique, Physique

3-2 Le cycle de vie
Il traduit le caractère vivant du système. Merise décompose le cycle de vie d'un SI en trois périodes : la conception, la réalisation et la maintenance.

- La période de conception se décompose en trois étapes : le schéma directeur, l'étude préalable, l'étude détaillée.
- La période de réalisation se décompose aussi en trois étapes : l'étude technique, la production du logiciel et la mise en service.
- La maintenance : elle a pour objectif de maintenir l'application informatique en bon état de fonctionnement.

3-3 Le cycle de décision
Intégré dans le cycle de vie, il traduit les orientations majeures, les décisions à prendre tout au long du développement du SI, la planification et surtout les résultats à produire à l'issue de chaque étape du cycle de vie.

En conclusion de cette partie, comme dans [NANCI 93], la méthode Merise, n'a pas à ce jour réussi à proposer et faire adopter un formalisme pour la spécification des programmes.

4 - Les caractéristiques des SGBD sous Windows

4-1 SGBD de type relationnel
Les SGBD sous Windows sont pour la plupart de type relationnel et comprennent donc un ensemble de tables pour le stockage des données. Celles-ci peuvent être accédées grâce à des langages de manipulation des données (LMD) non procéduraux et très puissants qui leurs sont associés. Ces langages comprennent deux types de commandes : l'un pour interroger la base de données et l'autre pour la modifier des données. Ils peuvent être incorporés dans des langages de programmation hôtes (cobol, c, pascal, ...) pour en augmenter la puissance en réalisant des transactions programmées.
4-2 SGBDR sous Windows
Tirant profit des possibilités de l'interface graphique de Windows, les SGBDR ont gagné en simplicité exploitant au mieux la convivialité de cet environnement. L'utilisateur peut désormais manipuler aisément les informations contenues dans la base de données et visualiser clairement les résultats de ses recherches. La quasi-totalité des SGBDR sous Windows offrent les mêmes outils [SEHA 94] à savoir :

- un éditeur de masques de saisie. Ces masques sont généralement appelés formulaires ou fiches,
- un éditeur d'état qui permet de créer les états ou rapports,
- le QBE est présent sur presque tous ces SGBDR. Il inclut très souvent une représentation graphique des tables interrogées et de leurs liens, constituant ainsi une requête.

Les SGBDR sous Windows se différencient avant tout par :

- leur moteur, car de la capacité du moteur dépendent les performances du logiciel,
- les possibilités de programmation (macro-commandes, L4G, ...),
- les environnements de développement.

Ainsi certains SGBDR sont de véritables environnements de développement pour professionnels, alors que d'autres ne sont destinés qu'à la gestion des informations courantes.

5 La problématique du passage du niveau logique au niveau physique des traitements dans les environnements des SGBDR sous Windows

Le passage des spécifications de Merise vers un SGBD sous Windows se fait à deux niveaux : au niveau des données et au niveau des traitements.

5-1 Passage au niveau des données

Le passage du modèle logique au modèle physique ne présente pas de difficultés majeure ; il se fait soit directement par le LDD SQL qui existe dans pratiquement tous les SGBD sous Windows, soit par la description graphique des tables à l'aide des masques de description qui tirent profit des fonctionnalités offertes par l'environnement ; celles-ci génèrent automatiquement la commande SQL appropriée à la description des tables.

5-2 Passage au niveau des traitements

Le passage du modèle logique au modèle physique des traitements quant à lui doit tenir compte de l'environnement d'exécution Windows et des moyens de développement (AGL, gestionnaire d'écrans, L4G, générateur de programmes, enregistreur de macrocommandes, etc..). A ce jour, Merise ne présente aucun formalisme de spécification des traitements au niveau physique et suggère plutôt de se référer à d'autres méthodologies telle IPT-HIPO d'IBM [NAMB 93a]. Pour les SGBDR fonctionnant dans des environnements autres que Windows, on peut à ce niveau les utiliser. Par contre, aucune méthode n'existe encore qui propose un modèle qui s'adapte correctement aux SGBDR sous Windows, ce qui contraint les développeurs à réaliser des logiciels dont la maintenance s'avère être par la suite très fastidieuse, du fait de l'absence de documents de spécification des traitements et d'une méthodologie adaptée [NAMB 93b].

Nous proposons, pour surmonter cet handicap, un modèle physique des traitements adapté aux environnements des SGBDR sous Windows.

6 - Proposition d'un méthodologie.

Pour permettre le passage du modèle logique au modèle physique des traitements tout en tenant compte de la spécificité de l'environnement graphique de développement, nous proposons ici un modèle de description des traitements. Ce modèle part de la maquette de l'application (enchaînement des états et écrans), de la description des états et écrans présentée dans le MOT, de la description des traitements automatisés du MOT et de la validation des modèles externes. Il se présente à la fois comme MLT et MPT en s'articulant en trois phases :

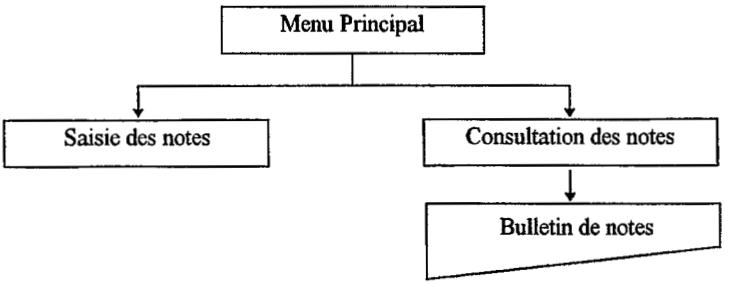
1 - Construction du diagramme d'enchaînement hiérarchique des blocs logiques d'entrée/sortie (BLE/S). Un BLE/S peut être un écran de consultation, de menu, de sélection de choix, de saisie, ou un état édité ; c'est un objet qui permet de hiérarchiser l'application de façon logique.

2 - Descriptions des BLE/S. Cette phase représente la façon dont les différents éléments sont agencés sur chaque BLE/S, la nature de ces éléments, les détails de leurs actions associées, et les différents objets de base (table, requête,...) utilisés.

3 - Enchaînement des objets de base pour les BLE/S ou leurs éléments. Dans cette phase est créé le diagramme d'enchaînement des objets qui montre comment obtenir l'objet sur lequel est basé un BLE/S ou l'un de ses éléments. Il est composé d'un enchaînement des différents objets de la base de données (tables, requêtes, codes, formulaires, états).

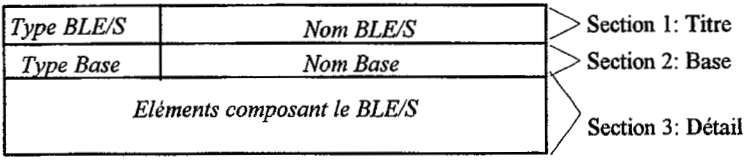
6-1) Diagramme d'enchaînement des blocs logiques d'entrée-sortie (BLES).
Pour construire le diagramme d'enchaînement hiérarchique des BLE/S, nous partirons des blocs logiques utilisés pour valider les modèles externes (ME), et du modèle organisationnel des traitements (MOT) qui décrit les traitements recensés. Ce diagramme part du tout premier écran qui doit apparaître à l'utilisateur (qui est généralement l'écran de connexion par mot de passe ou le menu principal), jusqu'aux BLE/S terminaux que sont les formulaires de saisie/consultation et les états édités. Il représente la maquette générale de l'application et tous les éléments à produire y sont représentés.
Notre méthode préconise un diagramme dont chaque élément est représenté par un rectangle pour les formulaires et un trapèze pour les états. Chaque élément contient le nom de l'objet qu'il représente.

Exemple de diagramme d'enchaînement des BLE/S : il s'agit ici d'une application de saisie des notes d'étudiants et d'édition des bulletins après visualisation de ces notes. Le diagramme d'enchaînement des BLE/S pourra être présenté comme suit :

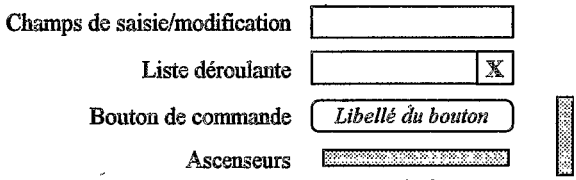


Ce diagramme indique qu'on devra créer un formulaire pour le menu principal incluant deux options à savoir *Saisie des notes* et *Consultation des notes* qui ouvriront respectivement les formulaires portant ces noms. Du *formulaire Consultation des notes*, on pourra directement imprimer l'état *Bulletin de notes* (en cliquant par exemple sur un bouton).

6-2) Description des BLE/S
Connaissant parfaitement quels sont les BLE/S (formulaires et états) que doit produire l'application, cette phase se propose de décrire de façon détaillée les différents BLE/S présentés dans le diagramme d'enchaînement des BLE/S. Chaque BLE/S sera représenté par un rectangle qui contiendra les différents éléments du BLE/S tels que les champs de saisie, les boutons de commande, les listes déroulantes, les cases à cocher, les groupes d'options, etc. On doit donc représenter chaque formulaire et état tel qu'il apparaîtra à la réalisation. Ainsi, Un BLE/S sera représenté comme suit :

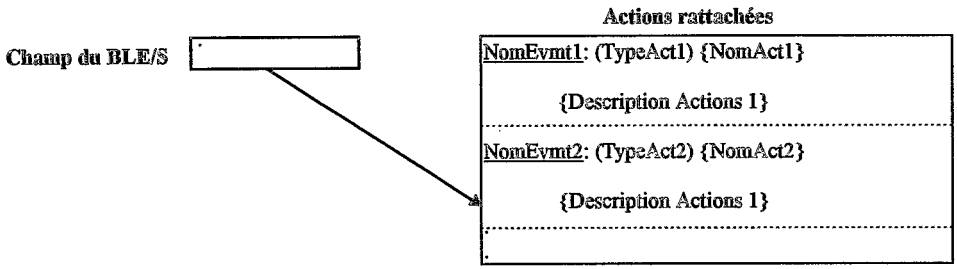


La description du BLE/S comprend trois sections.
- **Section 1 : Titre.** Elle est obligatoire et est composée de deux éléments :
Type BLE/S qui est le type du BLE/S : (F) s'il s'agit d'un formulaire et (E) s'il s'agit d'un état.
Nom BLE/S est le nom qui sera attribué au BLE/S lors de la réalisation.
- **Section 2 : Base.** Cette section n'est présente que si le BLE/S est fondé sur un objet de la base de données. Elle est composée de deux éléments :
Type base qui indique le type de l'objet de base (T = table, R = requête, V = Objet virtuel). Un objet de base est dit virtuel si l'outil de développement utilisé permet de créer un BLE/S fondé sur plusieurs tables sans auparavant créer une requête. Ainsi, ce qui devait être la requête est directement inclus dans la définition du BLE/S.
Nom base est le nom qui sera attribué à l'objet de base lors de la réalisation.
- **Section 3 : Détail.** Elle décrit les éléments qui composent le BLE/S (champs, boutons, listes déroulantes, etc.). Chaque élément sera représenté tel qu'il apparaîtra à la réalisation. On pourra aussi utiliser le formalisme suivant :



Les ascenseurs sont rereprésentés par des barres grisées.

Cette représentation peut être enrichie par d'autres types d'objets (case à cocher, groupe d'options, zone de liste, etc).
D'un élément du BLE/S peut partir un arc orienté ou en arriver un.
Un arc partant d'un élément du BLE/S indique les actions (macro-commandes, procédures ou fonctions) rattachées à ses propriétés. Ces actions sont des descriptions des opérations à réaliser lorsque certains événements surgissent (clic, entrée, sortie, mise à jour, suppression, ...). Elles sont présentées dans un rectangle placé en aval de l'arc et structuré comme suit:



Dans cette représentation,
- *NomEvt* est le nom de l'événement déclencheur (clic, maj, ...). Il est souligné pour sa mise en évidence.

- *TypeAct* est un caractère représentant le type d'action qui sera déclenché; il est mis entre parenthèses. Il existe en général trois types d'actions déclenchables représentés par:

(I) pour les actions internes au BLE/S. Il s'agit ici des procédures ou fonctions événementielles, donc propres au BLE/S,

(P) pour les procédures ou fonctions définies à l'extérieur des BLE/S,

(M) pour les macro-commandes.

Par défaut, on considérera *TypeAct* comme étant une procédure événementielle, donc de type (I).

- ***NomAct*** est le nom de l'action déclenchée. Il est obligatoire si cette action est externe au BLE/S (*TypeAct = P ou M*). Si par contre l'action déclenchée est interne au BLE/S, on peut omettre d'inscrire son nom, auquel cas on doit la décrire de façon détaillée dans la partie suivante.

- Si la fonction à exécuter est interne au BLE/S et n'est pas très longue, elle peut être détaillée dans la partie ***Description Actions***. Sinon, inscrire son nom dans la partie *NomAct* et spécifier brièvement ce que fait cette fonction, puis reporter ce nom dans la liste des objets constituée au fur et à mesure de l'avancement dans la description des BLE/S.

En amont d'un arc arrivant sur un élément du BLE/S se trouve l'objet de base de données sur lequel est fondé l'élément. Cet objet de base est représenté par un rectangle comprenant deux parties et schématisé comme suit:

Type	*Nom*

Dans cette représentation,

- *Type* est le type de l'objet de base. Utiliser **T** pour une table, **R** pour une requête et **V** pour un objet virtuel, c'est-à-dire un objet qui ne sera représenté dans la base que par sa description. On peut aussi utiliser **F** ou **E** pour spécifier un sous-formulaire ou un sous-état respectivement.

- *Nom* est le nom de l'objet. Tout objet, même virtuel doit posséder un nom.

Ce rectangle peut aussi bien comporter une seule partie, ce qui voudra dire que le contenu du champ concerné est obtenu à partir de l'instruction SQL ou de la formule qui sera inscrite dans le rectangle. Toute formule doit être précédée du signe égal (=), et toute instruction SQL doit commencer par la clause SELECT et se terminer par un point virgule (;).

Toute chose inscrite dans un champ sans être précédé du signe égal ni de la clause SELECT est considéré comme étant le nom de ce champ. Le nom d'un champ est un identifiant permettant d'adresser directement le champ dans la programmation événementielle.

Pour distinguer les noms de champs des libellés, dans la description des BLE/S, nous proposons que les noms de champs soient entourés de crochets.

Un libellé est un message adressé à l'utilisateur, lui indiquant ce qu'il doit faire. Ex: "Entrez le mot de passe : ", "Nom de l'étudiant : "

=Formule	ou	*[Nom_du_champ]*

L'objet ainsi représenté, s'il s'agit d'une requête ou d'un objet virtuel, sera ultérieurement décrit dans une autre section. Ainsi, nous proposons que soit tenu à ce niveau une liste des requêtes, procédures, fonctions, macro-commandes et objets virtuels. Cette liste sera mise à jour à chaque fois que l'un de ces éléments sera rencontré. Elle servira à ordonner la description de ces objets dans la troisième phase de notre méthode.

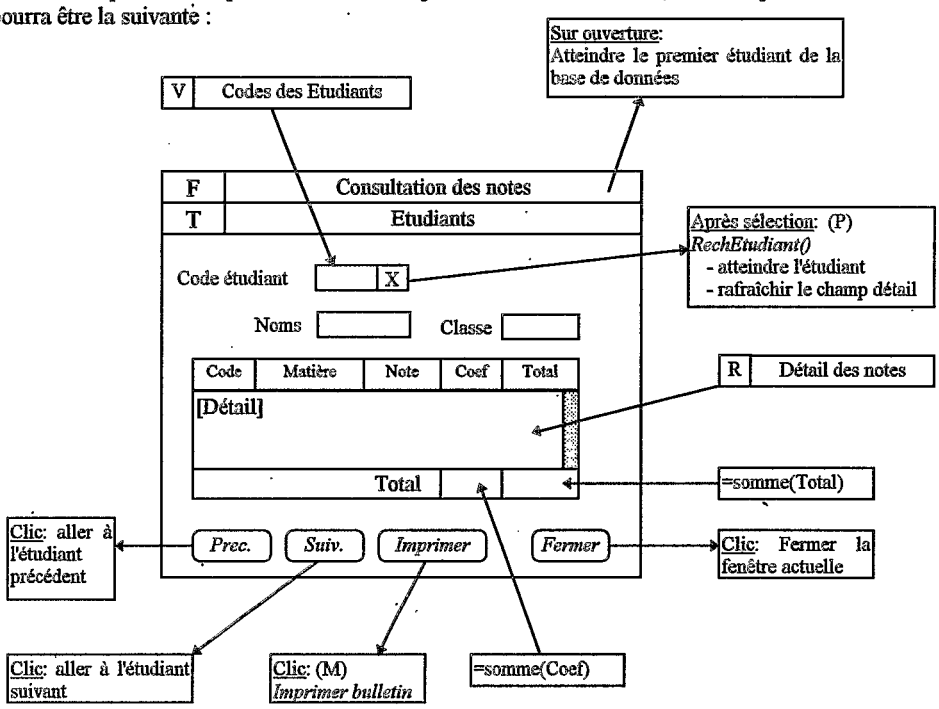
Si un BLE/S contient un sous-formulaire ou un sous-état, il serait souhaitable que ce dernier soit décrit juste après la description du BLE/S le contenant.

Hormis les champs calculés (champs contenant une formule de calcul), tout autre champ du BLE/S est directement associé à un champ de l'objet de base du BLE/S. Si un champ du BLE/S n'a pas de correspondant dans l'objet de base du BLE/S (champ indépendant), nous proposons que celui-ci soit grisé comme représenté ci-dessous :

=Formule ou *[Nom du champ]*

Exemple de description de BLE/S.

Soit le BLE/S *Consultation des notes* de l'exemple du diagramme d'enchaînement des BLE/S, en considérant que la description de l'écran à déjà été faite dans le MOT, la description de ce BLE/S pourra être la suivante :



6-3) Enchaînement des objets de base des BLE/S

Cette étape a pour but de montrer comment les différents objets de la base de données sont enchaînés en vue d'aboutir aux objets de base sur lesquels sont fondés les BLE/S ou leurs éléments. Elle comprend trois phases:

- la constitution de la liste des requêtes, objets virtuels, macro-commandes et codes de programmes à produire,
- la présentation des requêtes et objets virtuels,
- la présentation des macro-commandes et algorithmes recensés.

6-3-1 - Constitution de la liste des requêtes, objets virtuels, macro-commandes et codes à produire.
Cette phase est, dans un premier temps, réalisée pendant la description des BLE/S. En effet, lors de cette description, l'analyste doit noter les requêtes, objets virtuels, macro-commandes et codes de programmes à créer au fur et à mesure qu'il les rencontrera. Il construira ensuite une liste regroupée par catégorie d'objets (requêtes, objets virtuels, macro-commandes, codes de programmes) et ordonnée suivant l'ordre de rencontre des éléments. Si un élément de la liste est contenu dans un objet (ex: cas des procédures événementielles contenus dans les formulaires ou états), son nom dans la liste doit être précédé de celui de l'objet qui le contient suivi d'un point. Exemple: *[Consultation des notes].Bouton_Prec_Clic()* spécifie la procédure événementielle *Bouton_Prec_Clic* située dans le formulaire [Consultation des notes]. En dessous de chaque élément de la liste et un peu en retrait seront indiqués les différents objets de la base de données qui l'utilisent. Cette indication permettra en cas de modification de l'élément, de connaître l'impact de celle-ci sur les objets de la base de données qui l'utilisent.. Chaque élément de cette sous-liste sera précédée d'une lettre entre parenthèses spécifiant son type (F=formulaire, E=Etat, R=requête, V=objet virtuel, M=Macro-commande, etc).
Dans un second temps, la liste établie sera mise à jour lors de la troisième étape de notre méthode, c'est-à-dire pendant la présentation de l'enchaînement proprement dit des objets de la base de données, car dans cette étape, on pourrait être amené à créer de nouveaux objets tels que les requêtes, les codes de programmes ou les macro-commandes.

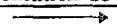
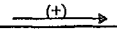
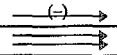
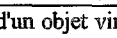
6-3-2 - Enchaînement proprement dit des objets de la base de données
Dans cette étape, chaque requête ou objet virtuel de la liste constituée précédemment sera représenté de façon graphique comme suit:

TYPE	**NOM DE L'OBJET**	ALIAS
DONNEES MANIPULEES OU OBTENUES AVEC FORMULES		
CONDITIONS		

- TYPE est le type d'objet représenté. Il peut s'agir, d'une table (T), d'une requête (R), d'un formulaire (F), d'un état (E), ou d'un objet virtuel (V).
- NOM DE L'OBJET est le nom de l'objet représenté. Si les traitements doivent être implémentés sous un SGBD qui n'autorise pas un grand nombre de caractères pour les noms d'objets, NOM DE L'OBJET doit être le nom physique, suivi d'une description de l'objet entre parenthèses.
- DONNEES MANIPULEES OU OBTENUES AVEC FORMULES est la liste des données utilisées ou résultat. Lorsque dans le cas des requêtes par exemple, l'on doit générer certaines données par des calculs, cette section doit en plus contenir les formules utilisées. Pour des données ou formules itératives, on peut utiliser des points de suspension. Exemple: ET1 ... ET25, EG_i=100*($Prés_i$/Total), ...
- ALIAS est un nom d'alias, généralement une lettre, qui sera attribuée à l'objet pour pouvoir faire la différence entre deux noms de champs identiques d'objets différents utilisés. Exemple: A.note, B.note, ... Ce champ est facultatif.
- CONDITIONS est facultative, c'est la liste des conditions que doivent vérifier des éléments des objets manipulés.

Les différents objets seront reliés entre eux par des arcs orientés (⟶), tout en respectant les règles suivantes:
Arc partant d'une table : un arc orienté d'une table vers une requête ou un objet virtuel indique que cette table est utilisée dans la requête ou l'objet virtuel.

Arc partant d'une requête :
- un arc orienté d'une requête vers une requête indique que la première requête est utilisée dans la deuxième
- un arc orienté d'une requête vers une table indique que la requête met à jour les données de la table.
La relation existant entre la requête et la table indique le type de la requête et détermine la représentation de l'arc comme résumé dans le tableau ci-dessous:

Type de requête	Description	Représentation de l'arc
Mise à jour	Modifie des enregistrements de la table	
Ajout	Ajoute des enregistrements dans la table	
Suppression	Supprime des enregistrements de la table	
Création	Crée la table et ses enregistrements	

Restriction : un arc ne peut pas être orienté d'une table vers une autre table, ni d'un objet virtuel vers une table, une requête ou un objet virtuel.
NB. Cette façon de représenter une requête est très efficace si le SGBD utilisé est doté d'un QBE (pratiquement tous les SGBD sous Windows vérifient cette condition). Il est aussi possible à ce niveau de présenter plutôt la description SQL de la requête.

6-3-3 - Présentation des macro-commandes et algorithmes ou codes de programme recensés

Les algorithmes des programmes (fonctions et procédures) recensés dans la première étape seront détaillés dans cette étape en langage algorithmique ou dans le langage de programmation utilisé.
Pour les macro-commandes, l'action sera séparée du premier argument par deux points, et les arguments seront également séparés entre eux par deux points. Si une macro-commande est contenue dans un groupe de macros (objet contenant un ensemble de macro-commandes), son nom sera formé du nom du groupe de macros suivi d'un point, puis du nom de la macro-commande. Exemple: *A.B* spécifie la macro-commande de nom B contenu dans le groupe de macros A.

6-3-4 - Application à notre exemple.

Exemple d'enchaînement des objets de base d'un BLE/S.

- Liste *requêtes, objets virtuels, macro-commandes et codes à produire*

i) Requêtes.
- Etudiant et notes
  (E) Bulletin des notes
- Détail des notes
  (F) Consultation des notes
-...

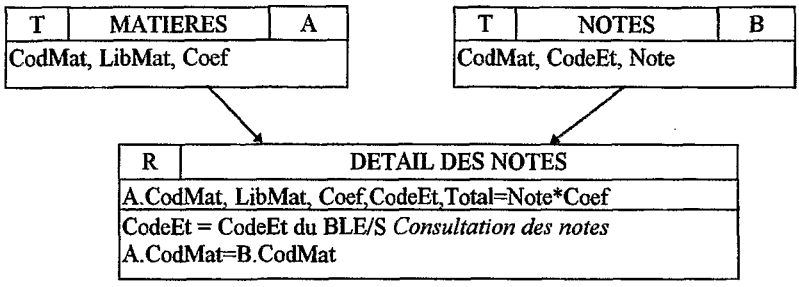
ii) Objets virtuels
- Codes des étudiants
  (F) Consultation des notes
-...

iii) Macro-commandes
- Imprimer bulletin
  (F) Consultation des notes

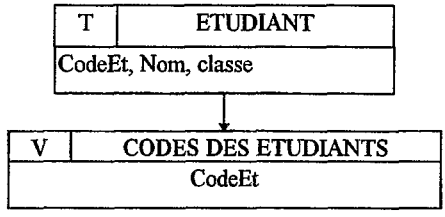
iv) Codes des programmes (procédures et fonctions).
- RechEtudiant()
  (F) Consultation des notes

♦ Enchaînement des objets.

Requête : Détail des notes.



Objet virtuel : codes des étudiants



3 - Présentation des macro-commandes et algorithmes

i) Présentation des macro-commandes.

- Macro-commande Imprimer Bulletin

OuvrirEtat: "Bulletin de Notes": En impression directe

ii) Présentation des algorithmes.

RechEtudiant() : *Procédure qui recherche un étudiant sélectionné ou saisi et affiche les informations sur lui.*

Début
** *atteindre l'étudiant sélectionné*
OuvrirTable (Etudiant)
Rechercher [CodEt]
si trouvé alors
    afficher noms et classe
    ** *rafraîchir le champ détail*
    Synchroniser [Détail] avec [CodeEt]
sinon
    Afficher message "Etudiant inexistant dans la base de données"
finsi
Fin

7- Discussions et conclusion

Le modèle que nous venons de proposer peut s'appliquer non seulement aux SGBD sous Windows, mais à tout SGBD manipulant les tables, requêtes, formulaires, états, et possédant éventuellement un langage de quatrième génération (L4G).
Pour ce qui est du passage de ce modèle à la réalisation à proprement parler, le programmeur aura à créer les requêtes et objets virtuels, et à écrire les codes de programmes :

* Les requêtes sont créées et enregistrées en tant que requêtes dans la base de données.
* Les objets virtuels sont créés comme des requêtes, mais ne sont pas enregistrés en tant que requêtes. Un objet virtuel est généralement enregistré en tant que chaîne SQL dans l'objet pour lequel il est la source.
* Un algorithme sera traduit en code de programme (procédure ou fonction) dans le langage de programmation qu'offre le SGBD. Ce code peut être directement rattaché à un objet dans le cadre de la programmation événementielle ou enregistré séparément dans un module.

L'enchaînement des objets de base s'il n'est pas continu, pourra être défini au moyen de modules, ou de macro-commandes pour les SGBD possédant la notion de macro.
La méthode présente plusieurs avantages qui sont :

→ la réalisation du diagramme d'enchaînement des BLE/S permet d'avoir une vue globale de tout ce qui sera développé dans l'application,
→ possibilité de prototypage très rapide de l'application lorsque les utilisateurs souhaitent avoir une idée précise de la future application,
→ le développement de l'application peut s'effectuer soit en "bottom-up", soit en "top-down", ou les deux, sachant que chaque BLE/S possède tous les éléments de sa réalisation,
→ la possibilité de réutiliser certaines procédures, macros, ou requêtes, grâce à leur description dans la "liste des requêtes, objets virtuels, macro-commandes et codes à produire", ceci permettant un gain de temps appréciable au cours du développement ou de la maintenance de l'application,
→ l'application peut se réaliser par partie suivant la priorité accordée par l'utilisateur,
→ etc ...

Une confrontation de la méthode dans une réalisation concrète permettra de tirer les conséquences de son applicabilité et aussi de déceler les éventuelles imperfections qui sont l'apanage de toute oeuvre humaine.

BIBLIOGRAPHIE

[CODD 70] CODD E.F, *A relational model of data for large shared data banks*, communication ACM V13, N6, Juin 1970

[DIONIS 93] Dominique DIONISI, *L'essentiel sur Merise*, Eyrolles, Décembre 1993.

[GARD 94] Georges GARDARIN, *Bases de données : les systèmes et leurs langages*, Eyrolles, Octobre 1994.

[GARD 95] Mokrane BOUZEGHOUB, Georges GARDARIN, Patrick VALDURIEZ, *De C++ à Merise Objet : concepts, langages,bases de données, méthodes, interfaces.* Eyrolles, Février 1995.

[IGL 80] IGL TECHNOLOGY, *SADT et outils associés*, IGL 1980

[MATH 90] Jean Patrick MATHERON, *Comprendre Merise : Outils conceptuels et organisationnels (cinquième édition)*, Eyrolles, 1990.

[MELESE 79] Jacques MELESE, Approche systémoque des organisations, Editions Hommes et techniques

[NAMB 93a] Ange NAMBILA, *Cours de Merise*, IAI 1993

[NAMB 93b] Ange NAMBILA, *Cours de bases de données*, IAI 1993

[NANCI 93] Dominique NANCI, Bernard ESPINASSE, Bernard COHEN, Henri HECKENROTH, *:Ingénierie des systèmes d'information avec Merise : Vers une deuxième génération*, Sybex, Décembre 1993.

[ORA 90] *Oracle RDBMS Database Administrator's Guide version 6.0*, Oracle, October 1990

[PANET 94] Georges PANET, Raymond LETOUCHE, *MERISE/2 : Modèles et techniques MERISE avancés*, Les Editions d'Organisation, Février 1994

[ROLL 87] C. ROLLAND, O. FOUCAUT, G. BENCI, *Conception des systèmes d'information : la méthode REMORA*, Eyrolles, Octobre 1987.

[SCHRY 93] Jacques de SCHRYVER, *Autoformation Visual Basic 3 pour Windows*, Editions Micro Application, Décembre 1993

[SEHA 94] Jean-François SEHAN, *Le guide expert Access 2.0 pour Windows*, Dunod, Août 1994.

[TARDI 83] A. TARDIEU ; A. ROCHFIELD. R. COLLETTI, *La méthode Merise Tome 1 : Principes et outils*, Editions d'organisation 1983