MASTER EN SCIENCES INFORMATIQUES

Conception et réalisation d'un atelier de développement dBase-III

Allamehzadeh, Abolhassan

Award date:
1990

Awarding institution:
Universite de Namur

Conception et réalisation d'un atelier de développement dBase-III

Mémoire présenté par

Abolhassan ALLAMEHZADEH

en vue de l'obtention du titre de
Licencié et Maître en Informatique

Promoteur: Professeur **Jean-Luc HAINAUT**

01/06/90

Table des matières

Je tiens tout spécialement à remercier

Le professeur Jean-Luc Hainaut sans qui ce sujet de mémoire ne m'aurait pas été confié. Il a su, malgré ses nombreuses occupations, rester disponible. Je le remercie pour ses enseignements, pour ses remarques et pour ses conseils.

Les professeurs de l'institut qui nous ont prodigué leur science.

Tout ceux et celles qui ont participé de près ou de loin à la réalisation de ce mémoire.

Chapitre 0. Introduction.

Des "environnement informatiques" dans le sens où ils rapprochent les utilisateurs peu spécialisés des outils informatiques standards peuvent faciliter l'utilisation et éventuellement élargir les applications de ces outils informatiques.

Avant de positionner les problèmes qui nous amènent à la nécessité de ces outils appelés "environnement informatiques", il est utile de rappeler et préciser quelques notions.

0.1. Rappels.

Une *donnée* est une représentation codée des propriétés d'un objet et une *information* est une signification potentielle associée à une ou des données susceptible d'influencer le comportement d'un acteur de l'organisation.

Un *système d'information* est une construction formée des ensembles suivants: informations stockées, messages, traitements et processus.

Un *modèle de structuration des données* est une collection de concepts bien définie et qui conduit à des propriétés statiques et des propriétés dynamiques. On entend par propriétés statiques des propriétés des objets proprement dits et les relations entre ces objets, tandis que des propriétés dynamiques regardent plutôt ce que font ces objets - donc des opérations - et éventuellement les relations entre ces opérations.

A un *objet*, on peut associer des propriétés appelées *attributs* (par exemple le *nom* ou une *adresse* d'un *client*), ainsi que des *contraintes d'intégrité* , c'est à dire des règles à respecter dans une structure de données.

Un *schéma* est une définition de tous les types d'objets, de leurs attributs, leurs relations et des contraintes d'intégrité.

Une instanciation d'un tel schéma dans le temps est appelé une *base de données..*

Un sous-schéma est une partie d'un schéma pour lequel une classe particulière des applications peuvent être réalisées en accédant sur cette seule partie du schéma.

Une *transaction* est constituée des opérations travaillant sur un sous-schéma et qui est atomique: Elle est exécutée complètement ou pas du tout.

Une base de données étant une instanciation d'un schéma, un modèle de structuration de données offre un langage pour la définition des objets et du schéma, ainsi que pour l'interrogation et la mise à jour de la base de données. La plupart des ces modèles offrent un langage pour définir des schémas et des sous-schémas (Data Definition Language: DDL), un langage de manipulation des données (Data Manipulation Language: DML), et un langage d'interrogation de la base de données(Query Language: QL). ces langages ne sont pas nécéssairement distincts.

Un système offrant ces langages, et en plus muni des processeurs d'implémentation des schémas et des traitements ainsi que des processeurs d'exécution des transactions et des interrogations interactives, est appelé un Système de Gestion de Base de Données (SGBD).

Un des modèles de structuration de données est celui du *modèle relationnel*. Le modèle relationnel structure les données sous forme de "schéma de relation" et ses concepts essentiels sont ceux de relation et ses attributs. Par exemple le fait qu'un *client* de nom *nom* et d'adresse *adresse* a acheté le produit de numéro *num-prod* à la date *date* est représenté sous la forme de la *relation* suivante: client (*nom, adresse, num-prod, date*).

A la conception d'une application, surtout quand il s'agit d'une application compliquée, la même réalité peut être décrite par plusieurs schémas selon des utilisateurs différents. Ces schémas se distinguent par exemple selon leurs performances. Le modèle relationnel de structuration des données

tend à caractériser le "bon" schéma et à trouver la procédure pour le déduire et ce grâce à la théorie des dépendances fonctionnelles et de la décomposition. On peut citer [DATE] comme une référence de la théorie relationnelle.

Parmi des SGBD actuellement disponibles, les SGBD relationnels ont des propriétés remarquables. C'est par exemple le cas de dBase-III d'Ashton-Tate qui va être discuté au chapitre 4.

En ce qui concerne le développement d'une base de données on distingue les trois phases suivantes: L'analyse conceptuelle, la conception logique et la conception physique qui sont développées au chapitre 1. Les SGBD et les langages de programmation sont bien sûr, à la disposition de l'utilisateur durant la phase de conception physique. Ces outils appartiennent à un niveau trop peu élevé et assez compliqué pour être utilisés directement. Alors il faut des outils dans la phase de la conception logique (donc des modèles et des pseudo-langages) afin de permettre d'avoir la possibilité d'exprimer un schéma conceptuel dans cette phase. Le Modèle d'Accès Généralisé (**MAG**) et le Langage de Description d'Algorithmes (**LDA**) sont des exemples de ces outils dans la phase de conception logique. De même qu'une application d'une base de données dans la phase de conception physique est exécutable par une machine réelle, dans la phase de conception logique aussi, et à l'aide de ces outils intermediaires, une application peut parfaitement être exécutée par une machine abstraite.(Possibilité de rédiger des algorithmes en pseudo-langage, possibilité d'accès etc.).

0.2. L'idée de réalisation d'un atelier.

L'idée de réalisation d'un atelier de développement d'un SGBD est donc d'aider l'utilisateur à la conception d'une base de données et une utilisation plus facile et éventuellemt moins chère d'un SGBD.

De nombreuses recherches ont été consacrées à la réalisation d'ateliers pour ces différentes étapes de conception des systèmes d'information. Aujourd'hui la réalisation des ateliers profit des outils informatiques de 4 ème génération et des techniques évoluées d'intelligence artificielle.

Dans la première partie de ce mémoire, on essaye de montrer la démarche de conception d'application d'une base de données et la définition générale d'un atelier de développement et ses principes.En deuxième partie on trouvera le développement d'une base de données dBaseIII, les modèles utilisés, les principes de traduction de schémas, et une description complète des fonctions de l'atelier. On y trouve aussi la conception globale et détaillée des modules qui composent l'atelier.

Un mémoire parallèle réalisé par Jan WASIAK, présente le développement de programmes dBase et donne les détails de la partie dynamique de la conception des bases de données.

Partie I :

NOTION D'ATELIER DE DEVELOPPEMENT

Chapitre 1: Démarche de conception d'applications sur une base de données

D'une manière générale, la démarche de conception d'une base de données consiste essentiellement en trois phases. A chaque phase correspond un schéma qui est le résultat de sous-démarches de la phase. On peut dire qu'une démarche de conception d'applications sur une base de données consiste en un ensemble de processus de transformation.

Ce découpage en phase est le résultat des recherches en base de données développées dans [BENCI, 74] et [ANSI, 75] on trouve les phases suivantes: l'*analyse conceptuelle*, la *conception logique* et la *conception physique*.

L'analyse conceptuelle:

L'analyse conceptuelle repose sur un modèle de structuration des informations de la mémoire du système d'information (par exemple modèle Entité/Association qu'on verra au chapitre 4) et sur des modèles de structuration des traitements. Cette phase analyse les besoins des utilisateurs sous forme d'un schéma conceptuel.

La démarche d'analyse conceptuelle elle même se décompose en trois sous-phases fondamentales: la construction du sous-schéma conceptuel de chaque phase de traitement, l'élaboration du schéma conceptuel global du projet et le relevé des quantifications. Globalement, le résultat de ces trois sous-phases sera la description du système d'information. Cette description du système d'information est indépendante des outils informatiques et aide le concepteur de la base de données à exprimer la sémantique des données. [BOD-PIG, 83/88] donne des détails d'une étude approfondie sur l'analyse conceptuelle.

La conception logique:

Cette phase produit un schéma logique à partir du schéma conceptuel (c'est à dire le schéma conceptuel des données et celui des traitements) tout en reprenant la sémantique exprimée dans le schéma conceptuel. Ce schéma logique décrit les besoins des applications en termes d'accès logiques tout en étant conforme à un SGBD.

[HAINAUT, 86] explique une démarche générale de la conception de base de données ainsi que son application à différents SGBD.

La conception physique:

Une traduction des spécifications de la phase précédente à une base de données opérationnelle sera faite durant la phase de la conception physique. C'est durant cette phase qu'on traduit des procédures de la phase logique en des procédures du langage exécutable sur des machines réelles.

Chapitre 2: Atelier de développement d'application d'une base de données.

2.1: Objectifs:

Un atelier de développement d'application a pour objectif d'assister l'élaboration de descriptions correctes de base de données et de programmes. Cette assistance consiste à automatiser une partie du travail à effectuer pour une utilisation correcte et plus rapide d'un système de gestion de base de données.

2.2 :Fonctions

Pour un atelier de développement d'un SGBD, les fonctions essentielles sont:

2.2.1 : fonction de documentation:

Cette fonction a pour but de permettre l'introduction et la gestion des spécifications d'une base de données.

Un rôle important de cette fonction sera de permettre des modifications éventuelles des descriptions stockées, et d'éviter de refaire une saisie complète d'une base de données suite à une correction éventuelle.

Après avoir introduit les informations de descriptions de base de données, on aura donc la possibilité de consulter, de manipuler et mettre à jour la base de données introduite. Cette fonction permet aussi la production d'un rapport contenant les principaux objets de la base de données utiles à l'utilisateur.

2.2.2 : Fonction de validation :

En général, un atelier propose à l'utilisateur un modèle de structuration de données propre à l'atelier. Ce modèle de structuration de données impose une série de contraintes et de limites par rapport au modèle général de structuration de données.

Ces contraintes imposées par un atelier exigent d'être vérifiées. La fonction de validation vérifie toutes les règles et les contraintes du modèle.

Les contraintes ayant des natures et des origines différentes, seront vérifiées soit au moment de la saisie des données, soit au moment de l'exécution d'une autre fonction de l'atelier.

2.2.3. Fonction de génération :

Les règles de transformations sont la base des fonctions de génération. En effet c'est grâce à une série de règles de traduction qu'à partir des descriptions conceptuelles, on atteint une descriptions de base de données.

La fonction de génération implémentée sur base des règles de transformation, peut traduire un schéma conceptuel introduit par la fonction de documentation et vérifié par la fonction de validation à un schéma conforme au SGBD.

2.3 : Architecture.

Le figure 2.1 donne une idée des différentes composantes de l'atelier et de leurs communication avec différents objets éxterieurs. En effet l'atelier et l'utilisateur communiquent l'un avec l'autre pour la saisie des données et l'obtention des résultats. L'atelier génère un rapport sur une base de données, il génère aussi des codes objets (c'est-à-dire des programmes en langage propre au SGBD), ainsi que la description d'une base de données conforme à ce SGBD. Pour faire tout cela l'atelier reçoit les textes sources des programmes et les descriptions des bases de données.

2.3.1. Base de spécifications.

La base de spécifications étant la source commune pour tous les processeurs de l'atelier, elle joue un rôle central dans l'architecture de l'atelier. Elle contient les descriptions de schémas conceptuels introduits par l'utilisateur et touts les processeurs de l'atelier pour être exécutée, cherchant des informations dans cette base. Il faut ajouter que les descriptions de schémas transformés et les textes de programmes peuvent être stockés aussi dans la base de spécifications.

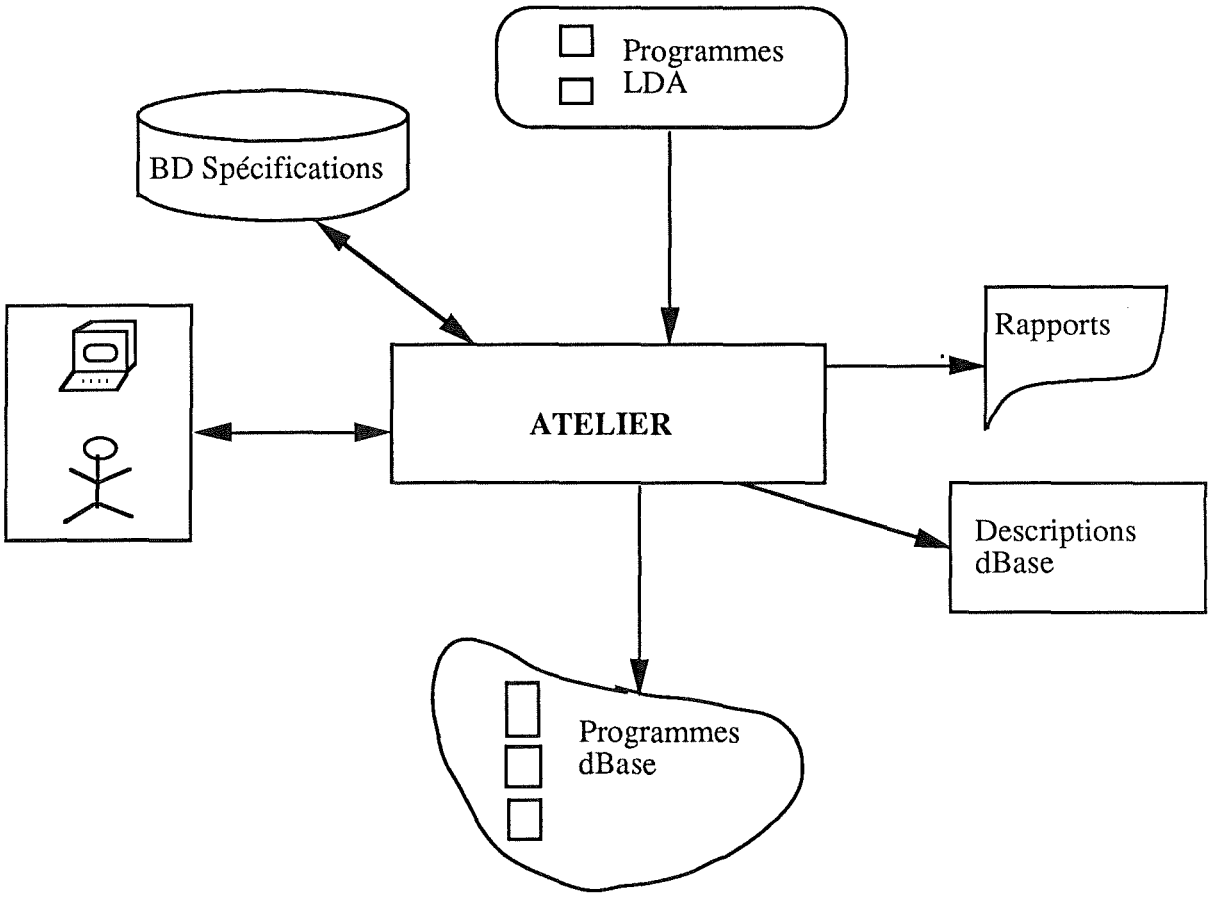


Fig 2-1. Architecture générale d'un atelier de développement

2.3.2. Processeurs fonctionnels.

Les différentes fonctions décrites ci-dessus sont prises en charge par des processeurs indépendants les uns des autres et opérant sur le contenu de la base des spécifications. Les principaux processeurs sont :

Monitoring et présentation : Ce processeur contrôle la présentation sur écran ainsi que le dialogue avec l'utilisateur.

Navigateur : Assure l'accès aux informations de la base des spécifications en fonction des demandes de l'utilisateur (accès par nom, par liste, par pointage, etc).

Chargeur de spécifications externes : Introduit dans la base des spécifications des informations issues de sources extérieures.

Processeur de modification : Réalise les modifications dans la base des spécifications en en garantissant la cohérence.

Vérificateur de conformité : Analyse les spécifications par rapport à un jeu de contraintes prédéfinies.

Générateur de rapport: Produit divers rapports.

Processus de saisie : Réalise la saisie des principaux objets de la base de données.

Générateur de schémas : Produit le schéma conforme à un SGBD.

Générateur de descriptions exécutables : Produit le texte dans le langage d'un SGBD.

Chapitre 3. Principes d'un atelier de développement dBase:

3.0. Introduction

Les principes de développement dBase sont bien sûr, l'instanciation de principes généraux et en particulier, l'architecture générale d'un atelier de développement. L'hypothèse d'utilisation de cet atelier par des machines légères et des utilisateurs non professionnels, nous permet de simplifier le modèle de structuration des données et celui du langage de programmation. C'est le cas aussi de la simplification de l'interface utilisateur et les fonctions de l'atelier.

3.1. Introduction aux modèle E/A et dBase.

Le modèle Entité/Association, en tant que modèle de structuration des données, est le modèle destiné à produire une description conceptuelle d'une base de données qui est le plus populaire à l'heure actuelle. L'existence de représentations graphiques rend le modèle particulièrement attractif comme support de communication.

Le modèle Entité/Association est un modèle qui permet de structurer les informations à l'aide des concepts essentiels d'entité, d'attribut, d'associations et d'identifiant.

Pour l'atelier à élaborer, on a choisi ce modèle E/A, dont la description détaillée est abordée au chapitre suivant. On a choisi dBase-III comme SGBD à développer.

DBase-III d'Ashton-Tate étant un système de gestion de bases de données, il offre à l'utilisateur la possibilité de concevoir et de gérer une base de données. En même temps il offre également des structures algorithmiques qui permettent de rédiger des programmes "base de données". Le langage de programmation dBase utilise beaucoup de codes, qui peuvent être sources d'erreurs.

3.2. Les choix de réalisation.

Le système de gestion de base de données NDBS est choisi pour cette base de spécifications. Dans [HAINAUT,87] les explications détaillées sur ce SGBD sont présentées. Il suffit ici de rappeler que le choix du langage de programmation PASCAL et le système d'exploitation MS-DOS pour cet atelier est attaché au choix de NDBS parce que NDBS est un SGBD pour des micro-ordinateurs et ses primitives sont utilisables en PASCAL.

Partie II

DEVELOPPEMENT D'UNE

BASE DE DONNEES dBASE

Chapitre 4. Les modèles:

4.1. Introduction.

Comme on l'a vu au chapitre précédent, le modèle Entité/Association est choisi comme modèle de structuration des données ainsi que le système de gestion de base de données dBase comme le SGBD à développer. Dans ce chapitre, outre une explication plus détaillée sur ces deux outils, on verra également les contraintes imposées par l'atelier sur le modéle E/A - c'est à dire le modèle E/A restreint qui est conforme au modèle de notre atelier- et aussi une explication sur la représentation d'un schéma dans la base de spécifications.

4.2. Modèle de spécification de structure de données.

Concepts d'entité, d'attribut, d'association et d'identifiant sont les concepts principaux du modèle de spécification de structure des données appelé modèle Entité/Association.

Entité et type d'entité:

La notion fondamentale est celle d'*entité.* Une entité est une chose concrète ou abstraite, ayant une existence autonome dans un domaine d'application. Les entités sont partitionnées en classes appelées *type d'entité.*

Comme l'indique la figure 4.1., chaque type d'entité est représenté graphiquement par une cartouche où figure le nom du type d'entité.

Attribut d'un type d'entité:

Toutes les propriétés d'un type d'entité sont représentées par des attributs. Par exemple un produit est caractérisé par un numéro, un libellé, un prix et une quantité du stock, on dira que le type d'entité PRODUIT est doté des attributs NPRO, LIBELLE, PRIX, QSTOCK. Le nom d'attribut comme l'indique la figure 4.1. sera inscrit dans la partie inférieure du rectangle qui présente le type d'entité.
Un *identifiant* d'un type d'entité est un attribut qui identifie les entités
de ce type. Par exemple PRODUIT possède un attribut NPRO tel qu'à chaque

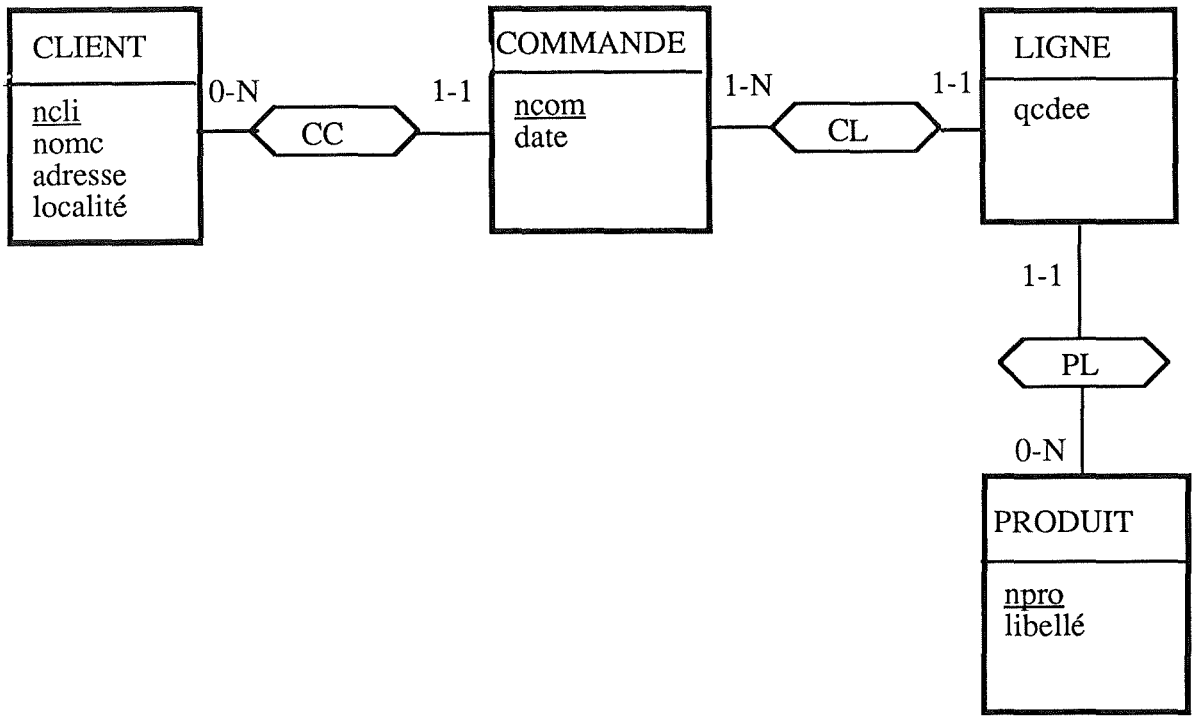


Fig 4-1. Représentation des types d'entité, des attributs et des types d'association.

entité PRODUIT correspond une valeur distincte. On dira que NPRO est l'identifiant de produitt.

Association et type d'association:

Une association est définie par une correspondance entre deux ou plusieurs entités (non nécéssairement distinctes) où chacune assume un rôle donné. Les associations sont classées en type d'association.

Un type d'association est représenté graphiquement par un hexagone relié par des segments de droites aux rectangles qui représentent les types d'entité sur lesquels on définit le type d'association. Dans l'hexagone, on indique le nom du type d'association.

4.3. Les principales contraintes de validation

Les contraintes de validation de l'atelier, seront dues au modèle choisi. En effet le modèle E/A que nous allons utiliser, comprend les notions de TYPE d'ENTITE,TYPE d'ASSOCIATION,d'ATTRIBUT et des contraintes d'intégrité d'IDENTIFIANT et celle de CONNECTIVITE. Les préconditions sur des schémas Entité/Association sont:

-Pas de type d'association d'un degré plus de 2 (pas de type d'association ternaire ,quaternaire,etc..).

-Pas de type d'association avec attribut.

-Des attributs doivent être élémentaires (non-décomposables) et simples (non-répétitifs).

-Deux types d'entité reliés à un type d'association doivent être distincts (pas de type d'association récurssif)

-La connectivité des deux côtés d'un type d'association doit être 0-1 et 0-N. On appellera le type d'entité du côté de la connectivité 0-1 type entité origne,et celui du côté 0-N type d'entité cible.

-Chaque type d'entité origine posséde obligatoirement un identifiant parmi ses attributs. c'est pour les transformations ultérieures à un schéma conforme à dBase.

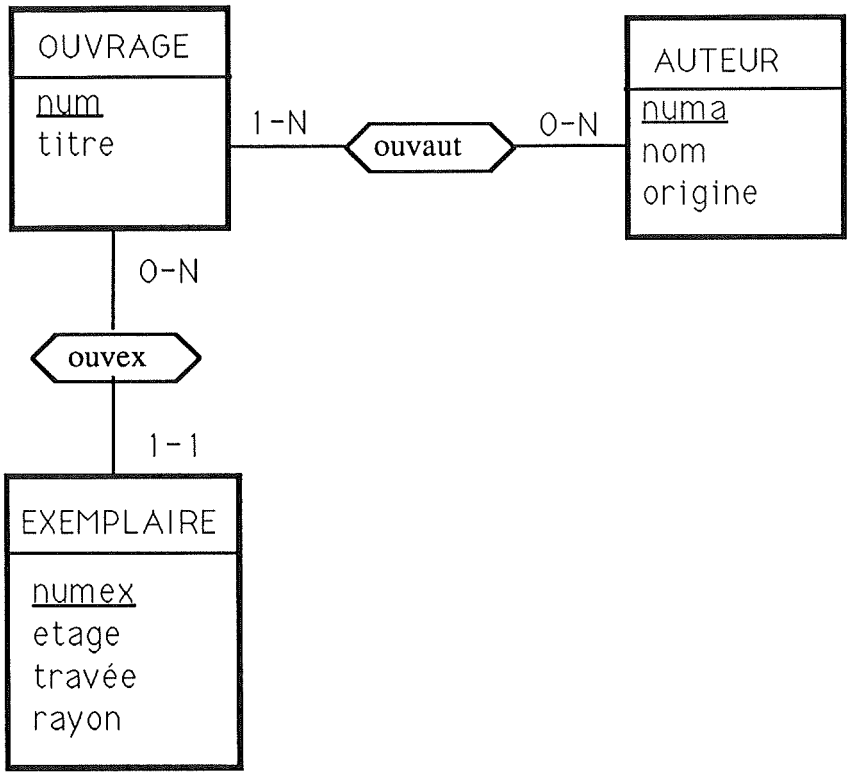


Fig 4-2 (a). exemple d'un schéma non-acceptable par l'atelier.

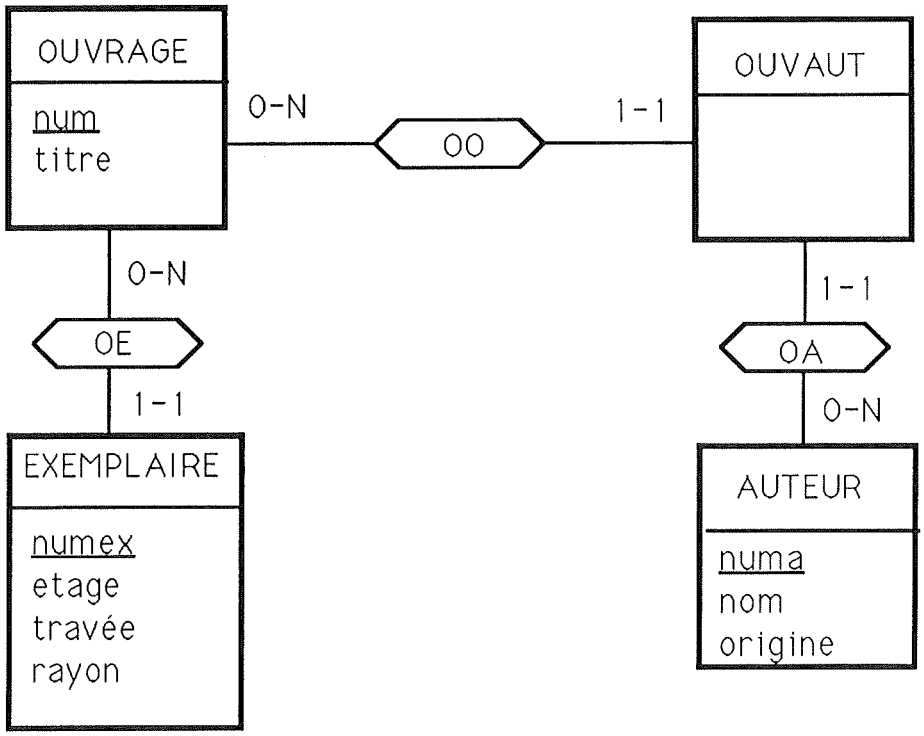


Fig 4-2 (b). Exemple d'un schéma acceptable par l'atelier.

La figure 4.2. montre un schéma acceptable par l'atelier. En effet dans ce schéma le type d'entité OUVAUT était une type d'association jouant deux rôles, tous les deux 0-N ou 1-N, et par conséquent non compatible avec le modèle de l'atelier et transformé par l'utilisateur lui même.

4.2 : Le modèle dBase

Nous avons choisi dBase-III comme SGBD de développement d'application de ce travail. Nous allons présenter une introduction sur dbase-III, sans insister sur toutes les versions de dBase actuellement disponibles.

dBase-III comme un successeur de dBase-II et les successeus de dBase-III (dBASEIII+ et dBaseIV), héritent leur popularité de dBase-II et leurs fiabilité de fonctionnement et performance actuelle de nouvelle adaption sur les PC au début des années 80 (apparition des PC d'IBM en1981).

Le logiciel dBaseIII d'Ashton-Tate est disponible sous MS-DOS et PC-DOS sur les ordinateurs personnels PC et PC/AT d'IBM, ainsi que sur leurs compatibles. Il exige un minimum de 256 K de mémoire centrale et peut fonctionner sur une machine qui n'est dotée que d'unités à disquettes.

4.2.1. Les concepts dBase.

Il ne faut pas confondre une "base de données" de dBase qui est mal nommée, avec notre définition plus générale d'une base de données. Data base ou base de données en dBase est simplement un fichier de données.

En ce qui concerne ce fichier de données dBase, il est constitué d'enregistrement (records), qui sont décomposés en champs (field). Un enregistrement est identifié par un numéro(record#)qui correspond à sa position dans le fichier. Un champ est de type numérique, caractère, logique, date ou mémo.
Un champ du type caractère est caractérisé par sa longueur maximale. Un champ logique peut prendre les valeurs "vrai" (représenté par .T.) et "faux" (.F.) et occupe 1 caractère. Un champ du type date contient une valeur qui peut faire l'objet de calculs chronologiques. Enfin, une valeur d'un champ du type mémo est un texte libre de longueur quelconque. Sa manipulation n'obéit cependant pas aux mêmes règles que les autres types de champ. En particulier, il n'est pas aisé d'y accéder par programme. Il est possible de construire un index sur un champ, mais aussi sur une

expression quelconque (la concaténation de champs ou un extrait d'un champ par exemple).

Outre les fonctions de gestion de fichiers et d'interpréteur, dbaseIII possède un gestionnaire d'écran, un générateur d'écran, un générateur de rapport, un générateur d'étiquettes et un éditeur de textes.

dBase III offre à l'utilisateur un langage de commande interactif qui permet notamment de définir un fichier, de modifier sa description, d'ajouter, supprimer et modifier des enregistrements, d'extraire des données d'un ou plusieurs fichiers et de consulter des enregistrements sélectionnés.

Il offre également des structures algorithmiques qui, ajoutées au langage de commande, permettent de rédiger des procédures et des programmes complexes. Ces derniers sont interprétés, ce qui n'assure pas des performances exeptionnelles en traitement de calcul (ce n'est d'ailleurs pas le but de ce logiciel), mais ce qui permet par contre d'offrir des fonctions particulièrement intréssantes(les macros par exemple) et une mise au point rapide des programmes.

4.2.3 : Exemple de représentation en dBase d'un schéma.

dBase-III étant un SGBD relationnel, la transformation d'un schéma conceptuel en schémas conformes à ce SGBD suit une série de règles bien définies. Ces transformations seront développées au chapitre suivant. Ici tout en ignorant ces règles de transformation, on présente un exemple de la représentation d'un schéma en dBASE.

Les descriptions des fichiers du schéma classique CLIENT-COMMANDE en dBase se présentent comme suit:

Description du fichier CLIENT.DBF

Field	field name	type	width	dec
1	NCLI	character	4	
2	NOM	character	12	
3	ADRESSE	character	20	
4	LOCALITE	character	12	
5	CAT	character	2	
6	COMPTS	numeric	9	2

Description du fichier COMMANDE.DBF

Field	field name	type	width	dec
1	NCOM	character	4	
2	NCLIE	character	4	
3	DATE	character	8	

Description du fichier LIGNECOM.DBF

Field	field name	type	width	dec
1	NCOME	character	4	
2	NPROE	character	5	
3	QCOM	character	4	

Description du fichier PRODUIT.DBF

Field	field name	type	width	dec
1	NPRO	character	5	
2	LIBBELE	character	20	
3	PRIX	numeric	5	
4	QSTOCK	numeric	6	

Dans ces descriptions, nous voyons des champs(Fields), dûs au processus de la transformation (Par exemple NCLIE dans le fichier COMMANDE). C'est la raison pour la quelle on complète ce schéma en dBase avec les contraintes d'intégrité suivantes:

NCLIE(:COMMAND) in NCLI(:CLIENT)
NUME(:LIGNE) in NUM(:COMMANDE)
NUMPE(:LIGNE) in NUMP(:PRODUIT)

N.B. La lettre **E** est ajoutée à la fin de chaque attribut qui apparaît dans un autre fichier (Par exemple l'attribut NCLI de CLIENT qui apparaît en COMMANDE sous le nom de NCLIE), pour le repérer comme un attribut (ou comme un champ de dBASE) **E**tranger.

4.4. Représentation d'un schéma dans la base de spécifications.

L'atelier est essentiellement basé sur une base de données qui contient les spécifications des schémas de bases de données et qui est appelée ***base de spécifications.***

La figure 4.3. donne le schéma E/A de la base de spécifications. Dans ce schéma les types d'entité et les relations sont définies comme suit :

BASE: Pour stocker les noms des bases de données.
PROGRAMME: Pour stocker les noms de programmes.
ENTITE: Contient touts les noms de types d'entité de différentes bases de données qui sont dans la bae de spécifications.
ASSOCIATION: Contient les noms de types d'association.
ATTRIBUT: Contient les noms et les états (identifiant ou non) des attibuts.

BP: Liens entre une base de données et ses programmes.
BE: Liens entre une base de données et ses types d'entité.
ORIGINE: Lien entre un type d'entité et ses types d'association, associé par ORIGINE.
CIBLE: Lien entre un type d'entité et ses types d'association associé par CIBLE.
EA: Lien entre un type d'entité et ses attributs.

BAT: Liens entre une base de données et ses fichiers dBaes traduits.
AAT: Lien entre un fichier dBase et ses champs(Field).

Ces descriptions seront développer au chapitre 6.
La base de spécifications contient donc toutes les spécifications de bases de données introduites dans l'atelier, ainsi que celles de schémas traduits. Comme toutes les autres bases de données, grâce aux requêtes d'application, on peut parcourir cette base de spécifications afin de réaliser les différentes fonctions de l'atelier. Par exemple à la demande de l'utilisateur d'avoir un rapport sur une base de données, il suffit d'accéder sur cette base par le nom de la base de données, puis la parcourir pour obtenir les types d'entité, attributs, types d'associations .. et donner le rapport désiré. La figure 4.4. donne un exemple d'occurrence d'une base de données COMMANDE-CLIENT dans la base de spécifications.

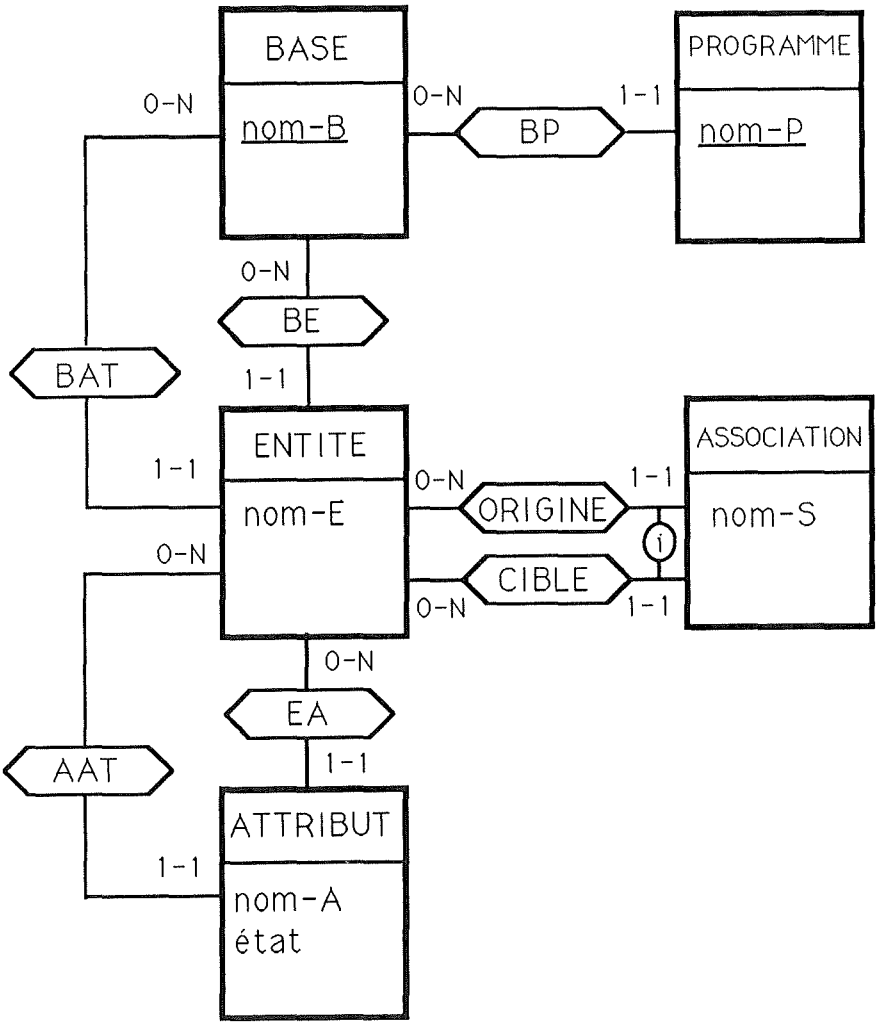


Fig 4-3. Schéma E/A de la base des spécifications

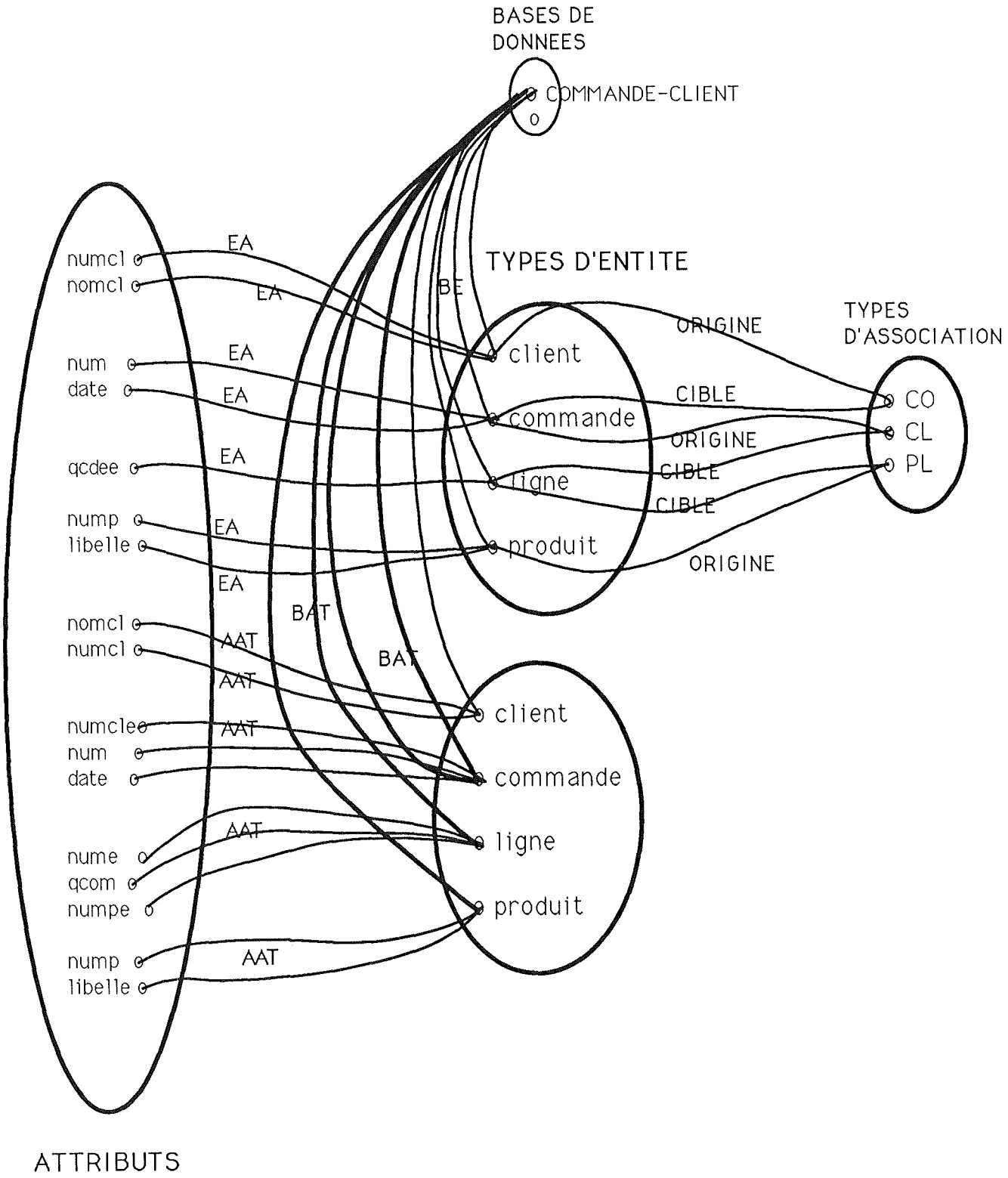


Fig 4.4. Schéma d'occurrence du schéma COMMANDE-CLIENT
et sa traduction sur BD spécifications.

Comme on le constate dans ce schéma , si l'attribut NUMCL de type d'entité CLIENT était nommé NUM, on aurait deux attributs 'NUM' dans le type d'article COMMANDE du schéma traduit, ce qui poserait un problème au moment de la validation du compilateur pour valider le schéma au sein d'un programme donné, et aussi pour l'identification des contraintes d'intégrité etc. C'est la raison pour la quelle l'atelier impose à l'utilisateur une règle d'unicité de nom des attributs au sein d'une base de données.

CHAPITRE 5 : TRADUCTION DE SCHEMAS.

5.1 : Objectifs.

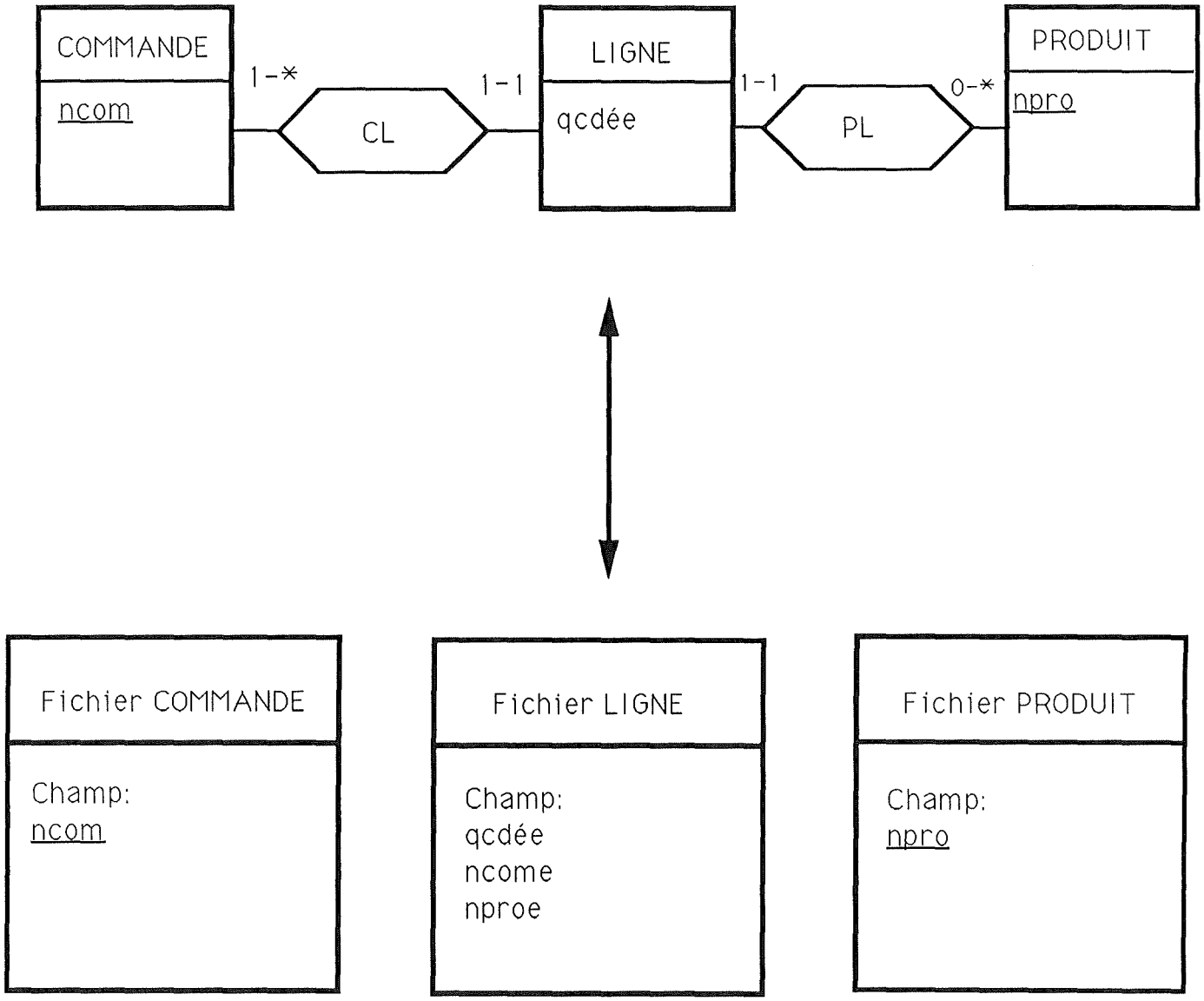
La réalisation correcte de transformation d'une structure de données, constitue une partie importante d'une démarche de conception de BD. Intuitivement, une réalisation correcte signifie la garantie de l'équivalence sémantique des expressions, c'est-à-dire la conservation de la spécification d'origine. [HAINAUT,87] explique ces transformations en détails dans une démarche de conception de bases de données.

Dans ce chapitre outre l'explication de la transformation dans notre cas, on verra les règles de transformation pour lesquelles le générateur de l'atelier se base pour la traduction d'un schéma E/A.

5.2. Les transformations.

Le système de gestion de bases de données dBaseIII étant un SGBD relationnel, ne comprend pas de types d'association. En effet en dBase - comme dans les autres SGBD relationnels d'ailleurs - la relation entre deux types d'entité se réalise en possédant des attributs communs dans ces types d'entité. Alors l'élimination de types d'association de schémas E/A est indispensable. En même temps, cette élimination de types d'association doit assurer l'équivalence sémantique du schéma E/A.

La fig 5.1. présente un schéma E/A et la transformation de l'élimination de types d'association. En éliminant chaque type d'association, l'identifiant de type d'entité origine apparaît aussi sur le type d'entité cible. Par exemple NCOM de COMMANDE qui apparaît aussi comme NCOME de type d'entité LIGNE et qui provoque une contrainte d'intégrité :
NCOME(:LIGNE) in NCOM(:COMMANDE)



CONTRAINTES D'INTEGRITE:
ncome(:LIGNE) in ncom(:COMMANDE)
nproe(:PRODUIT) in npro(:PRODUIT)

Fig 5.1. Exemple d'élimination de types d'association.

5.3 : règles de transformation de l'atelier.

Comme on l'a indiqué plus haut, un générateur de traduction de l'atelier transformera le schéma E/A -introduit par l'utilisateur et vérifié par la fonction de validation- en schéma conforme à dBase.

Ce générateur se base sur les règles suivantes qui en effet sont les règles d'élimination de types d'association. Avant de présenter ces règles, il faut rappeler que dans un schéma E/A un type d'entité peut se présenter une ou plusieurs fois comme un type d'entité cible ou un type d'entité origine.

règle 1 : Chaque type d'entité du schéma Entité/ association devient un fichier dBase.

règle 2 : Tous les attributs de ce type d'entité deviennent les champs (Fields) du fichier dBase.

règle 3 : Pour chaque type d'entité on vérifie s'il s'agit d'un type d'entité cible, si c'est le cas alors:

3.1 : On ajoute l'identifiant du type d'entité d'origine comme un champ du fichier traduit.

3.2 : On ajoute au schéma dBase la contrainte d'intégrité correspondante.

... et les sous-règles 3.1 et 3.2 se répètent autant fois qu'on constate que ce type d'entité est un type d'entité cible.

CHAPITRE 6 : DESCRIPTION ET ANALYSE DES FONCTIONS DE L'ATELIER.

6.1 : Introduction.

Dans ce chapitre on va voir les descriptions des fonctions et des composantes de l'atelier. La représentation d'un schéma sera aussi discutée. On présentera les interfaces de l'atelier avec l'utilisateur pour une utilisation plus commode de l'atelier, et enfin une analyse fonctionnelle classique du schéma conceptuel de la base des spécifications nous permettra une future réalisation plus claire de l'atelier.

6.2. Les fonctions de l'atelier:

Les principales fonctions de l'atelier sont: la gestion de base de spécifications, la transformation de la structure du schéma, la génération des textes du langage dBase et la production des rapports sur les schémas.

Gestion de la base de spécifications:

Cette fonction sert à introduire les descriptions d'une base de données exprimées en modèle E/A dans la base des spécifications. Un interface permet à l'utilisateur d'introduire ces descriptions. Il y a 12 sous-fonctions pour gérer la base: 3 opérations soit positionner, créer ou supprimer un élément parmi 4 éléments (soit une base de données, un type d'entité, un attribut ou un type d'association). Ces 12 sous-fonctions permettent à l'utilisateur de créer ou mettre à jour une base de données ou les supprimer.

Transformation de la structure du schéma:

A l'aide de cette fonction un schéma conservé dans la base des spécifications peut être transformé en un schéma conforme à dBase. La condition est bien sûr de respecter les contraintes du modèle E/A de l'atelier expliqué au chapitre 4.

Transformation des textes:

Un texte rédigé en LDA et qui est un traitement opérationnel sur une base de données existante sur la base des spécifications, peut être transmis en texte du langage dBase.

Productions des rapports:

Pour un schéma donné, un rapport peut être produit à la demande de l'utilisateur.

La figure 6.1. présente les différentes fonctions de l'atelier.

6.3. Présentation d'un schéma:

Autre les contraintes sur le modèle Entité/Association acceptables par l'atelier, les différents éléments d'un schéma seront présentés comme suit:

Attibut: Chaque attribut d'un type d'entité est présenté par:
- son nom. (qui est unique parmi des attributs d'un type d'entité)
-son état. (identifiant ou non)

Type d'entité: Chaque type d'entité est présenté par:
-son nom. (qui est unique dans une base de données)

Type d'association: Chaque type d'association est présenté par:
-son nom. (qui est unique dans une base de données)
-nom du type d'entité origine.
-nom du type d'entité cible.

Base de données: Chaque base de données est présentée par:
son nom. (qui est unique parmi toutes les bases de données de la base des spécifications)

NB: L'unicité du nom d'un type d'entité et / ou un type d'association au sein d'une base de données ne dit pas nécessairement qu'il est uniqe au sein de toutes les bases de données existantes sur la base de spécifications.

6.4. Interface avec l'utilisateur

La dialogue entre l'utilisateur et l'atelier permet à l'utilisateur de saisir des spécifications ou de demander un rapport ou encore des autres fonctions de l'atelier actuellement disponibles. La transmission des commandes se fait par l'intermédiaire du clavier et elles sont visibles sur l'écran du terminal.

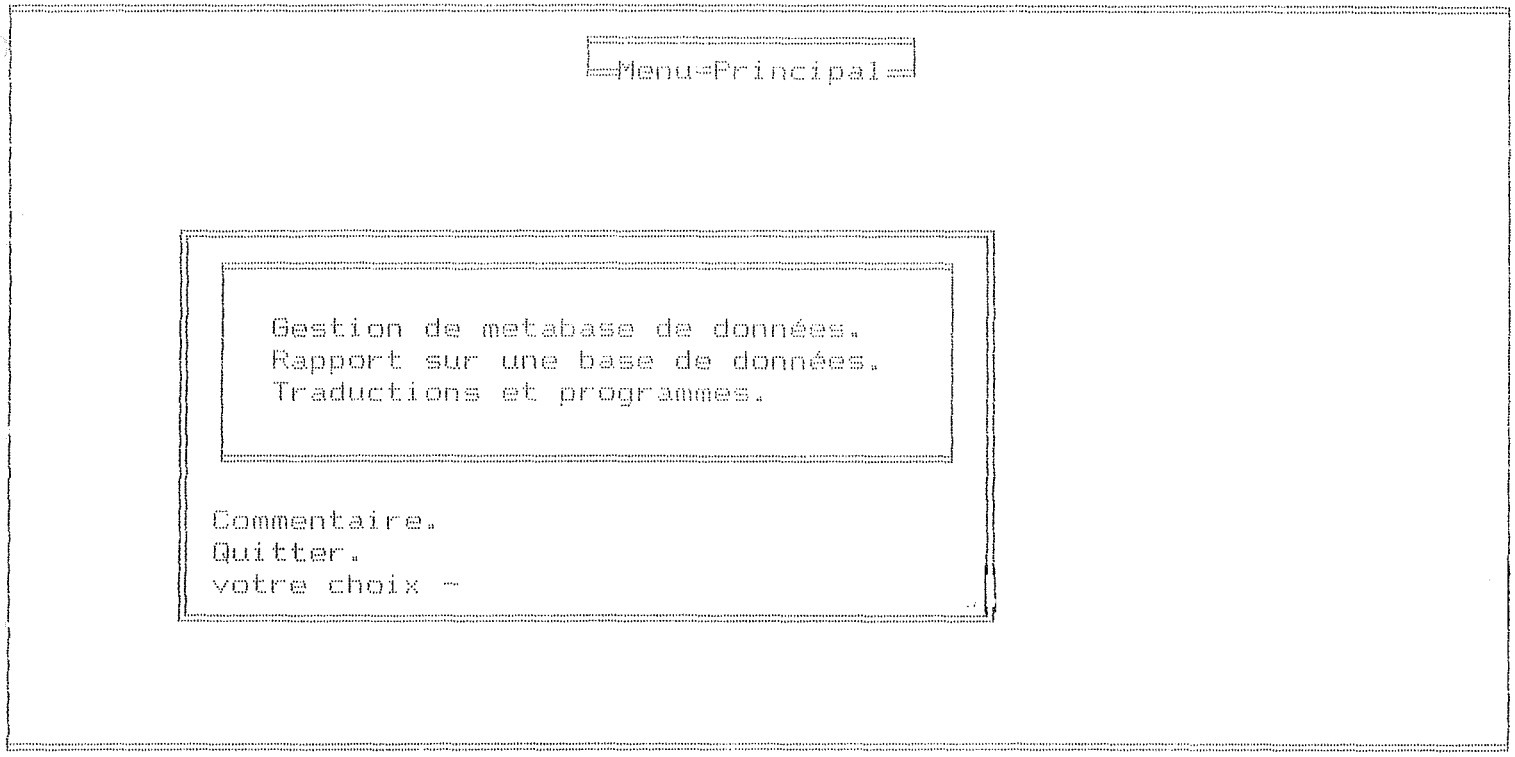
L'utilisateur introduit deux types d'informations: les informations concernant le choix d'une fonction sur les menus de l'atelier et les données et les descriptions d'un schéma à conserver. Ces informations seront introduites dans les *champs* des menus. Il est utile de rappeler qu'il existe deux types de champ : champ simple et champ de texte. Un champ simple contient une valeur de taille limitée, ne passant pas une ligne de la fenêtre. Un tel champ sera du type numérique ou caractère. Les champs de texte contiennent un texte de lonugeur quelconque. Dans cet atelier on utilise des champs simples.

Différents écrans de l'atelier sont:

6.4.1. Choix d'une fonction.

Nom de l'écran: "Menu Principal"

Format de l'écran:



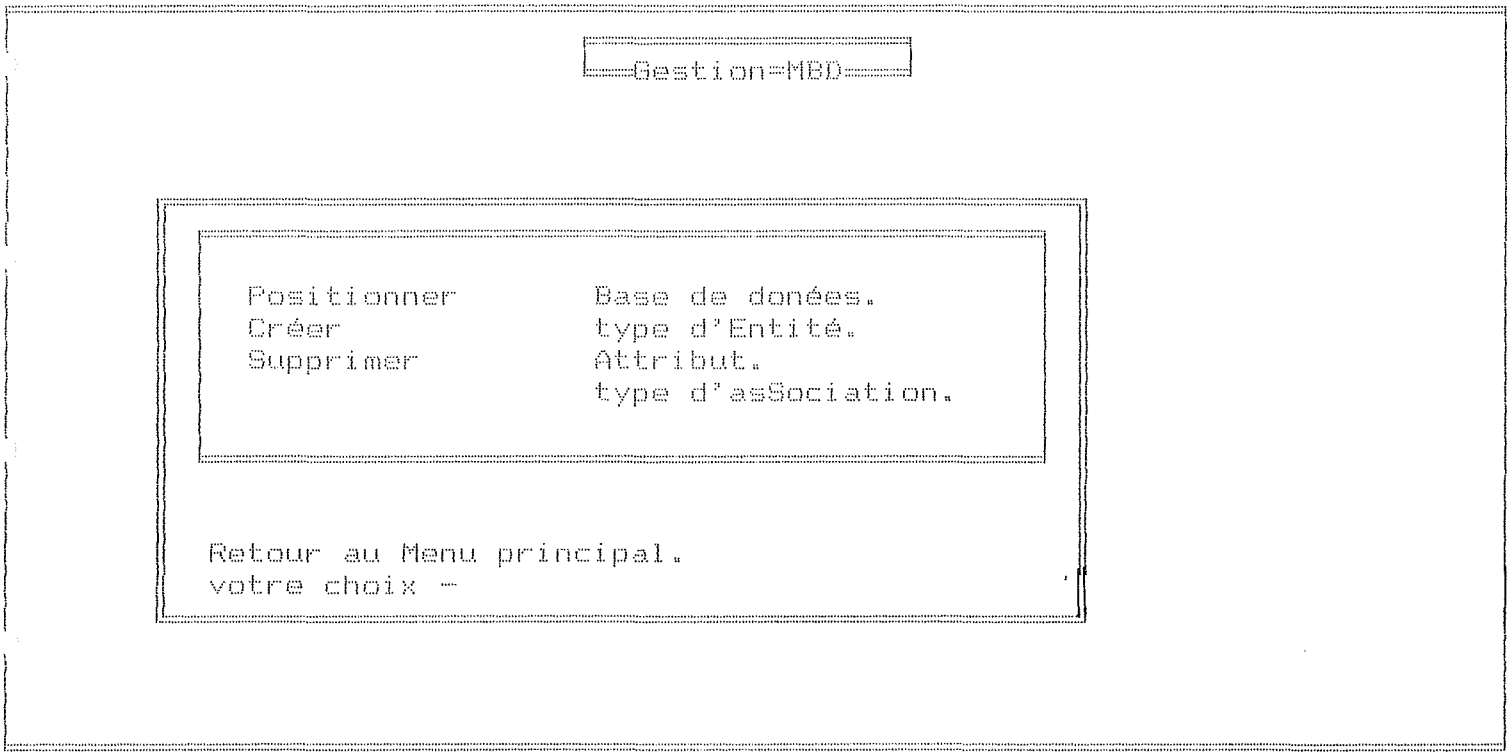
Commentaires:

L'objectif de ce menu principal est de choisir une fonction de l'atelier c'est à dire:
- Gestion de méta base de données. (base des spécifications).
- Fonction de production d'un rapport.
- Fonction de traduction d'un schéma ou traduction d'un texte de traitement ou introduire un programme LDA.
-Avoir les commentaires sur les fonctionnements de l'atelier.

6.4.2. Gestion de base des spécifications.

Nom de l'écran: "Gestion MBD"

Format de l'écran:



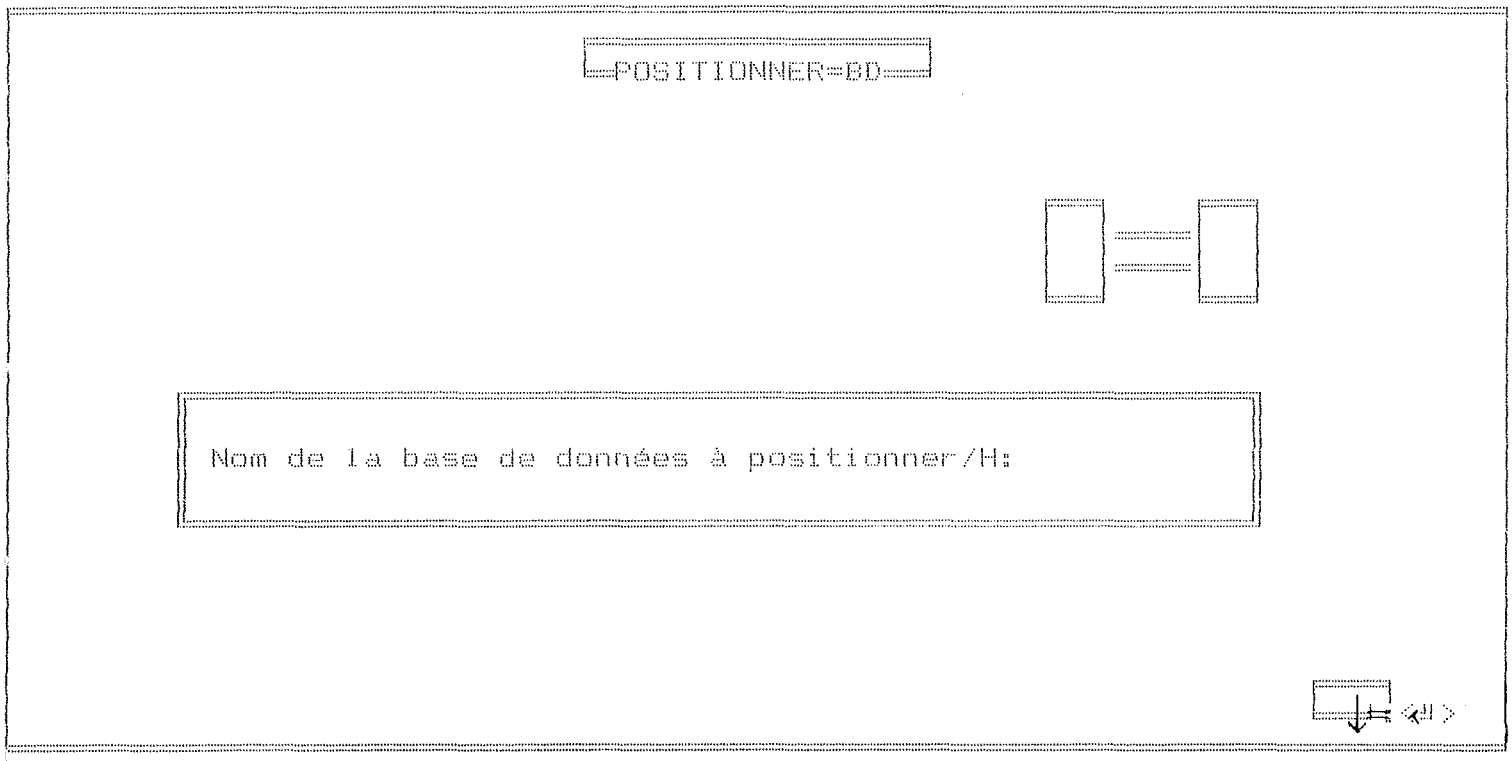
Commentaires:

On aura donc 12 choix disponibles: les écrans de 6.4.2.1. à 6.4.2.12 :

6.4.2.1.Positionner une base de donnée.

Nom de l'écran: "POSITIONNER BD"

Format de l'écran:



Commentaires:

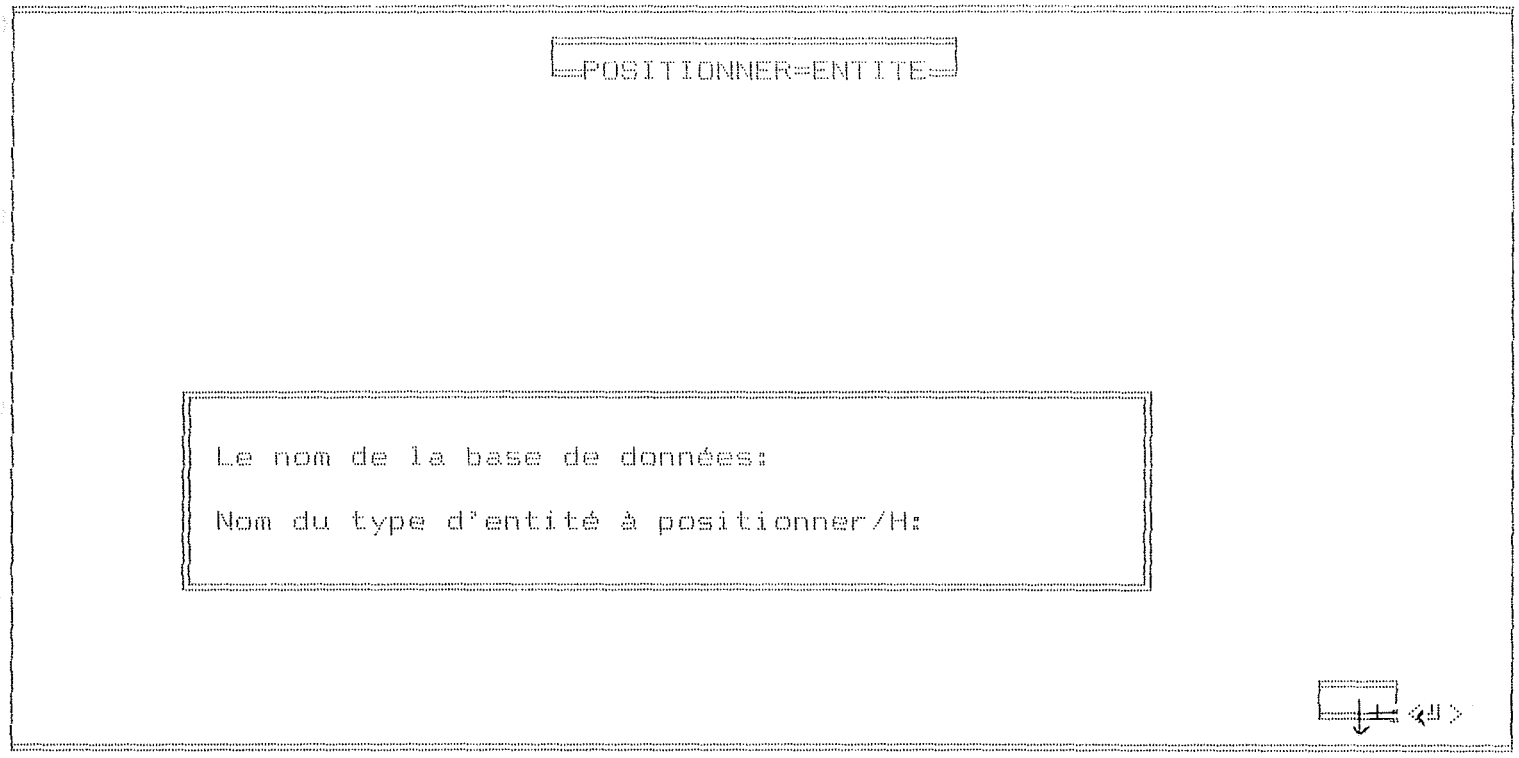
Une fois que l'utilisateur positionne le nom d'une base de données, pour effectuer toutes les autres fonctions et les sous-fonctions, il n'aura plus besoin d'introduire le nom de la base de données. En même temps l'atelier permet à l'utilisateur de manipuler sa base de données sans être obligé de positionner une base de données . C'est utile pour les cas de manipulations très courtes. Par exemple si l'utilisateur ne veut qu'ajouter un attribut dans une base de données dèjà introduite, l'atelier ne l'oblige pas à le faire, mais c'est clair que pour une opération plus longue, il est préfférable de positionner la base de données.

Pour positionner une base de données comme on voit sur le schéma de l'écran "Positionner BD", l'utilisateur doit introduire le nom de la base de données à positionner. Lorsque l'utilisateur hésite sur le nom exact de la base de données ou hésite sur l'orthographe exact du nom, la possibilité suivante est prévue:
Au lieu d'introduire le nom de la base de données,on introduit la lettre "H". Tous les noms des bases de données actuellement existantes dans la base des spécifications s'affichent, en défilant dans une fenêtre sur l'écran.

6.4.2.2. Positionner un type d'entité.

Nom de l'écran: "POSITIONNER ENTITE"

Format de l'écran:



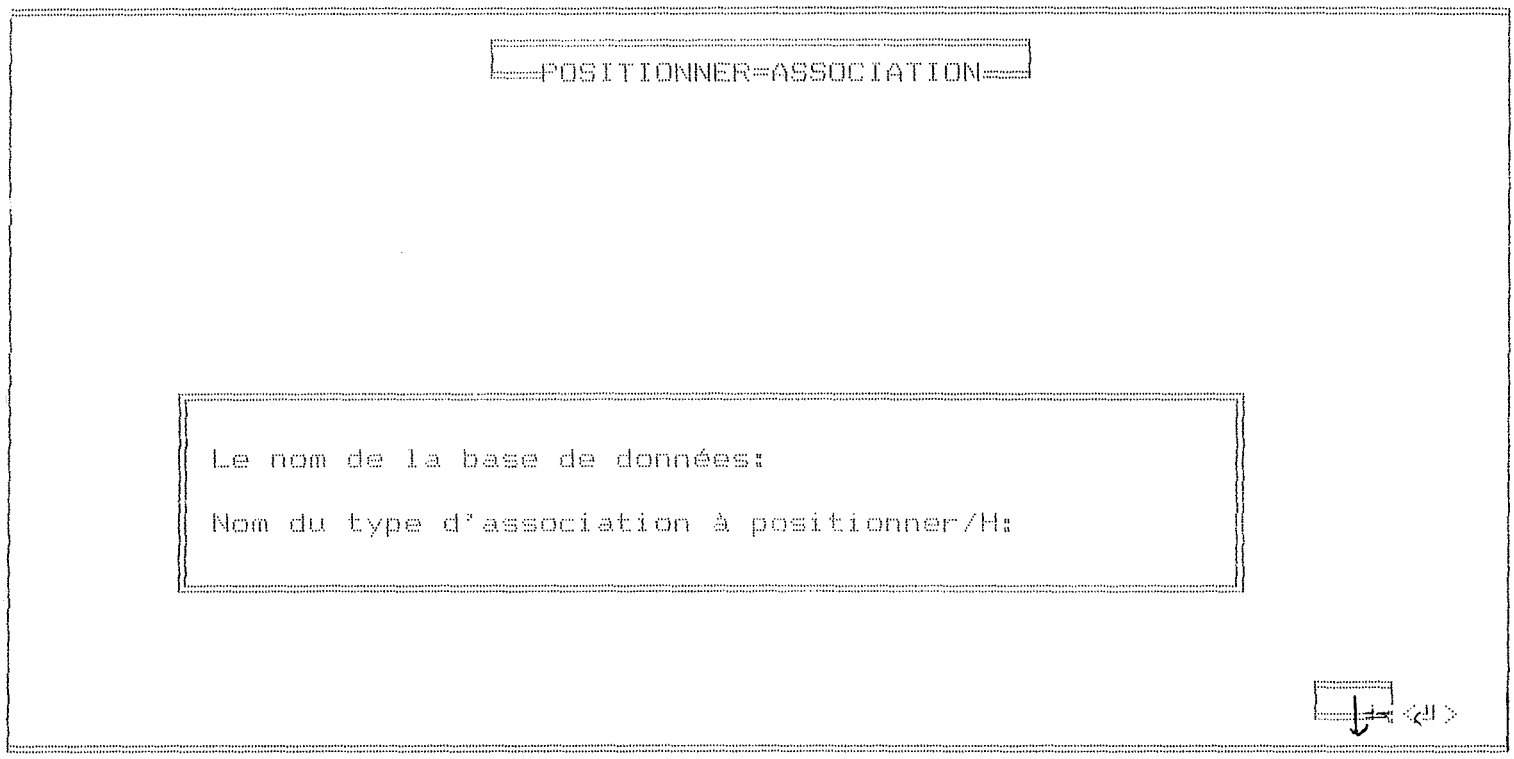
Commentaires:

Tout comme l'écran de positionnement d'une base de données, on peut positionner un type d'entité pour éviter de répéter de taper le nom du type d'entité quand on manipule un type d'entité. Pour les opérations et manipulations en cours, on n'a plus besoin de positionner un type d'entité.

6.4.2.3. Positionner un type d'association.

Nom de l'écran: "POSITIONNER ASSOCIATION"

Format de l'écran:

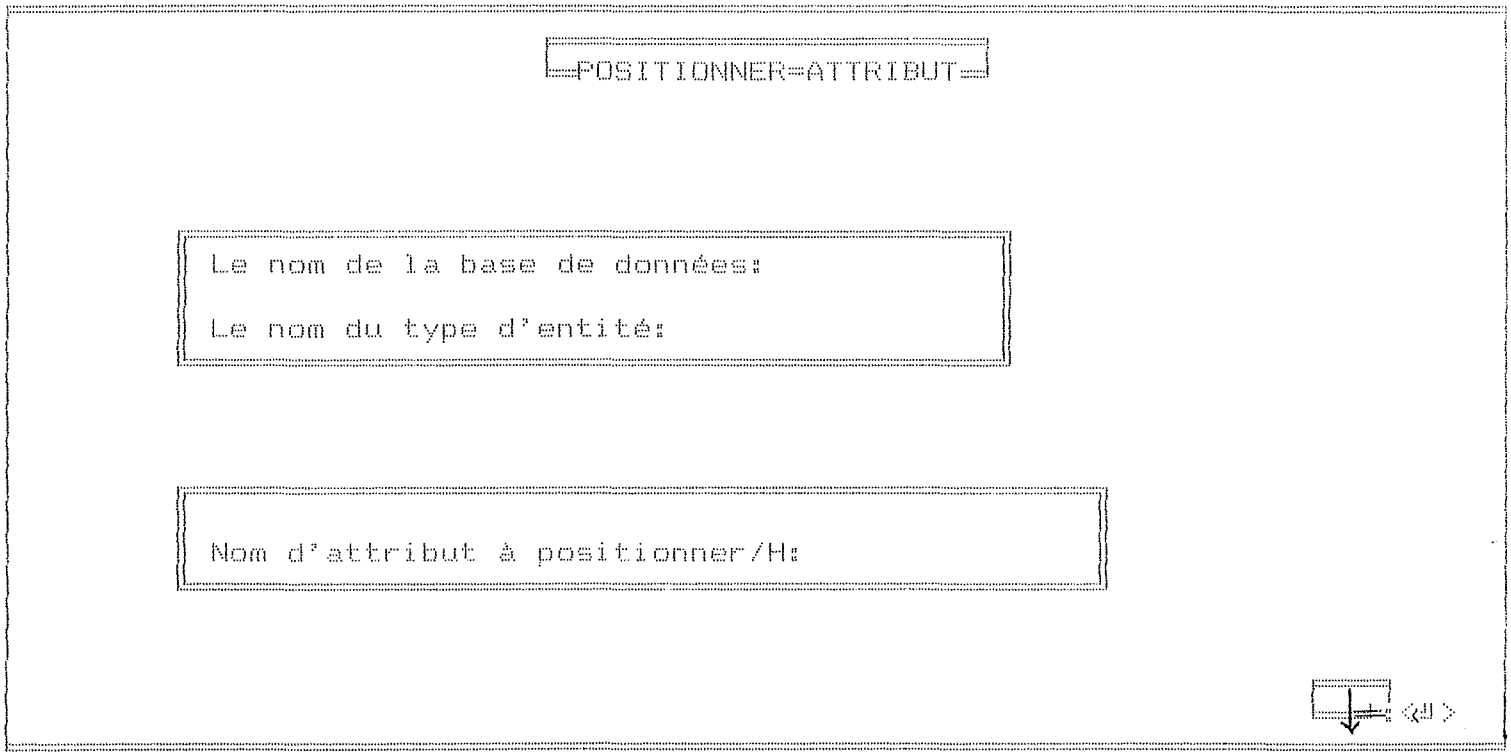


Commentaires: Voir positionner un type d'entité.

6.4.2.4. Positionner un Attribut.

Nom de l'écran: "POSITIONNER ATTRIBUT"

Format de l'écran:

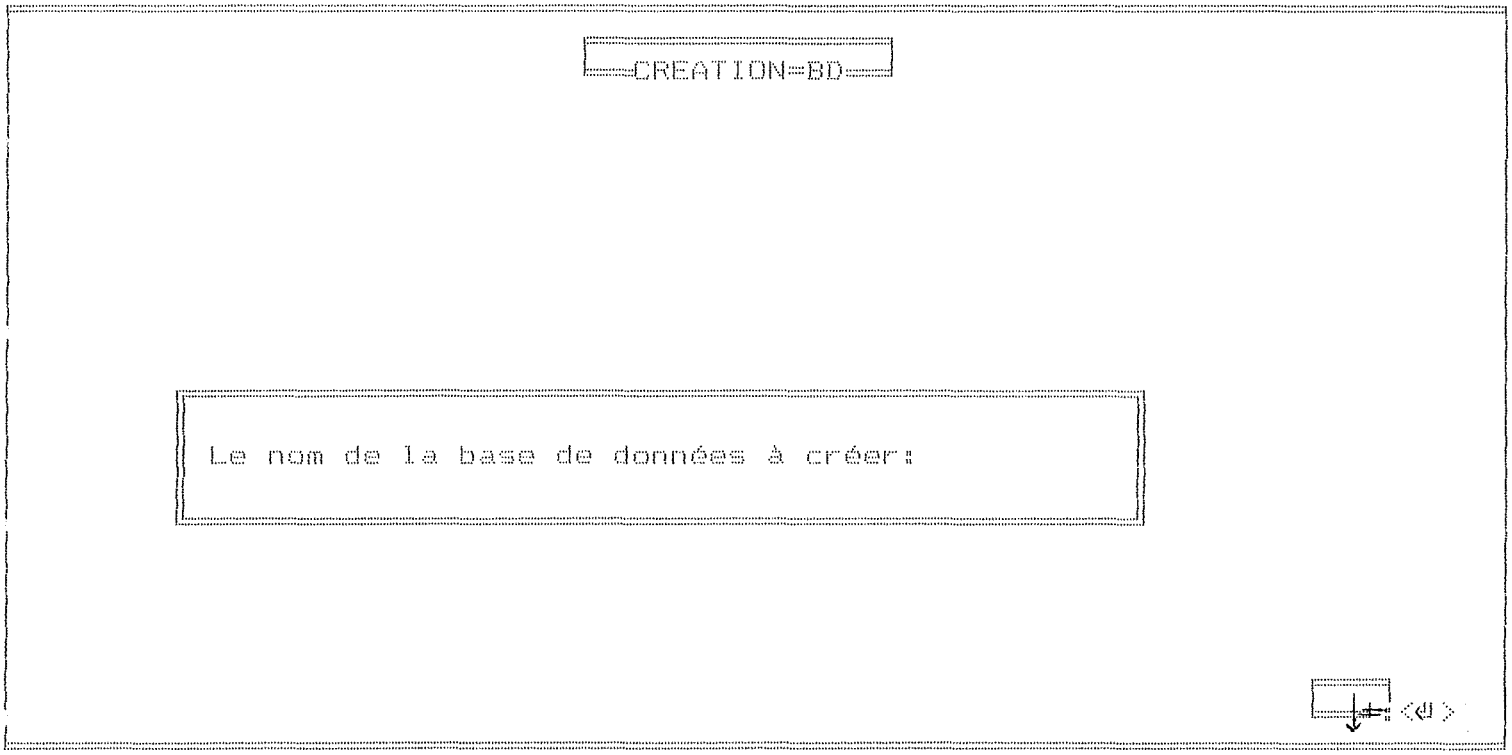


Commentaires: Un attribut va être positionné pour des traitements ultérieurs

6.4.2.5. Créer une base de données.

Nom de l'écran: "CREATION BD"

Format de l'écran:

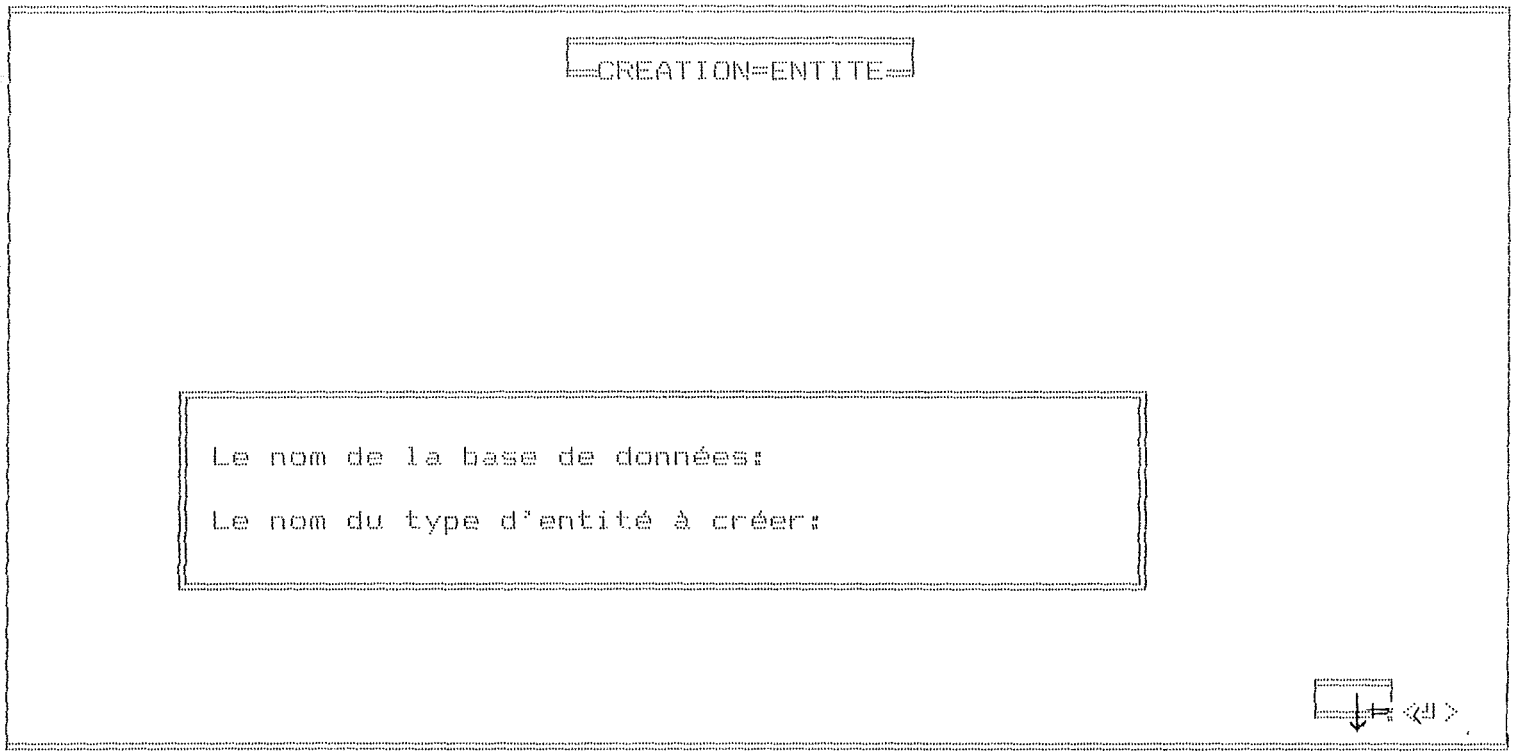


Commentaires: Une base de données est ajoutée dans la base des spécifications. Les écrans suivants (créer type d'entité et créer des attributs sont automatiques).

6.4.2.6. Créer un type d'entité.

Nom de l'écran: "CREATION ENTITE"

Format de l'écran:

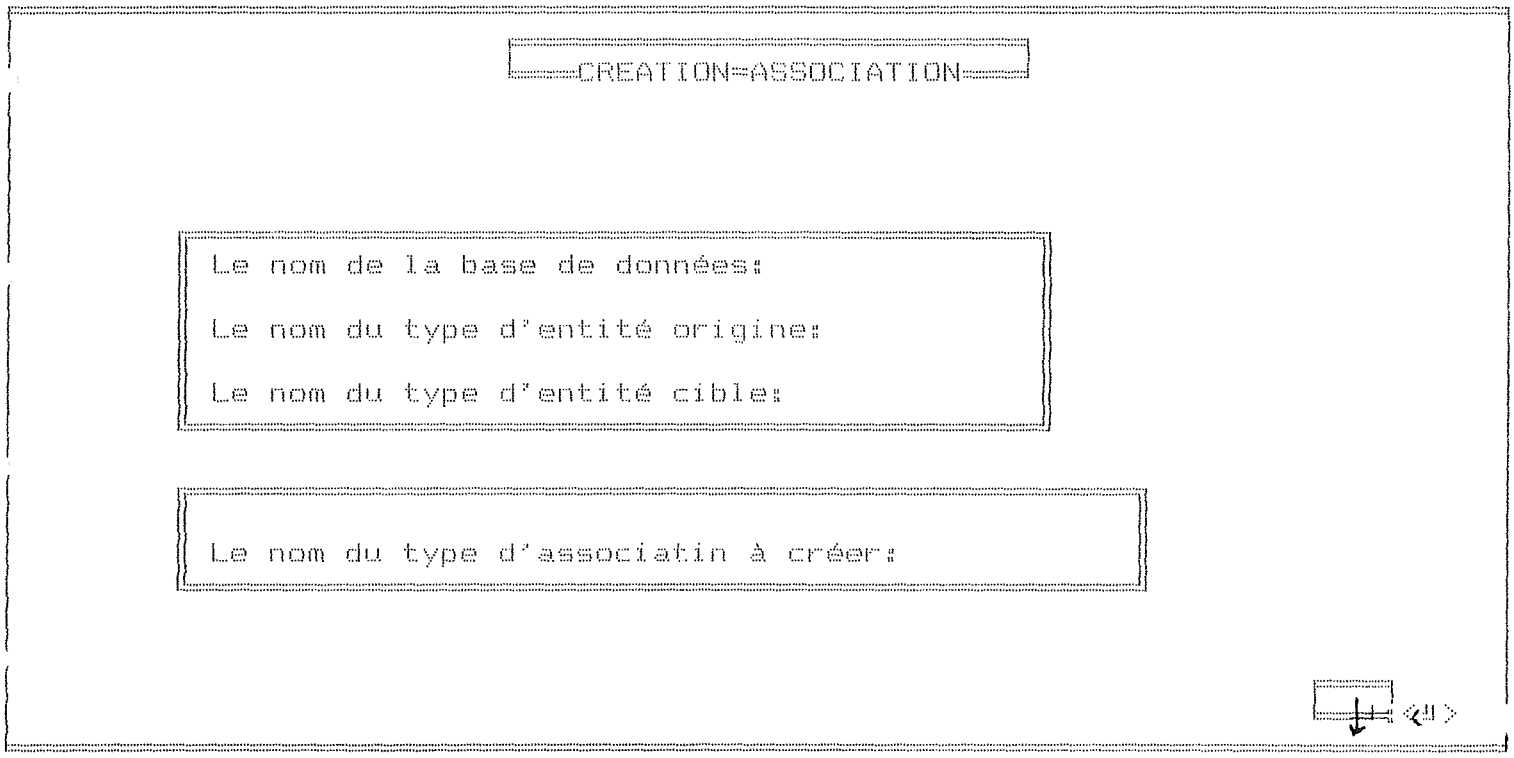


Commentaires: Une base de données possédera un type d'entité en plus. Les écrans de ses attributs sont automatiques.

6.4.2.7. Créer un type d'association.

Nom de l'écran: "CREATION ASSOCIATION"

Format de l'écran:

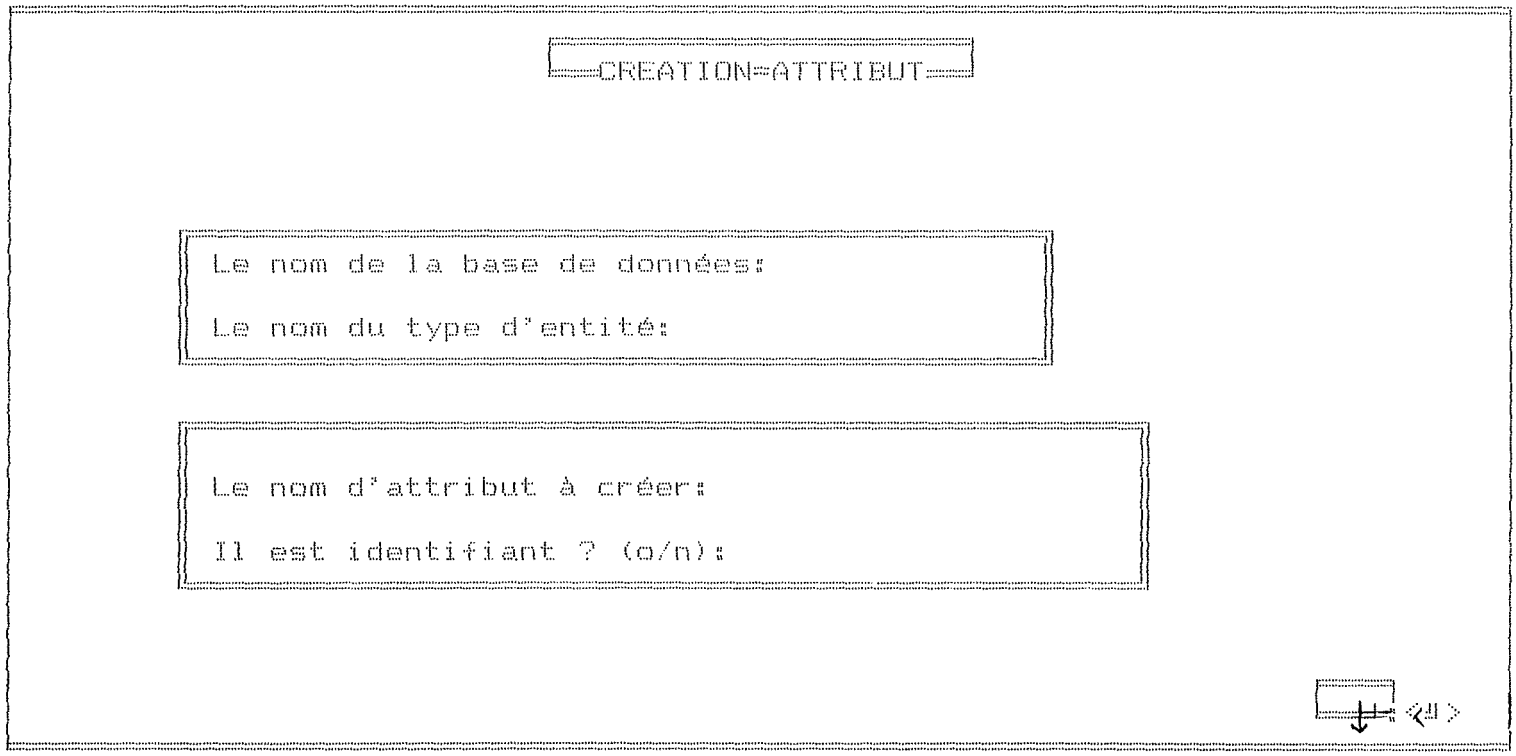


Commentaires: La relation entre un type d'entité origine et un type d'entité cible sera créée.

6.4.2.8. Créer un attribut.

Nom de l'écran: "CREATION ATTRIBUT"

Format de l'écran:

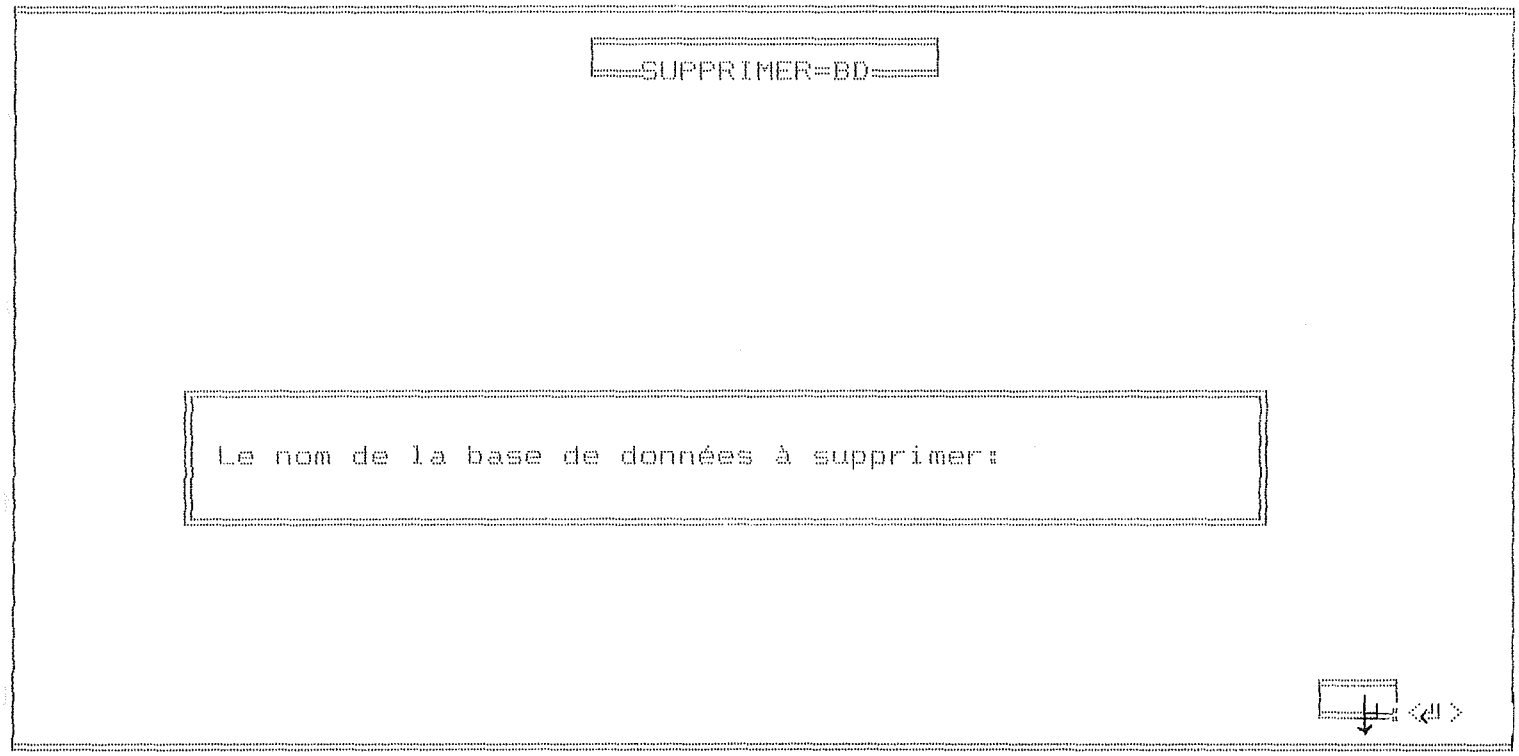


Commentaires: Un attribut et son état vont être créés, ainsi que sa relation avec son type d'entité.

6.4.2.9. Suppression d'une base de données.

Nom de l'écran: "SUPPRIMER BD"

Format de l'écran:



Commentaires: En introduisant le nom de la base de données à supprimer, tous les éléments de cette base de données vont être supprimés. Dans la base des spécifications on ne trouve plus les types d'entité, les attributs et les types d'association de cette base de données.

6.4.2.10. Supression d'un type d'entité.

Nom de l'écran: "SUPPRIMER ENTITE"

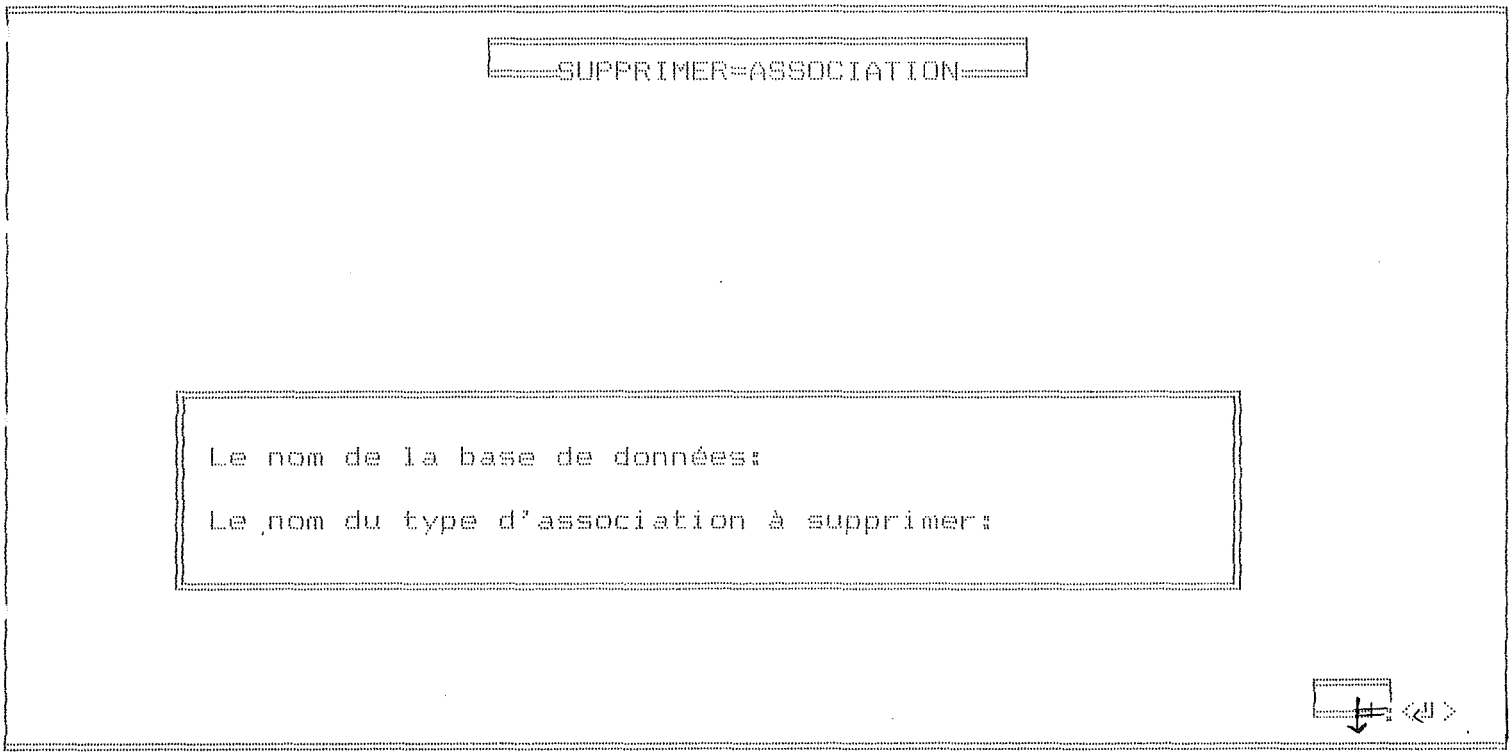
Format de l'écran:

Commentaires: Un type d'entité et ses attributs seront supprimés. Ainsi que tous les types d'association qui relient ce type d'entité aux autres types d'entité.

6.4.2.11. Suppression d'un type d'association.

Nom de l'écran: "SUPPRIMER ASSOCIATION"

Format de l'écran:

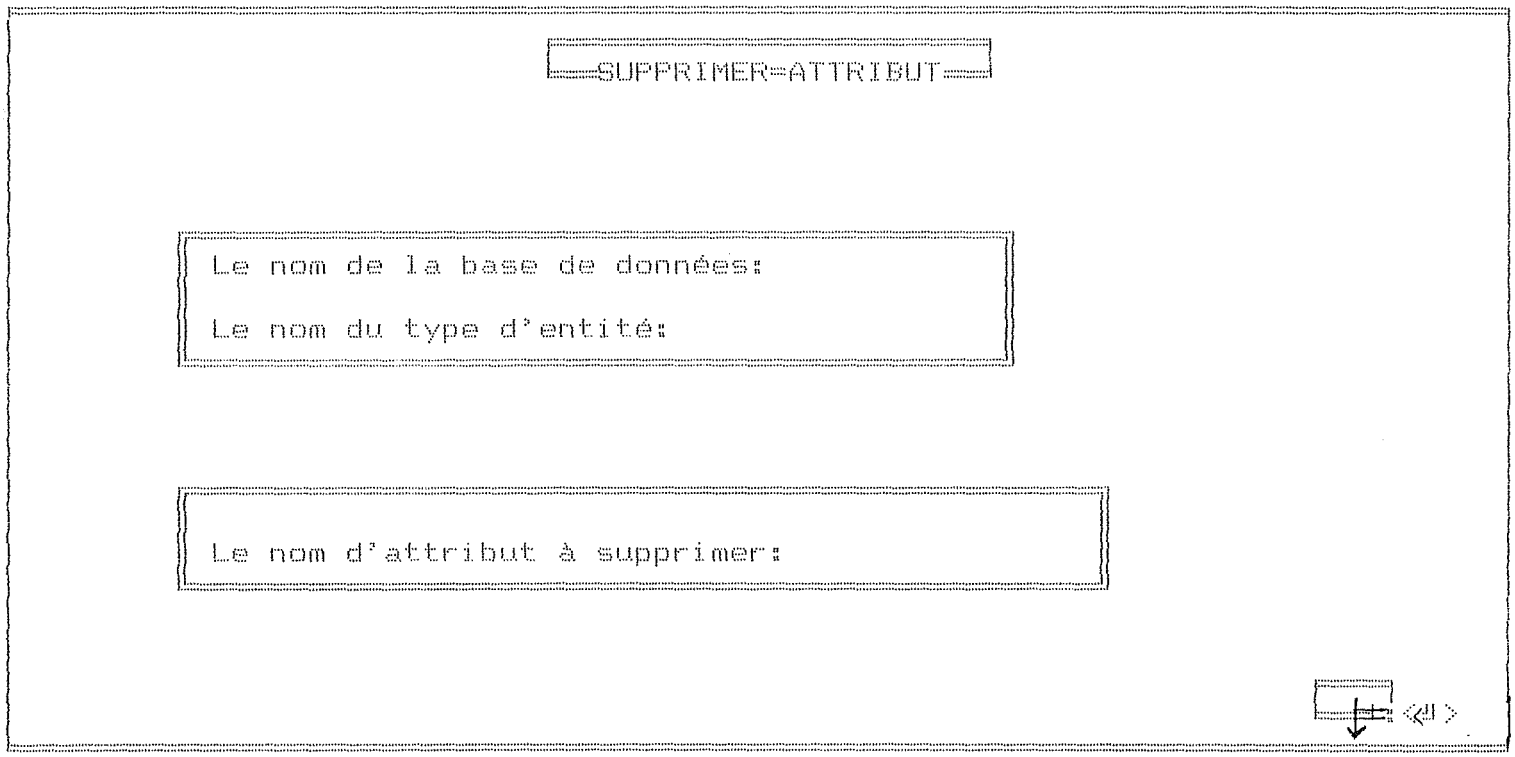


Commentaires: Supprime un type d'association qui relie deux types d'entité et ses liens avec les types d'entité origine et cible.

6.4.2.12. Suppression d'un attribut.

Nom de l'écran: "SUPPRIMER ATTRIBUT"

Format de l'écran:

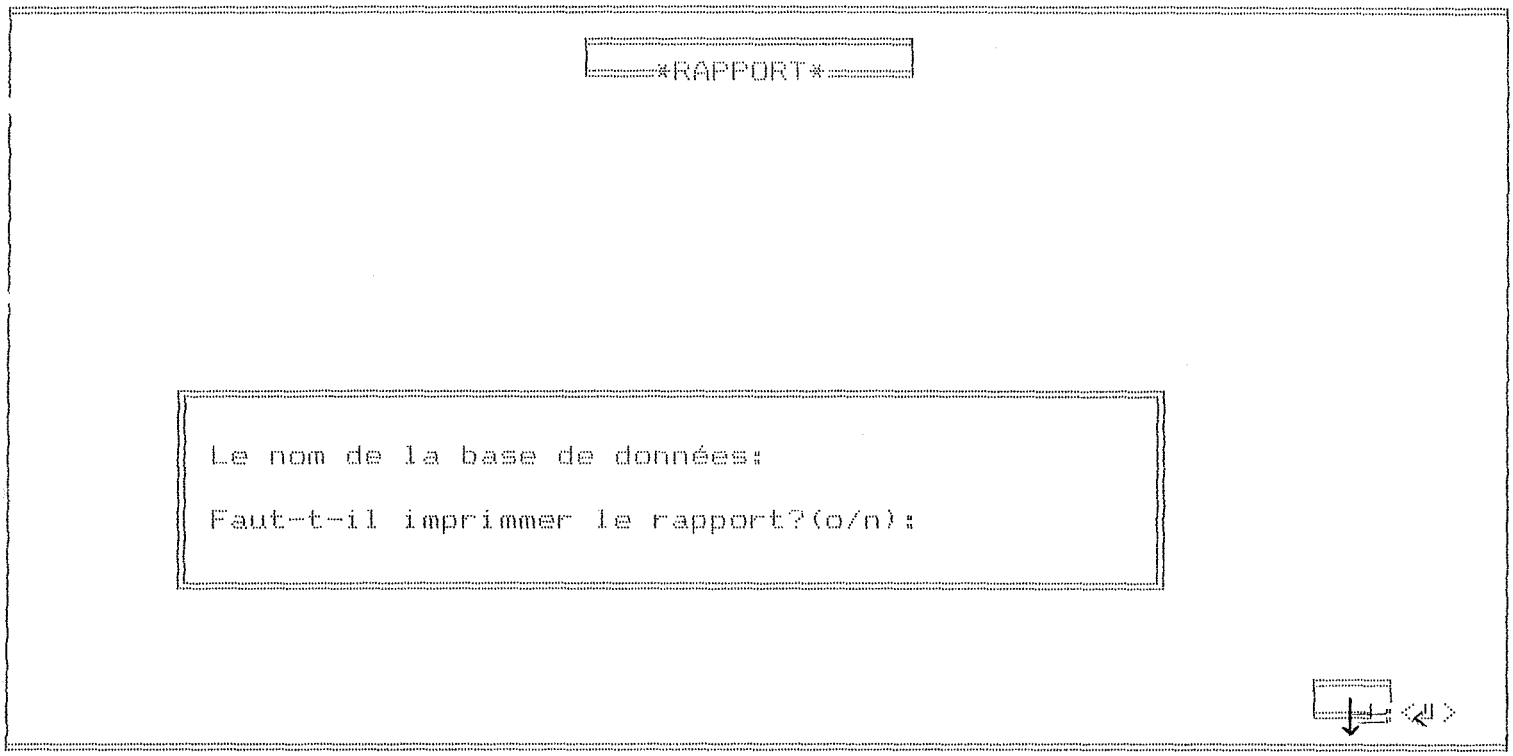


Commentaires: Supprime un attribut et le lien entre cet attribut et le type d'entité correspondant.

6.4.3. Rapport sur une base données.

Nom de l'écran: "RAPPORT"

Format de l'écran:

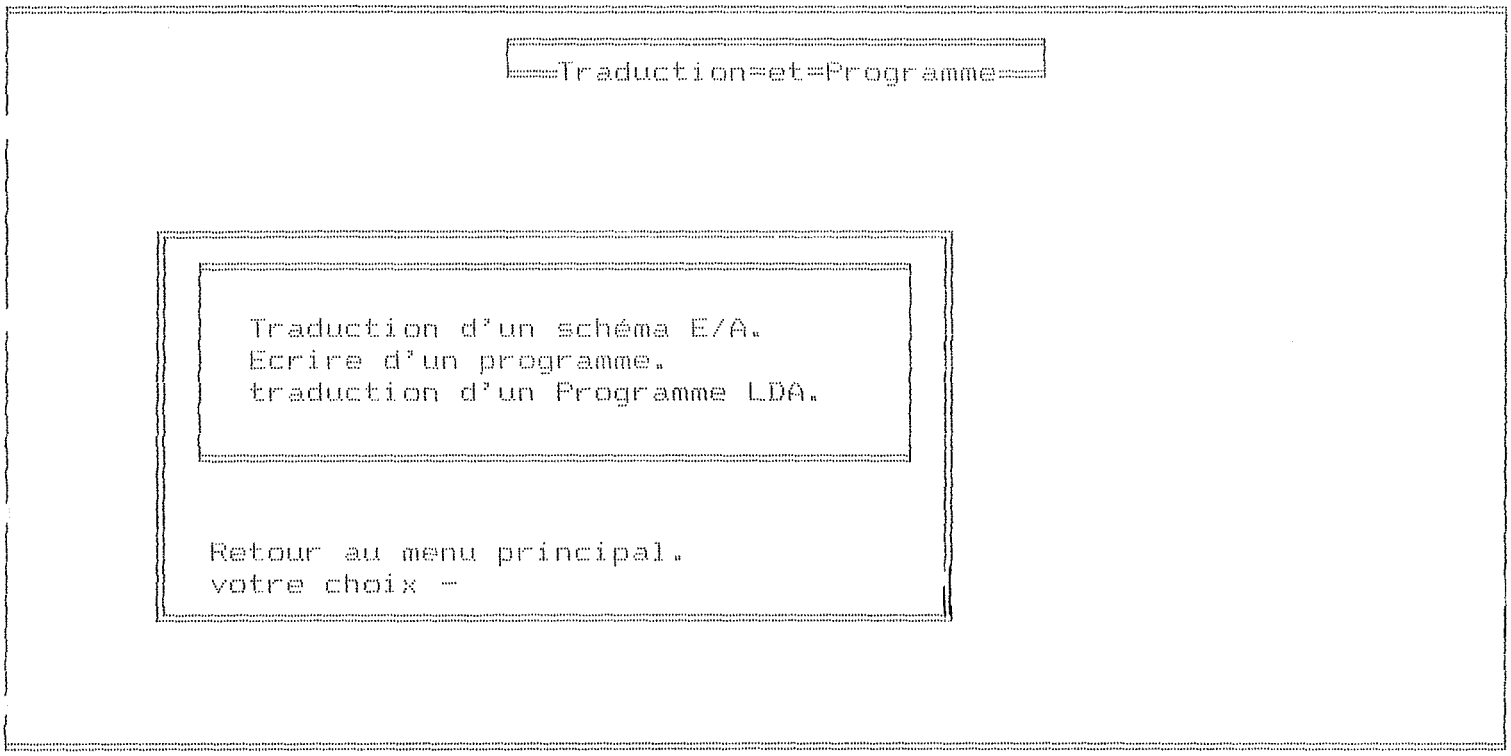


Commentaires: Affiche un rapport sur une base de données. il n'a aucun effet sur la base de spécifications.(Voir l'exemple du chapitre 8)

6.4.4.Traduction d'un schéma ou d'un programme LDA.

Nom de l'écran: "Traduction et Programme"

Format de l'écran:

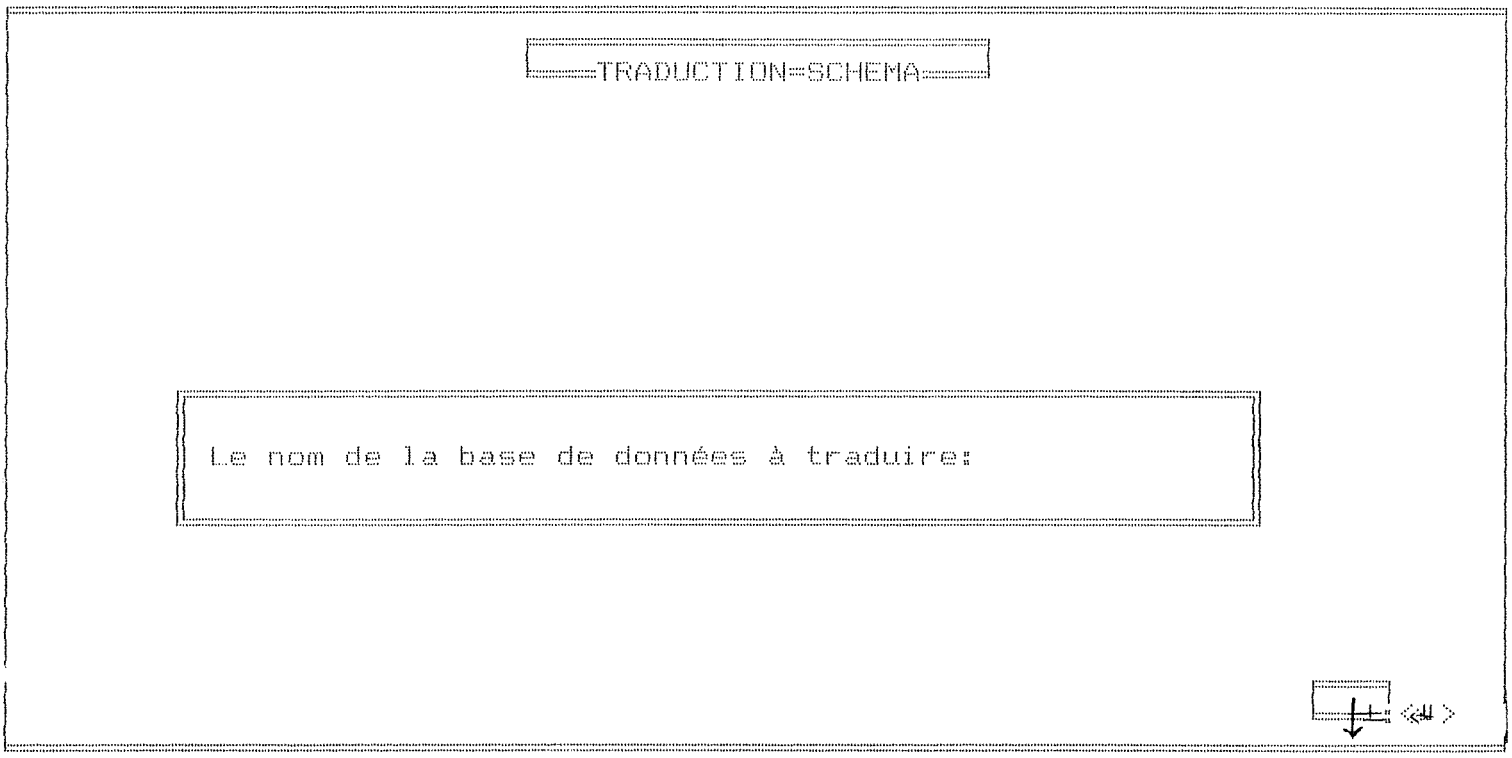


Commentaires: Exécute le processus de la traduction d'un schéma et affiche le résultat.

6.4.5. Traduction d'un schéma.

Nom de l'écran: "Traduction Schéma"

Format de l'écran:



Commentaire: Exécute le processeus de la traduction pour un schéma et affiche le résultat.

6.5. Analyse fonctionnelle de la base de spécifications.

Une réalisation future de l'atelier nécessite aussi une analyse fonctionnelle à partir du schéma conceptuel de la base de spécifications. (Schéma 4.3.)

6.5.1.Définition des types d'entité et des types d'association

1 :Type d'entité "BASE"

définition: L'ensemble des noms des bases de données introduites dans la base de spécifications.

identifiant: nom-B le nom de la base de données.

2 :Type d'entité "ENTITE"

définition: L'ensemble des noms des types d'entité de toutes les bases de données déjà introduites par l'utilisateur.
attribut: nom-E le nom du type d'entité.
identifiant: nom-E + le rôle "BD est ensemble". (le nom de chaque type d'entité est unique dans une base de données).

3 :Type d'entité "ATTRIBUT"

définition: L'ensemble de tous les attributs de tous les types d'entité.
attributs: nom-A le nom de l'attribut.
état l'état de l'attribut selon qu'il soit identifiant ou non.
identifiant: nom-A + rôle "entité et ensemble"

4 :Type d'entité "ASSOCIATION"

définition: l'ensemble des noms d'association des bases de données.
attribut: nom-S le nom de type d'association.
identifiant: nom-S + le rôle ent-or
nom-S + le rôle ent-cible.

5 :Type d'entité "PROGRAMME"

définition: contient les noms des programmes introduits dans la base.
attribut: nom-P le nom du programme.
identifiant: nom-P

6 :Type d'association "BE" associe une base de données nom-B à des types d'entités qu'elle possède.

7 :Type d'association "EA" associe un type d'entité à des attributs auxquels il appartient.

8 :Type d'association "ORIGIN" associe un type d'entité origine à des types d'association auxquels ils sont reliés.

9 :Type d'association "CIBLE" associe un type d'entité cible à des types d'association auxquels ils sont reliés.

10 :Type d'association BP associe une base de données à des noms des programmes qui lui sont reliés.

11 :Type d'association BAT associe une base de données à des types d'articles traduits.

12 :Type d'association AAT associe un type d'article traduit à ses itèmes.

6.5.2 :Analyse fonctionnelle:
description des fonctions:

1 :CREER_BD

objectif: introduire des informations concernant une base de données.

message d'entrée: nom-B le nom de la base de données à introduire.

contraintes: les données vérifient les contraintes suivantes:
CT1: nom-B est unique dans l'ensemble des noms actuellement existants.

message de sortie:
sortie 1 : "nom de la BD existe déjà"
sortie 2 : "nom de la BD est introduit"

effet sur le Base des spécifications:
E1 :une occurrence de type d'entité "BASE" est créée.

Régles de traitement: etant donné C(i) et E(i) les conditions et les effets suivants:
C1 :le nom introduit est unique parmi les occurrences de "BASE" actuellement existantes.
E2 :la sortie 1 est produite.
E3 :la sortie 2 est produite.
les régles à respecter sont les suivantes :
régle1: si C1 est vrai alors E1 et E2 sont produits.
régle2: si C1 est faux alors E2 est produit.

2: CREER-ENTITE
objectif: introduire les informations d'un type d'entité pour une base de données déjà positionnée ou via création d'une base de données.

message d'entrée:
nom-E le nom du type d'entité.
nom-B le nom de la base de données.

contraintes:
CT1: nom-B correspond à une base de données existante.
CT2: nom-E est unique dans la base de données nom-B.

message de sortie:
sortie1 : "une occurrence du type d'entité est créé"
sortie2 : "le base de données nom-B n'existe pas"
sortie3 : "la base de données posséde déjà un type d'entité nom-E"

effet sur le BS:
E1 :une occurrence de "ENTITE" et l'association de "BE" sont créées.

règle de traitement:
soient les conditions et les effets suivants:
C1 :nom-B existe déjà parmi des noms déjà introduits.
C2 :nom-E existe déjà dans la base de données 'nom-B'.
E2 :la sortie1 est produite.
E3 :la sortie2 est produite.
E4 :la sortie3 est produite.
les régles à respecter sont les suivantes:
régle1 :si C1 et C2 sont vraies alors E1 et E2 sont produits
régle2 :si C1 est faux alors E3 est produit.
régle3 :si C2 est faux alors E4 est produit.

3: CREER_ATTRIBUT
objectif:introduire les informations d'un attribut après avoir créé un type d'entité ou bien ajouter un attribut directement. Dans ce cas si la base de données n'est pas positionnée on demandra à l'utilisateur de donner les renseignements nécessaires.

message d'entrée:
nom-A le nom de l'attribut.
nom-E le nom du type d'entité auquel l'attribut appartient.
nom-B le nom de la base de données auquel le type d'entité appartient.

contraintes:
CT1 :le nom d'attribut est unique dans la base de données.
CT2 :nom-B correspond à une base de données existante.
CT3 :cette base de données contient bien le type d'entité 'nom-E'.

message de sortie:
sortie1 : "l'attribut est créé"
sortie2 : "la base de données n'existe pas"
sortie3 : "dans la base de données 'nom-B', le type d'entité 'nom-E' n'existe pas"
sortie4 : "attribut 'nom-A' existe déjà dans le type d'entité 'nom-E'"

effet sur BS:
E1 :une occurrence de "ATTRIBUT" et une occurrence de type d'association "EA" sont créées.

règle de traitement:
soient les conditions et les effets suivants:
C1 :nom-B existe déjà dans la base de données.
C2 :la base de données nom-B contient bien le" type d'entité nom-E.
C3 :nom-A est unique parmi les attributs du type d'entité nom-E.
E2 :sortie1 est produite.
E3 :sortie2 est produite.
E4 :sortie3 est produite.
E5 :sortie4 est produite.
les règles à respecter sont les suivantes:
règle 1 :si C1 et C2 et C3 sont vrais alors E1 et E2 sont produits.
règle 2 :si C1 est faux alors E3 est produit.
règle3 :si C1 est vrai et C2 est faux alors E4 est produit.
règle4 :si C3 est faux alors E5 est produit.

4: CREER-ASSOCIATION
objectifs: introduire les informations d'un type d'association.

message d'entrée:
nom-S le nom de l'association.
nom-E1 le nom de type d'entité origine.
nom-E2 le nom de type d'entité cible.
nom-B le nom de la base de données.

contraintes:
CT1: pour une base de données 'nom-S' est unique.
CT2: nom-B correspond à une base de données existante.
CT3: cette base de données contient le type d'entité 'nom-E1'
CT4: cette base de données contient le type d'entité 'nom-E2'
CT5: nom-E1 et nom-E2 sont distincts.

message de sortie:
sortie1 : "type d'association est créée"
sortie2 : "base de données n'existe pas"
sortie3 : "base de données ne contient pas le type d'entité nom-E1"
sortie4 : "base de données ne contient pas le type d'entité nom-E2"
sortie5 : "type d'association nom-S existe déjà dans la base de données"
sortie6 : "il faut deux types d'entité distincts"

effet sur le BS:
E1 :une occurrence de "ASSOCIATION" et une occurrence des types d'association "ORIGINE" et "CIBLE" sont créées.

règle de traitement:
C1 :nom-B existe dans la base de données.
C2 :base de données 'nom-B' contient le type d'entité 'nom-E1'.
C3 :base de données 'nom-B' contient le type d'entité 'nom-E2'.
C4 :nom-S est unique dans la base de données 'nom-B'.
C5 :nom-E1 et nom-E2 sont distincts.
E2 :sortie1 est produite.
E3 :sortie2 est produite.
E4 :sortie3 est produite.
E5 :sortie4 est produite.
E6 :sortie5 est produite.
E7 :sortie6 est produite.

les régles à respecter:
règle1: si C1 à C5 sont vrais alors E1 et E2 sont produits.
règle2: si C1 est faux alors E3 est produit.
règle3: si C1 est vrai et C2 est faux alors E4 est produit.
règle4: si C1 est vrai et C3 est faux alors E5 est produit.
règle5: si C4 est faux alors E6 est produit.
règle6: si C5 est faux alors E7 est produit.

5: SUPPRIMER-BD
objectif: supprimer une base de données.

message d'entrée: nom-B le nom de la base de données à supprimer.

contraintes: les données vérifient les contraintes suivantes:
CT1: nom-B existe parmi les occurrences uniques dans l'ensemble des noms actuellement existants.

message de sortie:
sortie 1 : "BD n'existe pas"
sortie 2 : "BD est supprimée"

effet sur le système d'information:
E1 :les objets suivants sont supprimés:
une occurrence de type d'entité "BASE" est supprimée.
tout type d'entité,attributs,type d'association et leurs liens sont supprimés.

Régles de traitement: étant donné les conditions et les effets suivants:
C1 :le nom introduit existe parmi les occurrences de "BASE".
E2 :la sortie 1 est produite.
E3 :la sortie 2 est produite.

les régles à respecter sont les suivantes :
règle1: si C1 est vrai alors E1 et E3 sont produits.
règle2: si C1 est faux alors E2 est produit.

6: SUPPRIMER-ENTITE
objectif:supprimer un type d'entité et ses attributs.

message d'entrée:
nom-E le nom du type d'entité à supprimer.
nom-B le nom de la base de données.

contraintes:
CT1: nom-B correspond à une base de données existante.
CT2: nom-E correspond à un type d'entité de la base de données 'nom_b'.
message de sortie:
sortie1 : "la base de données nom-B n'existe pas"
sortie2 : "la base de données ne posséde pas le type d'entité nom-E"
sortie3 : "une occurrence du type d'entité est supprimée"

effet sur le BS:
E1 :une occurrence de "ENTITE" et tous ses attributs et types d'associations et leurs liens sont supprimés.

Règle de traitement:
soient les conditions et les effets suivants:
C1 :nom-B existe déjà parmi des noms introduits.
C2 :nom-E existe déjà dans la base de données 'nom-B'.
E2 :la sortie1 est produite.
E3 :la sortie2 est produite.
E4 :la sortie3 est produite.
les régles à respecter sont les suivantes:
règle1 :si C1 et C2 sont vrais alors E1 et E4 sont produits
règle2 :si C1 est faux alors E2 est produit.
règle3 :si C2 est faux alors E3 est produit.

7: SUPPRIMER-ATTRIBUT
objectif:suppression d'un attribut d'un type d'entité.

message d'entrée:
nom-A le nom de l'attribut.
nom-E le nom du type d'entité correspondant
nom-B le nom de la base de données auquel le type d'entité appartient.

contraintes:
CT1 :le nom d'attribut appartient au type d'entité 'nom-E'.

CT2 :nom-B correspond à une base de données existante.
CT3 :cette base de données contient bien le type d'entité 'nom-E'.

message de sortie:
sortie1 : "base de données n'existe pas"
sortie2 : "la base de données 'nom-B' ne possède pas le type d'entité 'nom-E'"
sortie3 : "attribut 'nom-A' n'existe pas dans le type d'entité 'nom-E'"
sortie4 : "l'attribut est supprimé"

effet sur BS:
E1 :une occurrence de "ATTRIBUT" et une occurrence de type d'association "EA" sont supprimées.

soient les conditions et les effets suivants:
C1 :nom-B existe déjà dans la base de données.
C2 :la base de données nom-B contient bien le type d'entité nom-E.
C3 :nom-A est parmi les attributs du type d'entité nom-E.
E2 :sortie1 est produite.
E3 :sortie2 est produite.
E4 :sortie3 est produite.
E5 :sortie4 est produite.

les règles à respecter sont les suivantes:
règle1 :si C1 et C2 et C3 sont vrais alors E1 et E5 sont produits.
règle 2 :si C1 est faux alors E2 est produit.
règle3 :si C2 est faux alors E3 est produit.
règle4 :si C3 est faux alors E4 est produit.

8: SUPPRIMER-ASSOCIATION
objectifs: suppression d'un type d'association.

message d'entrée:
nom-S le nom de l'association.
nom-B le nom de la base de données.

contraintes:
CT1: nom-B correspond à une base de données existante.
CT2: la base de données 'nom-B' posséde un type d'association 'nom-S'.

message de sortie:
sortie1 : "base de données n'existe pas"
sortie2 : "type d'association nom-S n'existe pas dans la base de données"
sortie3 : "type d'association est supprimée"

effet sur le BS:
E1 :une occurrence de "ASSOCIATION" et une occurrence des types d'association "ORIGINE" et "CIBLE" sont supprimées.

Règle de traitement:
C1 :nom-B existe dans la base de données.
C2 :nom-S est dans la base de données 'nom-B'.
E2 :sortie1 est produite.
E3 :sortie2 est produite.
E4 :sortie3 est produite.

les régles à respecter:
règle1: si C1 et C2 sont vrais alors E1 et E4 sont produits.
règle2: si C1 est faux alors E2 est produit.
règle3: si C2 est faux alors E3 est produit.

9: RAPPORT
objectif: obtenir un rapport sur une base de données.

message d'entrée:
nom-B le nom de la base de données pour la quelle on demande un rapport.

contrainte:
CT1: nom-B existe dans la base de données.

message de sortie:
sortie1: "base de données 'nom-B' n'existe pas"
sortie2: rapport de mandé.

effet sur le SI: aucun.

Règles de traitement:
C1: base de données 'nom-B' existe.
E1: la sortie1 est produite.
E2: la sortie2 est produite.

les règles à réspecter:
règle1: si C1 est vrai alors E2 est produit.
règle2: si C1 est faux alors E1 est produit.

10: TRADUCTION-E/A
objectif:traduction d'un schéma E/A à un schéma conforme à dBase.

message d'entrée:
nom-B le nom de la base de données pour la quelle on demande une traduction.

contrainte:
'nom-B' existe parmi les bases de données.

message de sortie:
sortie1: "base de donnée 'nom-B' n'existe pas.
sortie2: la traduction demandée.

effet sur le BS: aucun.

Règles de traitement:
C1: base de données 'nom-B' existe.
E1: la sortie1 est produite.
E2: la sortie2 est produite.

les règles à réspecter:
règle1: si C1 est vrai alors E2 est produit.
règle2: si C1 est faux alors E1 est produit.

CHAPITRE 7: CONCEPTION GLOBALE.

7.0. Introduction.

Aprés avoir un résultat concret de l'analyse fonctionnelle, une étude sur la conception globale (c-à-d la hiérarchisation et la modularisation) est nécessaire pour obtenir une conception détaillée qui suit à une implémentation physique.

7.1: Hiérarchisation.

La relation la plus utilisée dans le domaine du génie-logiciel pour faire une hiérarchisation est la relation *utilise.* Sans entrer dans les détails, cette relation permet à chaque module d'utiliser seulement les modules qui appartiennent aux niveaux plus bas. On a basé l'architecture logique sur une hiérarchie définie sur la relation *utilise.*

-niveau 4 : modules fonctionnels, c-à-d les modules dérivés directement, par composition ou décomposition des fonctions de l'analyse fonctionelle.

-niveau 3: composant de contrôle des contraintes.

-Niveu 3': composant de recherche des éléments.

-niveau 2 : module outil.

-niveau 1 : système d'exploitation.

7.2 : Modularisation

7.2.0 : Notation! les justifications des choix apparaîtront après le signe '**' et seront terminées par le signe '))'

7.2.1 : Module de niveau 1: Système d'exploitation.

7.2.2 : Modules de niveau 2:

SGBD : ce module est défini par :
-la structure des données qu'il gère,
-les primitives qu'il offre pour gérer ces données et pour y accéder,
-les régles d'enchaînements de ces primitives.

GESTION DE L'ECRAN : Contient toutes les fonctions concernant la gestion de l'écran (affichages, menus, saisies etc..).

7.2.3 : module de niveau 3.

contrôl les CI : il s'agit d'une série de fonctions permettant de contrôler l'existence ou la non-existence des éléments de la MBD.

** Les modules des niveaux supérieurs utilisent une série de primitives offertes par ce module pour toutes vérifications de validation))..

7.2.3 : Modules de niveau 3'.

OBTENIR-ELMTS : contient des fonctions travaillant directement sur le SGBD pour obtenir différents éléments du schéma.

** module de niveau 3' Permet aux modules des niveaux supérieurs de travailler d'une façon plus indépendante du SGBD. Il regroupe toutes les recherches et obtention des éléments de la base et les met à la disposition des autres modules utilisateurs.)).

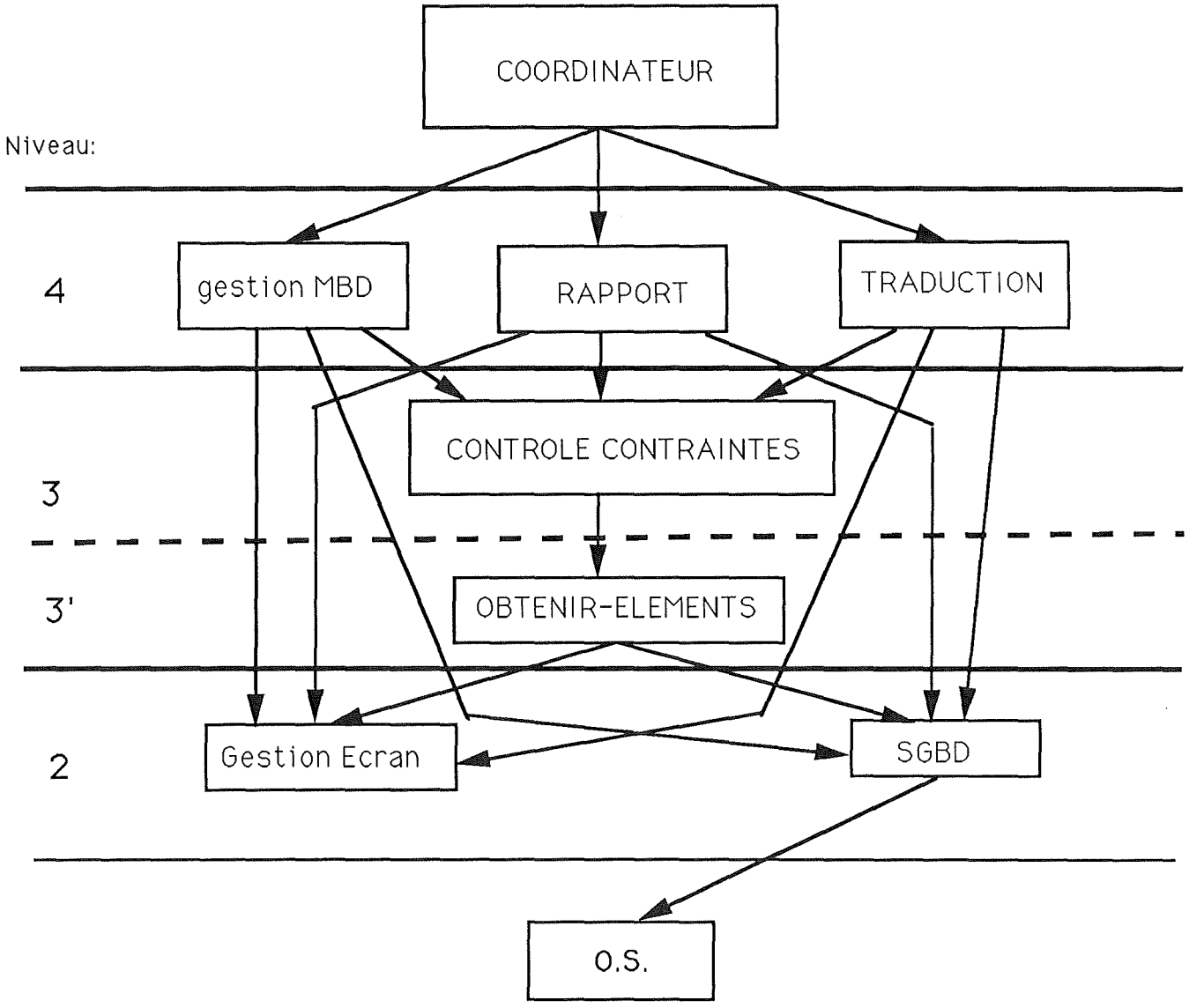


Fig 7.1. Schéma de structure logique de l'atelier.

7.2.6.Modules de niveau 4.

GESTION MBD : ce module regroupe les fonctions suivantes:

-créer, supprimer ou positionner une base de données.
-créer, supprimer ou positionner un type d'entité.
-créer, supprimer ou positionner un attribut.
-créer, supprimer ou positionner une association.

RAPPORT : Ce module se charge de présenter un rapport sur une base de données.

TRAD.SCHEMA : ce module a pour but une transformation d'un schéma E/A à un schéma dBase.

** on a regroupé des fonctions de MAJ de l'analyse fonctionnelle dans le module 'GESTION MBD' car elles traitent le même concept)).

7.2.7 : module "COORDINATEUR"

ce module contiendra le scénario de l'application de l'atelier.

7.3. Définition succincte de chaque module.

La description des modules sera faite du niveau supérieur vers le niveau inférieur pour partir de la couche connue, c-à-dire du schéma conceptuel, vers le couche plus fine à découvrir:

7.3.1. Module coordinateur: Gère le scénario de l'atelier et aiguille l'utilisateur vers les fonctions disponibles et demandées par l'utilisateur. Pour faire cela, il utilise une série de menus avec la possibilité de revenir sur les menus précédents.

7.3.2. Niveau4. Modules fonctionnels. Qui sont dérivés directement des fonctions de l'analyse fonctionnelle:

7.3.2.1. Module Gestion MBD.

CREER-BD: Créer une occurrence du type d'entité *base* (voir schéma de la base des spécifications) en vérifiant l'unicité du nom dans la base. A la fin il appelle à créer un type d'entité sans demande de l'utilisateur. Ce dernier peut la refuser ou créer un type d'entité.

CREER-ENTITE: Créer une occurrence du type d'entité *entité* en vérifiant l'unicité du nom dans la base de données. Comme la fonction précédente , à la fin il appelle à créer un attribut sans demande de l'utilisateur. Ce dérnier peut la refuser ou créer un attribut.

CREER-ATTRIBUT. Créer une occurrence du type d'entité *attribut,* avec la même vérification.

CREER-ASSOCIATION. Créer une occurrence du type d'entité *association,* avec la même vérification.

SUPPRIMER-BD. Supprimer une base de données actuellement existante sur la base des spécifications. Cette existence est une contrainte à vérifier par cette fonction. Au moment de la demande de suppression d'une base de données, cette fonction doit supprimer toutes les occurrences suivantes:

- l'occurrence de la base de données à supprimer sur le type d'entité *base*
-les occurrences éventuelles des types d'entité de la base à supprimer.
-les occurrences éventuelles des attributs de chaque type d'entité de la base à supprimer.
-les occurrences éventuelles des types d'associations de la base à supprimer.

SUPPRIMER-ENT. Supprime l'occurrence du type d'entité, ses attributs (si il y en a) et tous les liens avec des autres entités (y compris des type d'association qui relient un type d'entité à un autre).

SUPPRIMER-ATT. Supprime un attribut d'un type d'entité.

SUPPRIMER-ASS. Supprime l'occurrence du type d'association, ses liens avec des type d'entité origine et le type d'entité cible.

POSITIONNER-BD. Met dans une variable le nom de la base de données saisi par l'utilisateur. A ce moment là, la base de données courante est celle qu'on vient d'introduire et pour toute manipulation de cette base de données, l'utilisateur n'a plus besoin d'introduire le nom de la base de données jusqu'à la fin des opérations.

POSITIONNER-ENT. Met dans une variable le nom du type d'entité saisi par l'utilisateur. Comme la fonction précédente à ce moment là, le type d'entité courante est fixé.

POSITIONNER-ATT. Met dans une variable le nom d'attribut saisi par l'utilisateur.

POSITIONNER-ASS. Comme les fonctions précédentes pour ne pas répéter l'introduction du nom du type d'association.

7.3.2.2. Module RAPPORT. Il s'agit d'afficher ou d'imprimer la structure et la description d'une base de données à la demande de l'utilisateur. Il donne tous les noms et les liens de tous les type d'entités et ses attributs et des types d'associations d'une base de données déjà existante sur la base des spécifications.

7.3.2.3. Module TRADUCTION. Ce module traduit un schéma conceptuel en un schéma conforme au dBase et affiche le résultat.

7.3.3. Niveau 3. Contrôle des contraintes.

Il s'agit des fonctions suivantes pour les quelles les modules du niveau supérieur contrôlent des contraintes détectées en analyse fonctionnelle:

CONTROL-NON-EXISTENCE-BD. La fonction contrôle la non-existence d'une base de données dans la base de spécifications. Une variable booléenne prend les valeurs faux ou vrai pour indiquer le résultat.

CONTROL-EXISTENCE-BD. La fonction contrôle l'existence d'une base de données dans la base de spécifications. Pour cette fonction ainsi que pour toutes les fonctions de contrôle d'existence ou non-existence qui vont suivre une variable booléenne prend les valeurs faux ou vrai pour indiquer le résultat.

CONTROL-NON-EXISTENCE-ENT. La fonction contrôle la non-existence d'un type d'entité dans une base de données.

CONTROL-EXISTENCE-ENT. La fonction contrôle l'existence d'un type d'entité dans une base de données.

CONTROL-NON-EXISTENCE-ATT. La fonction contrôle la non-existence d'un attribut dans un type d'entité.

CONTROL-EXISTENCE-ATT. La fonction contrôle l'existence d'un attribut dans un type d'entité.

CONTROL-NON-EXISTENCE-ASS. La fonction contrôle la non-existence d'un type d'association dans une base de données.

CONTROL-EXISTENCE-ASS. La fonction contrôle l'existence d'un type d'association dans une base de données.

CONTROL-NON-EXISTENCE-ID. La fonction contrôle la non-existence d'un identifiant dans un type d'entité.

Niveau 3'. OBTENIR-ELMTS. Ces fonctions facilitent les recherches des éléments dans la base de données:

OBTENIR-ENTITE. Cherche un type d'entité dans une base de données.

OBTENIR-ATTRIBUT. Cherche un attribut dans un type d'entité.

OBTENIR-ASS-VIA-CIBLE. Cherche un type d'association dont le type d'entité cible est connu.

OBTENIR-ASS-VIA-ORIGINE. Cherche un type d'association dont le type d'entité origine est connu.

7.3.4. Niveau 2.

SGBD: Les primitives du NDBS sont à la disposition des niveaux supérieurs.

Gestion d'écran:

A: Affichages: Les fonctions qui affichent sans saisir des données ou demande d'un choix:

REQUETE-MESSAGE. Affiche sur l'écran un des messages suivants selon le paramètre indiqué par le module utilisateur:
-base de données existe déjà.
-une base de données est créée.
-un type d'entité est créé.
-base de données n'existe pas.
-base de données contient déjà ce type d'entité.
-un attribut est créé.
-la base de données ne contient pas ce type d'entité.
-cet attribut existe déjà dans ce type d'entité.
-une occurrence de type d'association est créée.
-base de données contient déjà ce type d'association.
-il faut deux types d'entité distinctes.
-une base de données est supprimée.
-un type d'entité est supprimé.
-cet attribut n'existe pas.
-attribut est supprimée.

-ce type d'association n'existe pas.
-type d'association est supprimée.
-votre choix est incorrect.
-type d'entité n'existe pas.
-le type d'entité 'ENT.NOME' n'a pas un identifiant.
-fin du RAPPORT.
-... à suivre.
-fin des types d'entité.
-le type d'entité a déjà un identifiant.
-fin de traduction ...

AFFICHAGE-BD. Affiche le défilement de toutes les bases de données existantes sur la base des spécifications dans une petite fenêtre.

AFFICHAGE-ENT. Affiche le défilement de tous les types d'entité existants sur la base de données dans une petite fenêtre.

AFFICHAGE-ATT. Affiche le défilement de tous les attributs existants dans le type d'entité, sur une petite fenêtre.

AFFICHAGE-ASS. Affiche le défilement de tous les types d'association existants sur la base de données dans une petite fenêtre.

B: Les fonctions d'affichage des menus qui demandent une seule donnée de choix d'une fonction.

MENU-PRINCIPAL. Choix d'une fonction principale de l'écran 'Menu-Principal'.

MENU-GESTION. Choix d'une fonction de l'écran ' Gestion MBD'.

MENU-RAPPORT. Affiche le menu de rapport sur une base de données.

MENU-TRADUCTION. Affiche le menu de la traduction d'un schéma ou un programme.

C: Les écrans de saisie: Les fonctions suivantes affichent les menus correspondants à leurs noms pour saisir des données: **MENU-CREATION-BD, MENU-CREATION-ENT, MENU-CREATION-ATT, MENU-CREATION-ASS, MENU-SUPPRIMER-BD, MENU-SUPPRIMER-ENT, MENU-SUPPRIMER-ATT, MENU-SUPPRIMER-ASS.**

Niveau 2'.

CADRE. Crée un rectangle sur l'écran du terminal dont les dimentions sont choisies par les modules qui l'utilisent. Un module qui utilise cette fonction doit donner les paramètres qui communiquent les dimensions du cadre (un rectangle) et sa place sur l'écran. La justification de l'existence de cette fonction est celle d'avoir une bonne visibilité de tous les menus et les affichages.

SORTIR-SAISIR. Affiche sur l'écran le moyen de sortir d'un écran de saisie.

SCHEMA-AIDE. Affiche sur l'écran une lettre 'H' pour les fonctions de positionnement. Cette lettre sur l'écran indique à l'utilisateur qu'il a la possibilité d'avoir des renseignements supplémentaires. (voir écrans positionner-BD etc...).

EFFACE-SCHEMA-AIDE. Efface l'écran précédent.

7.3.5. Niveau 1. Système d'exploitation.

CHAPITRE 8: REALISATION ET TESTS

Pour un test du programme réalisé, on considère le schéma CLIENT-COMMANDE de la page suivante qui est conforme à un schéma E/A utilisé par l'atelier.

En utilisant les écrans expliqués au chapitre 6, on peut introduire les spécifications de ce schéma dans la base des spécifications de l'atelier.

Si on demande un rapport sur ce schéma, il faut choisir sur le menu principal l'option du "rapport". En Fig 9.1., Fig 9.2., Fig 9.3., Fig 9.4. et Fig 9.5., on voit le résultat du rapport préparé par l'atelier. L'atelier donne - pour les raisons de visibilité - le résultat des types d'entité sur les rectangles dont les tailles varient automatiquement selon le nombre des attributs. Dans la partie de présentation des types d'association, chaque association est présentée par les types d'entités origine et cible.

Enfin si on demande à l'atelier de traduire ce schéma on choisit l'option "traduction-schéma". On voit le résultat de la traduction du schéma CLIENT-COMMANDE dans les Fig 9.6. à Fig 9.11.

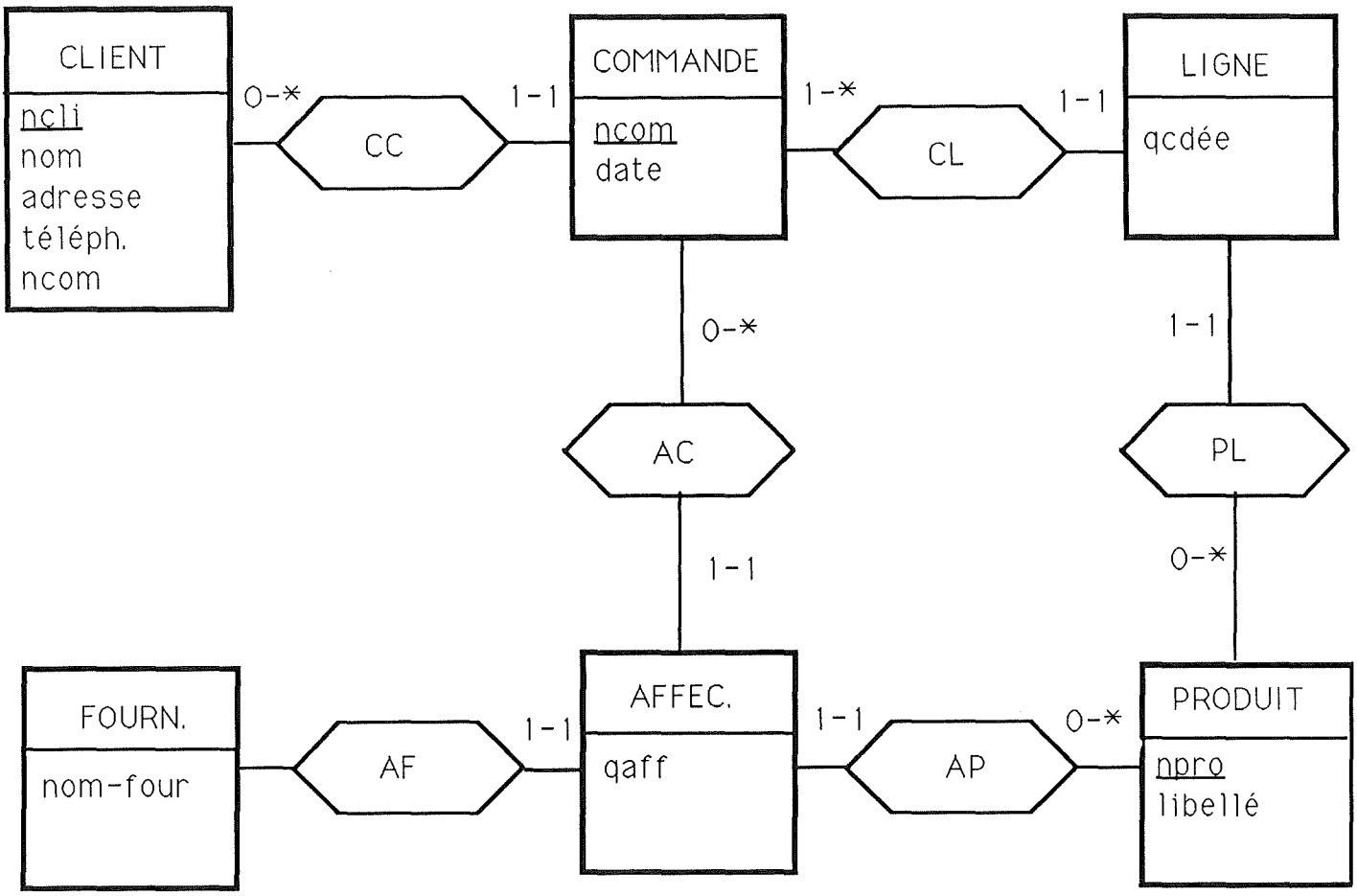
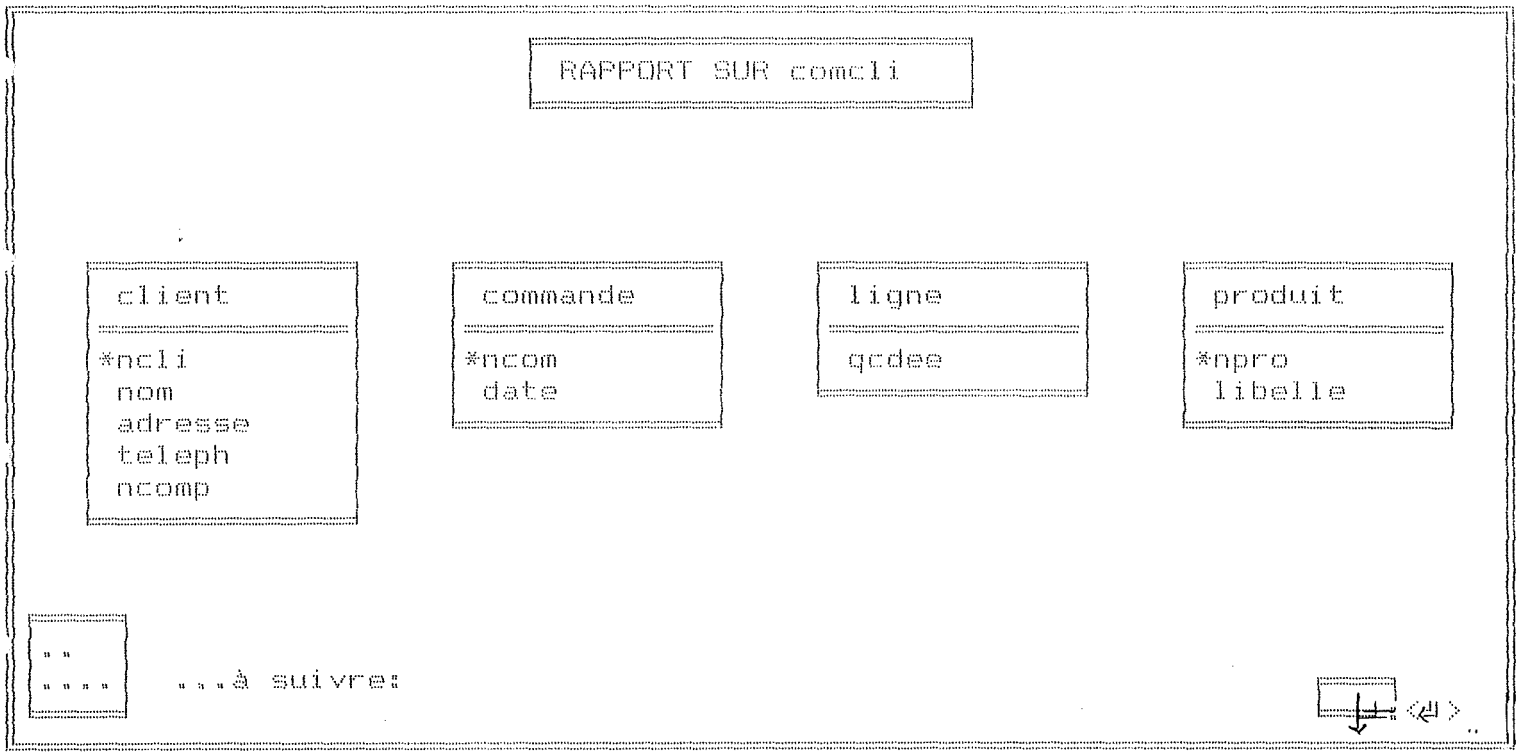


Fig 9.1. Schéma conceptuel CLIENT-COMMANDE.



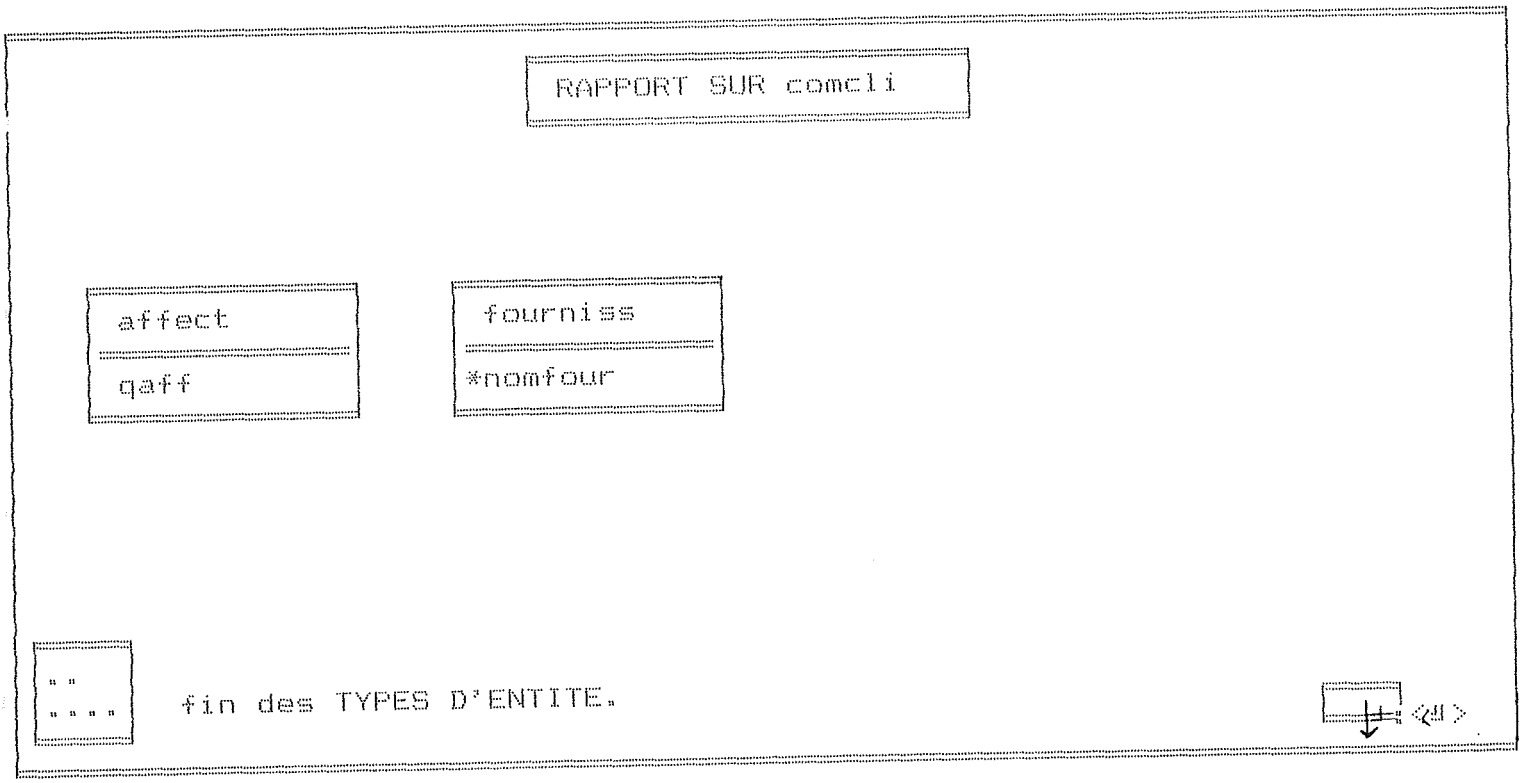
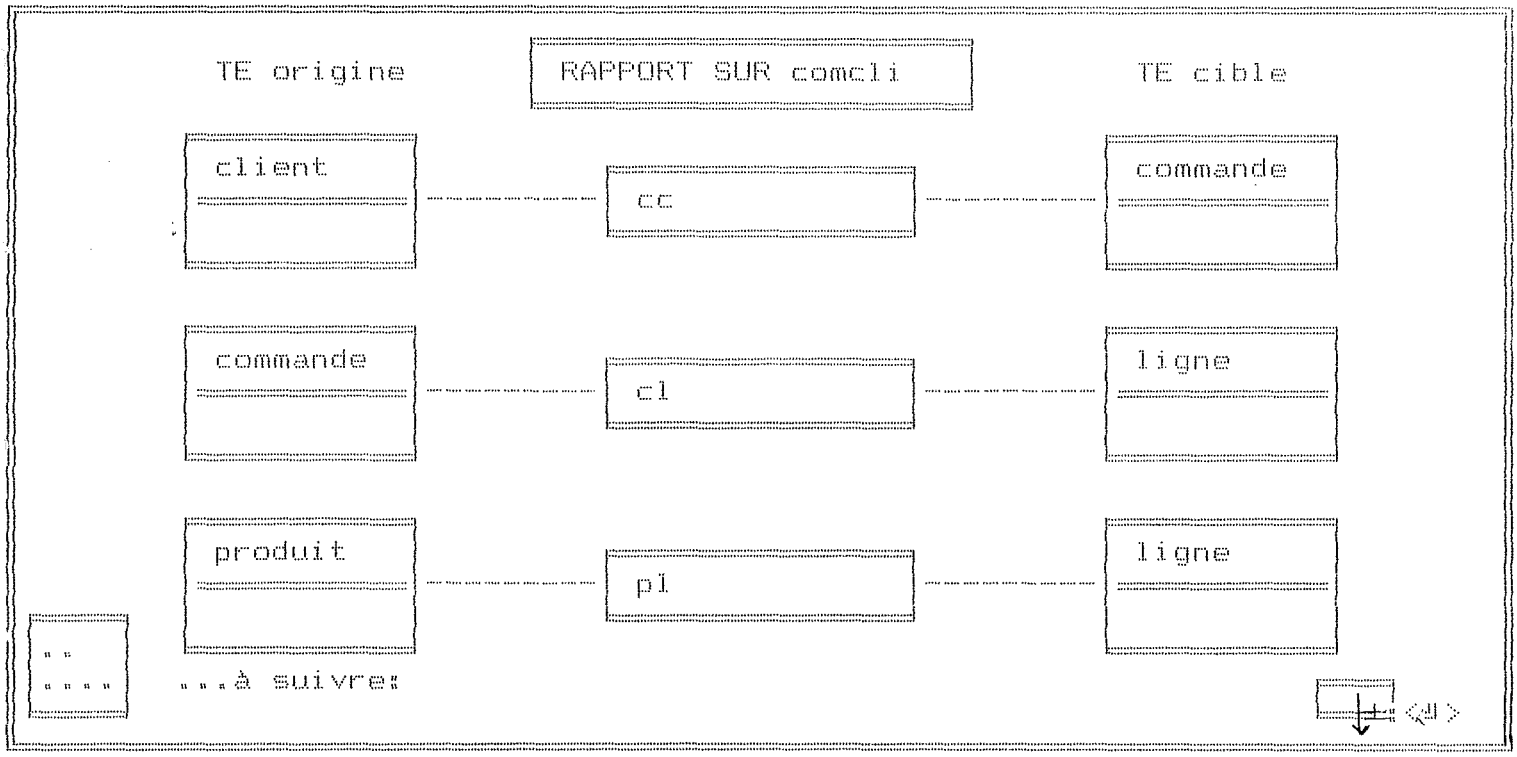


Fig 9.2 et Fig 9.3. Ecrans du rapport sur le schéma CLIENT-COMMANDE



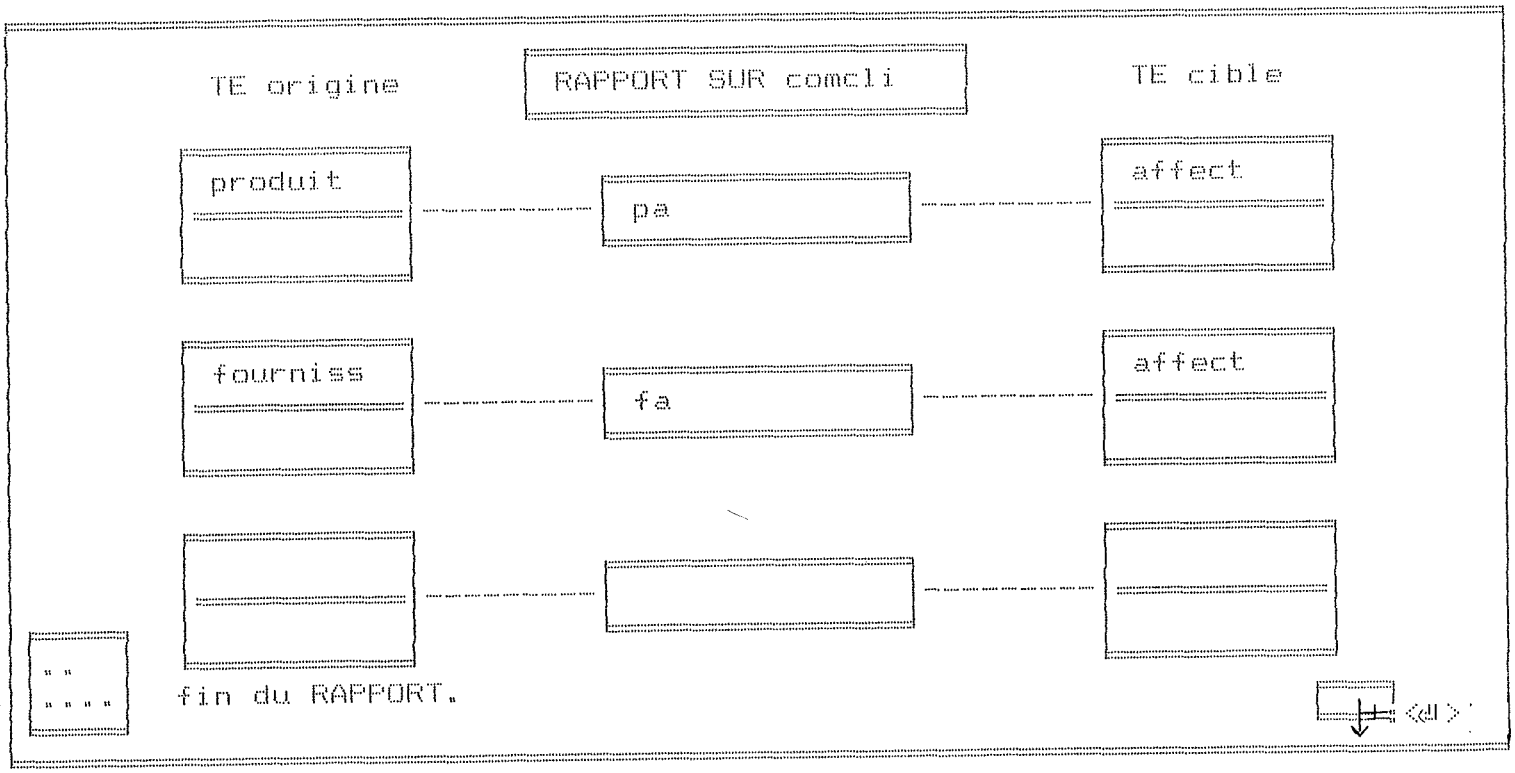
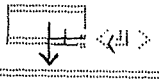


Fig 9.4. et Fig 9.5. Suite des écrans du rapport.

TRADUCTION-SCHEMA

```
Fichier dBase.........client
Champs (Fields):
   ... ncli
   ... nom
   ... adresse
   ... teleph
   ... ncomp
```



TRADUCTION-SCHEMA

```
Fichier dBase.........commande
Champs (Fields):
   ... ncom
   ... date
att.étranger:nclie
Contraintes d'intégrité:
nclie (:commande) in ncli (:client)
```

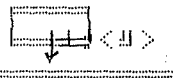
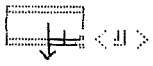


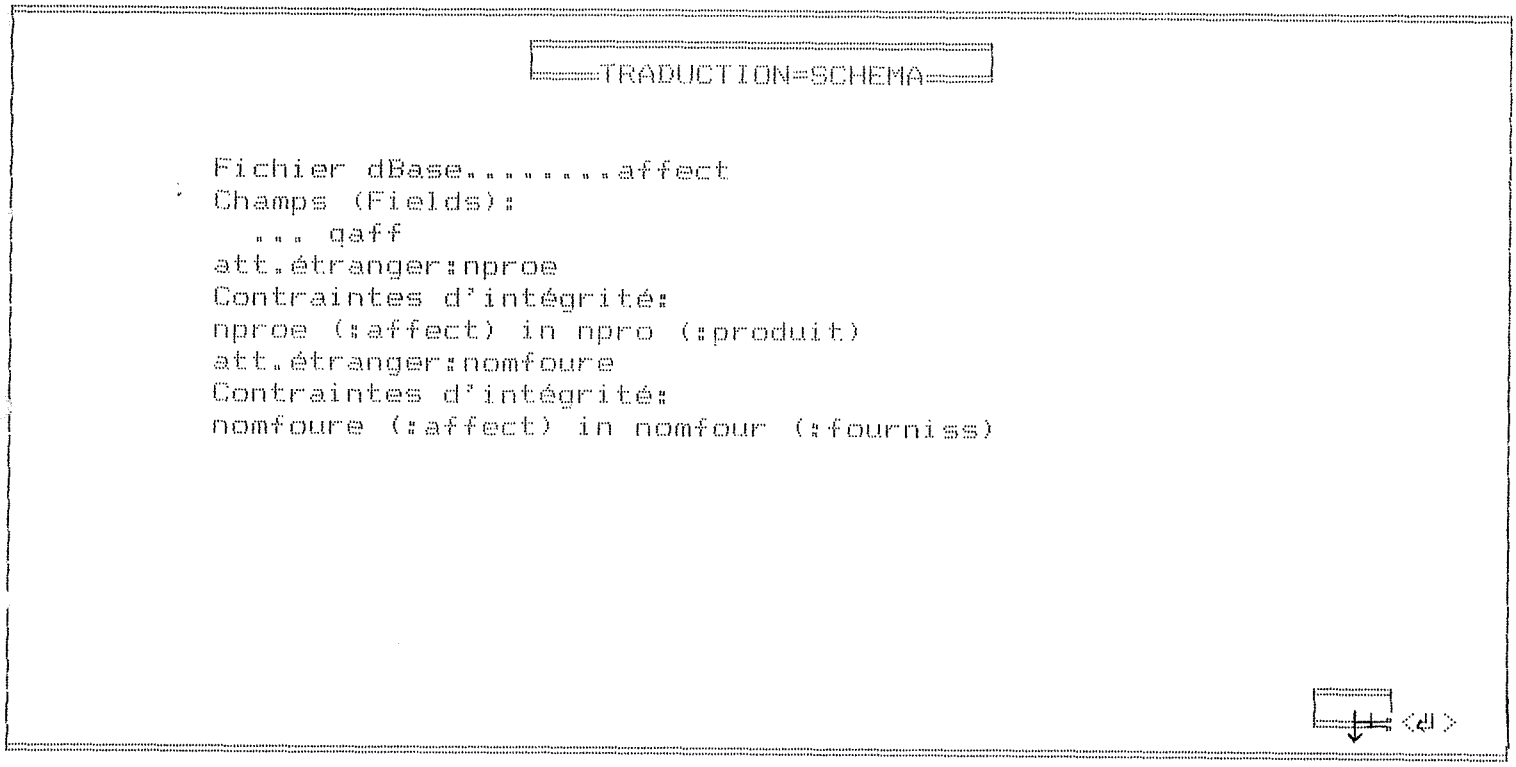
Fig 9.6 et Fig 9.7.

```
TRADUCTION=SCHEMA

Fichier dBase.........ligne
Champs (Fields):
   ... qcdee
. att.étranger:ncome
Contraintes d'intégrité:
ncome (:ligne) in ncom (:commande)
att.étranger:nproe
Contraintes d'intégrité:
nproe (:ligne) in npro (:produit)
```



```
TRADUCTION=SCHEMA

Fichier dBase.........produit
Champs (Fields):
   ... npro
   ... libelle
```

Fig 9.8. et Fig 9.8.



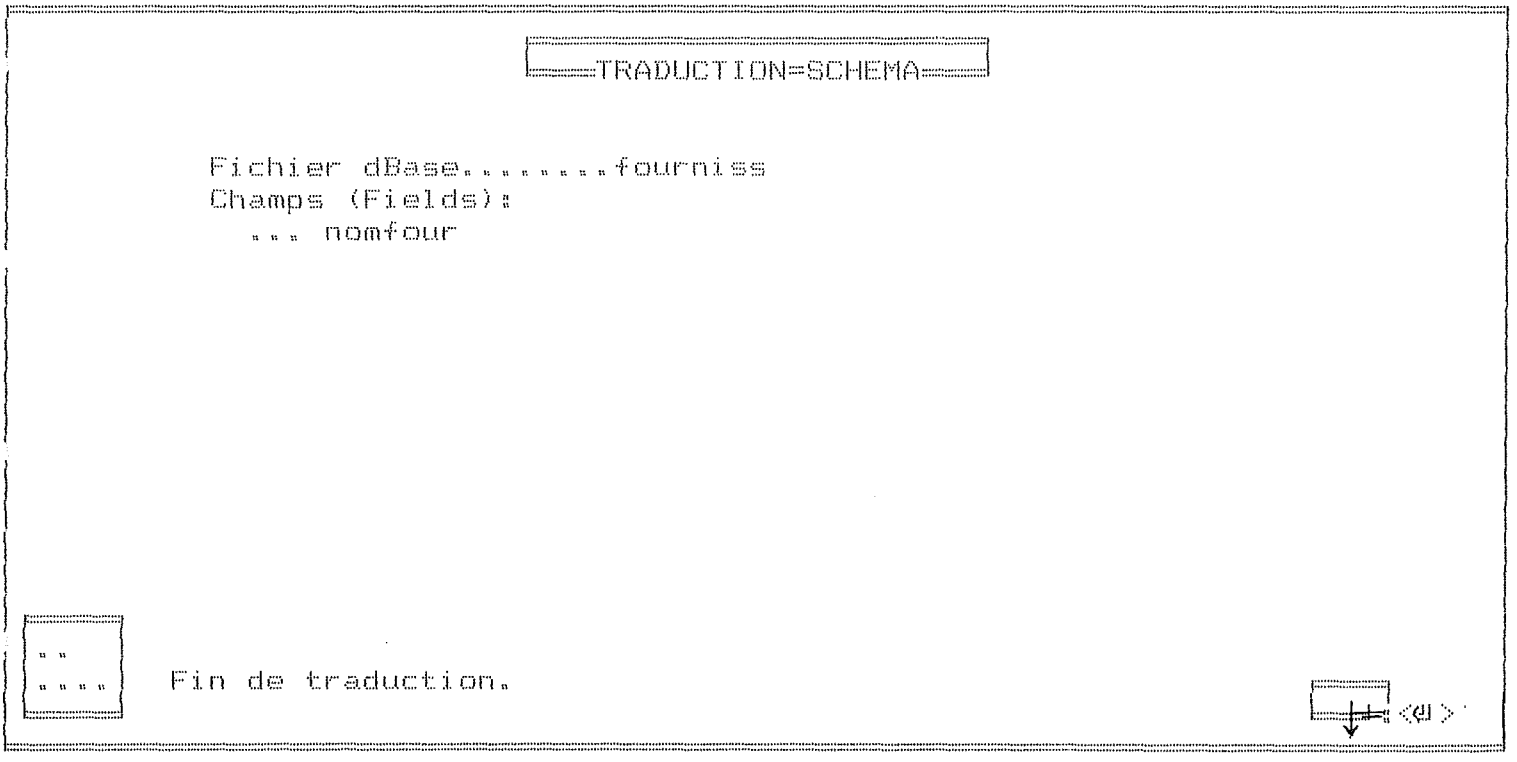


Fig 9.10. et Fig 9.11.

CHAPITRE 9: CONCLUSIONS

L'objectif de ce mémoire était d'élaborer un outil qui permet d'aider un utilisateur lors d'une partie de la démarche de conception des bases de données, notamment en ce qui concerne des applications dBaseIII.

Pour ce faire, nous avons choisi NDBS comme le SGBD dont la base des spécifications s'installe sur ce SGBD.

Une série de processeurs manipul cette base des spécifications afin d'offrir à l'utilisateur des opérations propres à l'atelier. (Fonction de gestion de base des spécifications, générateur de rapports, générateur de traduction d'un schéma conforme à dBase etc).

Comme modèle de structuration des données, on a choisi le modèle Entité/Association. L'atelier n'accepte qu'un modèle Entité/Association restreint et limité. On peut bien imaginer que si on définit un modèle E/A moins limité et plus général que celui acceptable par cet atelier, il devient beaucoup plus pratique et plus utile.

En ce qui concerne l'efficacité de l'atelier, on a essayé à partir d'une analyse fonctionnelle, de faire une hiérarchisation et une modularisation qui rendent une efficacité raisonnable à l'atelier. Cette modularisation nous a permit aussi d'avoir des modules réutilisables.

BIBLIOGRAPHIE

[BOD-PIG, 83/87]
BODART, PIGNEUR, Conception assistée des applications informatiques - 1. Etude d'opportunité et analyse conceptuelle, Masson, Paris, 1983/1988.

[BOUZEGHOUB, 80]
BOUZEGHOUB Mokrane, Thèse présentée à l'université Paris VI.

[BUBENKO, 83]
A. BUBENKO, Information Modeling, Chartwell-Bratt 1983

[DATE, 86]
C.J. DATE, An introduction to data base systems, vol 1, Addison-wesley, 1986.

[DATE,86]
C.J. DATE, Relational databases, Selected writing, Addison-wesley, 1986

[HAINAUT, 86]
HAINAUT, Conception assistée des applications informatiques -2. Conception de la base de données.

[HAINAUT, 86]
HAINAUT, Aspects théoriques et pratiques des modèles Entité-Association, Institut d'informatique, Facultés Universitaires de Namur, 1988.

[HAINAUT, 86]
HAINAUT, Application de l'informatique à la résolution de problèmes, Institut d'informatique, Facultés Universitaires de Namur, 1988.

[HAINAUT, 87]
HAINAUT, NDBS, A simple data base system for small computers, Institut d'informatique, Facultés Universitaires de Namur

[MARTIN, James]
Jamse MARTIN, L'informatique sans programmeur.

[MARTIN, 75]
James MARTIN, Computer Database Organisation.

[ZELKOVITZ, 79]
ZELKOVITZ, SHAW, GANNON, Principales of software Engineering and Design, Prentice Hall, 1979.

TANNEXE 1 :

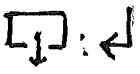
Description détaillée des modules.

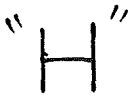
SPECIFICATIONS EXTERNES DE CHAQUE MODULE

module GESTION ECRAN.

Niveau : Il s'agit d'un module de niveau 2.
Relation avec les autres modules : Il n'utilise pas d'autres modules, mais il va être utilisé par des modules de niveaux supérieurs.

Fonction cadre :
Arguments: x,y,u,v : entier. poit1, poit2, poit3, poit4, poit5, poit6: chaînes de caractères.
Précondition: -40 < x,y < 40 et -12 < u,v < 12
Résultat: Affichage d'un cadre avec les dimetions précisées par x,y,u,v sur l'écran de termial.
Postcondition: /

Fonction sortir-saisir :
Arguments: /
Précondition: /
Résultat: Affichage d'un  indiquant comment il faut sortir de l'écran.
Postcondition: /

Fonction schéma-aide :
Arguments: /
Précondition: /
Résultat: Affichage d'un  pour indiquer le choix de "H" pour les cas de besoin d'aid.
Postcondition: /

Fonction efface-schéma-aide :
Arguments: /
récondition: /
Résultat: Simplemet éfface le schéma de la fonction précédante
Postcondition: /

Foncton requête-message :
Arguments: no-message : entier
Précondition: 0 =< no-message =< 27
Résultat: Affichage sur l'écran le méssage numéro no-message.
Postcondition: /

Fonction commentaire :
Arguments: /
Précondition: /
Résultat: Affiche sur l'écran les commentaires sur l'atelier.
Postcondition: /

Fonction affichage_bd :
Arguments: nom-bd : chaîne de caractères.
Précondition: nom-bd est une chaîne de caractères.
Résultat: Affichage de tous les noms des base de données actuellement existantes dans la base de spécifications, sur l'écran.
Postcondition: /

Fonction affichage_ent :
Arguments: nom-bd, nom-ent : chaînes de caractères.
Précondition: nom-bd et nom-ent sont des chaînes de caractères.
Résultat: Affichage de tous les noms des types d'entité actuellement existants dans la base de données nom-bd, sur l'écran.
Postcondition: /

Fonction affichage_att :
Arguments: nom-bd, nom-ent, nom-att: chaînes de caractères.
Précondition: nom-att, nom-bd et nom-ent sont des chaînes de caractères.
Résultat: Affichage de tous les noms des attributs actuellement existants dans le type d'entité
nom-ent, sur l'écran. Postcondition: /

Fonction affichage_ass :
Arguments: nom-bd, nom-ass: chaînes de caractères.
Précondition: nom-ass et nom-bd sont des chaînes de caractères.
Résultat: Affichage de tous les noms des types d'association actuellement existants dans la base de données nom-bd, sur l'écran.
Postcondition: /

Fonction menu_supprimer_att :
Arguments: nom-bd, nom-ent, nom-att: chaînes de caractères.
Précondition: att-positionne est FALSE.
Résultat: Si nom-bd ou nom-ent ou nom-att n'existent pas dans la base de spécifications alors affichage d'erreur, sinon affichage du menu de la suppression d'attribut.
Postcondition: /

Fonction menu_supprimer_ent :
Arguments: nom-bd, nom-ent : chaînes de caractères.
Précondition: ent-positionne est FALSE.
Résultat: Si nom-bd ou nom-ent n'existent pas dans la base de spécifications alors affichage d'erreur, sinon affichage du menu de la suppression de type d'entité .
Postcondition: /

Fonction menu_supprimer_ass
Arguments: nom-bd, nom-ass : chaîne de caractères.
Précondition: ass-positionne est FALSE.
Résultat: Si nom-bd ou nom-ass n'existent pas dans la base de spécifications alors affichage d'erreur, sinon affichage du menu de la suppression de type d'association.
Postcondition: /

Fonction menu_supprimer_bd :
Arguments: nom-bd : chaîne de caractères.
Précondition: bd-positionne est FALSE.
Résultat: Si nom-bd n'existe pas dans la base de spécifications alors affichage d'erreur, sinon affichage du menu de la suppression de la base de données. Postcondition: /

Fonction menu_creation_ass :
Arguments: nom-bd, nom-ass : chaîne de caractères.
Précondition: ass-positionne est FALSE.
Résultat: Si nom-bd n'existe pas dans la base de spécifications alors affichage d'erreur, sinon affichage du menu de la création de type d'association.
Postcondition: /

Fonction menu_creation_att :
Arguments: nom-bd, nom-ent, nom-att: chaînes de caractères.
Précondition: att-positionne est FALSE.
Résultat: Si nom-bd ou nom-ent n'existent pas dans la base de spécifications alors affichage d'erreur, sinon affichage du menu de la création d'attribut.
Postcondition: /

Fonction menu_creation_ent :
Arguments: nom-bd, nom-ent : chaînes de caractères.
Précondition: ent-positionne est FALSE.
Résultat: Si nom-bd n'existe pas dans la base de spécifications alors affichage d'erreur, sinon affichage du menu de la création du type d'entité .
Postcondition: /

Fonction menu_creation_bd :
Arguments: nom-bd : chaîne de caractères.
Précondition: bd-positionne est FALSE.
Résultat: Si nom-bd existe dans la base de spécifications alors affichage d'erreur, sinon affichage du menu de la création de la base de données.
Postcondition: /

Fonction menu_traduction_schema :
Arguments: /
Précondition:/
Résultat: Affichage d'un menu pour demander le nom de la base de données
Postcondition: /

Fonction menu_rapport :
Arguments: /
Précondition:/
Résultat: Affichage à l'écran le menu de la gestion de la base de spécifications et exporter au menu principal le choix de l'utilisateur.
Postcondition: /

Fonction menu_principal :
Arguments: /
Précondition:/
Résultat: Affichage du menu principal de l'atelier à l'écran.
Postcondition: /

Fonction menu_gestion :
Arguments: num-option : caractère.
Précondition:/
Résultat: Affichage d'un menu pour demander le nom de la base de données
Postcondition: /

Module OBTENIR ELEMENTS.

Niveau : Il s'agit d'un module de niveau 3'.
Relation avec les autres modules : Il utilise le module SGBD et le module GESTION ECRAN.

Fonction obtenir-entite :
Arguments: nom-ent, nom-bd : chaînes de caractères, contin ue: booléan.
Précondition: nom-bd est une chaîne de caractères.
Résultat: continue : booléan, nom-ent: chaîne de caractères.
Postcondition: continue est FALSE si la fonction ne trouve pas dans la base de spécifications nom-bd, sinon continue est TRUE, dans ce cas-ci ENT.NOME prend la valeur de nom-ent.

Fonction obtenir-attribut :
Arguments: nom-att,nom-ent, nom-bd : chaînes de caractères, continue : booléanne.
Précondition: nom-att, nom-ent et nom-bd sont des chaînes de caractères.
Résultat: continue : booléanne, nom-att : chaîne de caractères.
Postcondition: continue est FALSE si la fonction ne trouve pas dans la base de spécifications, sinon continue est TRUE, dans ce cas-ci ATT.NOMA prende la valeur de nom-att.

Fonction obtenir-ass-via-origine :
Arguments: nom-ass, nom-bd : chaînes de caractères, contiue : booléanne.
Précondition: nom-bd est une chaîne de caractères.
Résultat: continue : booléanne, nom-ass: chaîne de caractères.
Postcondition: continue est FALSE si la fonction ne trouve pas dans la base de spécifications nom-ass, sinon continue est TRUE, dans ce cas-ci ASS.NOMS prend la valeur de nom-ass.

Fonction obtenir_ass_via_cible :
Arguments: nom-ass,nom-bd : chaînes de caractères, continue:booléanne.
Précondition: nom-bd est une chaîne de caractères.
Résultat: continue : booléanne, nom-ass: chaîne de caractères.
Postcondition: continue est FALSE si la fonction ne trouve pas dans la base de spécifications nom-ass, sinon continue est TRUE, dans ce cas-ci ASS.NOMS prend la valeur de nom-ass.

Module CONTROL-CONTRAINTES.

Niveau : Il s'agit d'un module de niveau 3.
Relation avec les autres modules : Il utilise des modules SGBD (niveau 2) et OBTENIR ELEMENTS (niveau 3').

Fonction control_non_existance_bd :
Arguments: nom-bd-refuse: booléanne, nom-bd : chaîne de caractères.
Précondition: nom-bd est une chaîne de caractères.
Résultat: nom-bd :booléanne.
Postcondition: nom-bd est TRUE si nom-bd existe dans la base de spécifications, FALSE sinon .

Fonction control_existance_bd :
Arguments: nom-bd-refuse: booléanne, nom-bd : chaîne de caractères.
Précondition: nom-bd est une chaîne de caractères.
Résultat: nom-bd :booléanne. **Postcondition:** nom-bd est FALSE si nom-bd existe dans la base de spécifications, TRUE sinon .

Fonction control_non_existance_ent :
Arguments: nom-bd-refuse: booléanne, nom-bd : chaîne de caractères.
Précondition: nom-bd est une chaîne de caractères.
Résultat: nom-bd-refuse :booléanne.
Postcondition: nom-bd-refuse est TRUE si nom-bd existe dans la base de spécifications, FALSE sinon .

Fonction control_existance_ent :
Arguments: nom-ent-refuse : booléanne, nom-bd,nom-ent : chaîne de caractères.
Précondition: nom-bd et nom-ent sont des chaînes de caractères.
Résultat: nom-ent-refuse :booléanne.
Postcondition: nom-ent-refuse est FALSE si nom-ent existe dans la base de données nom-bd, TRUE sinon .

Fonction control_non_existance_att :
Arguments: nom-att-refuse : booléanne, nom-att, nom-bd, nom-ent : chaînes de caractères.
Précondition: nom-att, nom-bd et nom-ent sont des chaînes de caractères.
Résultat: nom-att-refuse :booléanne.
Postcondition: nom-att-refuse est TRUE si nom-att existe dans le type d'entité nom-ent, FALSE sinon .

Fonction control_existance_att :
Arguments: nom-att-refuse : booléanne, nom-att, nom-bd, nom-ent : chaînes de caractères.
Précondition: nom-att, nom-bd et nom-ent sont des chaînes de caractères.
Résultat: nom-att-refuse :booléanne.
Postcondition: nom-att-refuse est FALSE si nom-att existe dans le type d'entité nom-ent, TRUE sinon .

Fonction control_non_existance_id :
Arguments: etat-att-refuse : booléanne, nom-att, nom-bd, nom-ent : chaînes de caractères.
Précondition: nom-att, nom-bd et nom-ent sont des chaînes de caractères.
Résultat: etat-att-refuse : booléanne.
Postcondition: etat-att-refuse est FALSE si il existe un idetifiant parmi des attribut du type d'entité nom-ent, TRUE sinon .

Fonction control_non_existance_ass :
Arguments: nom-ass-refuse : booléanne, nom-ass, nom-bd : chaînes de caractères.
Précondition: nom-ass et nom-bd sont des chaînes de caractères.
Résultat: nom-ass-refuse : booléanne.
Postcondition: nom-ass-refuse est TRUE si nom-ass existe dans la base de données nom-bd, FALSE sinon .

Fonction control_existance_ass :
Arguments: nom-ass-refuse : booléanne, nom-ass, nom-bd : chaînes de caractères.
Précondition: nom-ass et nom-bd sont des chaînes de caractères.
Résultat: nom-ass-refuse : booléanne.
Postcondition: nom-ass-refuse est FALSE si nom-ass existe dans la base da données nom-bd, TRUE sinon .

module GESTION MBD.

Niveau : Il s'agit d'un module de niveau 4.
Relation avec les autres modules : IL utilise des modules de niveaux 2, 3 et 3'.

Fonction trt_supprimer_att :
Arguments: nom-bd, nom-ent, nom-att: chaînes de caractères.
Précondition: nom-bd, nom-ent et nom-att existent dans la base de spécifications.
Résultat: L'occurrence de l'attribut nom-att et sa relation avec type d'entité nom-ent sont supprimées .
Postcondition: /

Fonction trt_supprimer_ent :
Arguments: nom-bd, nom-ent : chaînes de caractères.
Précondition: nom-bd, nom-ent existent dans la base de spécifications.
Résultat: L'occurrence de type d'entité nom-ent et sa relation avec base de données nom-bd sont supprimées .
Postcondition: /

Fonction trt_supprimer_ass :
Arguments: nom-bd, nom-ass : chaînes de caractères.
Précondition: nom-bd, nom-ass existent dans la base de spécifications.
Résultat: L'occurrence du type d'association nom-ass et sa relation avec les type d'entité origine et cible sont supprimées .
Postcondition: /

Fonction trt_supprimer_bd :
Arguments: nom-bd : chaîne de caractères.

Précondition: nom-bd existe dans la base de spécifications.
Résultat: L'occurrence de la base de données nom-bd est supprimée.
Postcondition: /

Fonction trt_creation_ass :

Arguments: nom-bd, nom-ass, nom-ent-cible, nom-ent-origine : chaînes de caractères.
Précondition: nom-bd, nom-ent-cible, nom-ent-origine existent dans la base de spécifications.
Résultat: Une occurrence de type d'association nom-ass et sa relation avec les type d'entité origine et cible sont créées.
Postcondition: /

Fonction trt_creation_att :

Arguments: nom-bd, nom-ent, nom-att: chaînes de caractères.
Précondition: nom-bd et nom-ent existent dans la base de spécifications.
Résultat: Une occurrence de l'attribut nom-att et sa relation avec type d'entité nom-ent sont créées.
Postcondition: /

Fonction trt_creation_ent:

Arguments: nom-bd, nom-ent : chaînes de caractères.
Précondition: nom-bd existe dans la base de spécifications.
Résultat: Ue occurrence de type d'entité nom-ent et sa relation avec base de données nom-bd sont créées.
Postcondition: /

Fonction trt_creation_bd :

Arguments: nom-bd : chaîne de caractères.
Précondition: /
Résultat: Une occurrence de la base de données nom-bd est créée.
Postcondition: /

Fonction trt_traduction_schema :

Arguments: nom-bd, nom-ent : chaînes de caractères.
Précondition: nom-bd existe dans la base de spécifications.
Résultat: Affichage la traduction d'une base de données sous la forme d'un schéma conforme à dBase
Postcondition: /

Fonction supprimer_att :
Arguments: /
Précondition: /
Résultat: Il s'agit du scénario de la suppression d'un attribut. (appels aux fonctions menu-supprimer-att, trt-supprimer-att etc.)
Postcondition: /

Fonction supprimer_ent :
Arguments: /
Précondition: /
Résultat: Il s'agit du scénario de la suppression d'un type d'entité. (appels aux fonctions menu-supprimer-ent, trt-supprimer-ent etc.)
Postcondition: /

Fonction supprimer_ass :
Arguments: /
Précondition: /
Résultat: Il s'agit du scénario de la suppression d'un type dassociation. (appels aux fonctions menu-supprimer-ass, trt-supprimer-ass etc.)
Postcondition: /

Fonction supprimer_bd :
Arguments: /
Précondition: /
Résultat: Il s'agit du scénario de la suppression d'une base de données. (appels aux fonctions menu-supprimer-bd, trt-supprimer-bd etc.)
Postcondition: /

Fonction création_att :
Arguments: /
Précondition: /
Résultat: Il s'agit du scénario de la création d'un attribut. (appels aux fonctions menu-creation-att, trt-creation-att etc.)
Postcondition: /

Fonction création_ass :
Arguments: /
Précondition: /
Résultat: Il s'agit du scénario de la création d'un type d'association. (appels aux fonctions menu-creation-ass, trt-creation-ass etc.)
Postcondition: /

Fonction création_ent :
Arguments: /
Précondition: /
Résultat: Il s'agit du scénario de la création d'un type d'entité. (appels aux fonctions menu-creation-ent, trt-creation-ent etc.)
Postcondition: /

Fonction création_bd :
Arguments: /
Précondition: /
Résultat: Il s'agit du scénario de la création d'une base de données. (appels aux fonctions menu-creation-bd, trt-creation-bd etc.)
Postcondition: /

Fonction positionner_bd :
Arguments: /
Précondition: /
Résultat: Il s'agit du scénario de positionner d'une base de données. (appels aux fonctions menu-positionner-bd, affichage-bd etc.)
Postcondition: /

Fonction positionner_ent :
Arguments: /
Précondition: /
Résultat: Il s'agit du scénario de positionner d'un type d'entité. (appels aux fonctions menu-positionner-ent, affichage-ent etc.)
Postcondition: /

Fonction positionner_ass :
Arguments: /
Précondition: /
Résultat: Il s'agit du scénario de positionner d'un type d'association. (appels aux fonctions menu-positionner-ass, affichage-ass etc.)
Postcondition: /

Fonction positionner_att :
Arguments: /
Précondition: /
Résultat: Il s'agit du scénario de positionner d'une attribut. (appels aux fonctions menu-positionner-att, affichage-att etc.)
Postcondition: /

Module RAPPORT.

Niveau : Il s'agit d'un module de niveau 4.
Relation avec les autres modules : Il utilise des modules deniveaux inférieurs.

Fonction rapport :

Arguments:nom-bd : chaîne de caractères.
Précondition: nom-bd est une chaîne de caractères.
Résultat: Prépare un rapport sur la base de données nom-bd.
Postcondition: /

Module TRADUCTION SCHEMA.

Niveau : Il s'agit d'un module de niveau 4.
Relation avec les autres modules : Il utilise des modules deniveaux inférieurs.

Fonction traduction-schéma :

Arguments:nom-bd : chaîne de caractères.
Précondition: nom-bd est une chaîne de caractères.
Résultat: Traduit la base de données nom-bd en schéma conforme à dBase et l'affiche sur l'écran.
Postcondition: /

Module COORDINATEUR.

Il importe et exporte des informations avec les modules de niveau 4.

Fonction programme principal :

Arguments: /
Précondition: /
Résultat: Consiste à scénario et appels aus différentes composantes de l'atelier.
Postcondition: /

ANNEXE 2 :

Texte du programme de l'atelier.

```
      Line 12   Col 4   Insert     Indent  A:ATELIER.PAS
program memoir;
(*$I metabd.type*)
(*$I a:\dbms.pas*)
type
     chaine=string[20];
     fichtext=text;
var fichprg:fichtext;
   no_message,color:integer;
   num_option,no_requete_menu,pour_continuer,etat_att:char;
   point1,point2,point3,point4,point5,point6,imprime:char;
   bd_positionnee,ent_positionne,att_positionne,ass_positionnee:boolean;
   menu_bd_positionner,menu_ent_positionner,menu_att_positionner,
   menu_ass_positionner:boolean;
   nom_ass_refuse,nom_att_refuse,nom_ent_refuse,nom_bd_refuse:boolean;
   premiere_passage_ent,premiere_passage_bd,etat_att_refuse:boolean;
   nom_ent_or_refuse,nom_ent_cible_refuse:boolean;
   creation_attribut_direct,creation_entite_direct:boolean;
   supprimer_base,control_menu_principal,mbd,traduc:boolean;
   continue,retour_menu_principal,retour_sous_menu:boolean;
       x,y,u,v:integer;
       nom_prg,nom_ass,nom_att,nom_ent,nom_bd:chaine;
       nom_ent_or,nom_ent_cible:chaine;
       BD:TBASE;
       ENT:TENTITE;

       Line 35   Col 4   Insert     Indent  A:ATELIER.PAS
       ENT:TENTITE;
       ATT:TATTRIBUT;
       ASS:TASSOCIATION;

procedure cadre(x,y,u,v:integer;
                point1,point2,point3,point4,point5,point6:char);
       var i,j:integer;
       begin
         gotoxy((40-x),(12-u));writeln(point1);
         gotoxy((40+y),(12-u));writeln(point2);
         gotoxy((40-x),(12+v));writeln(point3);
         gotoxy((40+Y),(12+v));writeln(point4);
         for i:=41-x to 39+y do
             begin
                gotoxy(i,(12-u));writeln(point5);
                gotoxy(i,(12+v));writeln(point5);
             end;
          for j:=13-u to 11+v do
               begin
                 gotoxy((40-x),j);writeln(point6);
                 gotoxy((40+y),j);writeln(point6);
               end;
         end;
procedure sortir_saisir;
```

```
      Line 58   Col 4   Insert     Indent  A:ATELIER.PAS
procedure sortir_saisir;
begin
  gotoxy(74,23);writeln(':<',chr(17),chr(188),'>');
  cadre(-29,33,-10,11,chr(213),chr(184),chr(212),chr(190),chr(205),chr(17
  gotoxy(71,23);writeln(chr(25));
  end;

procedure schema_aide;
begin
  cadre(-18,23,4,-3,chr(213),chr(184),chr(212),chr(190),chr(205),' ');
  cadre(-15,18,5,-2,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179)
  cadre(-23,26,5,-2,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179)
end;

procedure efface_schema_aide;
begin
  cadre(-18,23,4,-3,' ',' ',' ',' ',' ',' ',' ',' ');
  cadre(-15,18,5,-2,' ',' ',' ',' ',' ',' ',' ',' ');
  cadre(-23,26,5,-2,' ',' ',' ',' ',' ',' ',' ',' ');
end;


procedure requete_message(no_message:integer);
begin

      Line 93   Col 4   Insert     Indent  A:ATELIER.PAS
   textcolor(7);
   sound(300);delay(300);nosound;
   cadre(38,-33,-8,11,chr(213),chr(184),chr(212),chr(190),chr(205),chr(1
   gotoxy(3,21);writeln('..');
   gotoxy(3,22);writeln('....');
   textmode;
   gotoxy(10,22);
   case no_message of
        0:writeln('Base de données déjà existe.
        1:writeln('Une base de données est crée.
        2:writeln('Un type d''entité est crée.
        3:writeln('Base de données n''existe pas.
        4:writeln('Base de données déjà contient ce type d''entité.
        5:writeln('Un attribut est crée.
        6:writeln('La base de données ne contient pas ce type d''entité.
        7:writeln('Attribut déjà existe dans ce type d''entité.
        8:writeln('Une occurrence de type d''association est crée.
        9:writeln('Base de données déjà contient ce type d''association.
        10:writeln('Il faut deux types d''entité distincts.
        11:writeln('Une base de données est supprimée.
        12:writeln('Un type d''entité est supprimé.
        13:writeln('Cet attibut n''existe pas.
        14:writeln('Attribut est supprimé.
        15:writeln('Ce type d''association n''existe pas.          ',AS
```

```
Line 95   Col 1   Insert    Indent  A:ATELIER.PAS
  16:writeln('Type d''association est supprimée.
  17:writeln('Votre choix est incorrect.
  18:writeln('Type d''entité n''existe pas.
  19:writeln('Le type d''entité  ',ENT.NOME,' n''a pas un identifia
  20:writeln('fin du RAPPORT.
  21:writeln('...à suivre:
  22:writeln('fin des TYPES D''ENTITE.
  23:writeln('Le type d''entité a déjà un attribut identifiant.
  24:writeln('Base de donnée n''est pas positionée.
  25:writeln('Type d''entité n''est pas positionné.
  26:writeln('Fin de traduction.
  27:writeln('Fin commentaire.
  end;
  textcolor(7);
  sortir_saisir;
  gotoxy(79,23);readln(pour_continuer);
  textmode;
end;
```

```
      Line 54    Col 1    Insert     Indent   A:ATELIER.PAS

procedure menu_gestion;
var choix_article,choix_operation:char;
begin
  clrscr;
  creation_entite_direct:=true;
  creation_attribut_direct:=true;
  cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179)
  cadre(9,8,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179));
  gotoxy(34,3);writeln('Gestion=MBD');
  cadre(31,17,5,8,chr(201),chr(187),chr(200),chr(188),chr(205),chr(186));
  textcolor(7);
  cadre(29,15,4,3,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179));
  gotoxy(15,10);writeln('ositionner         ase de donées.');
  gotoxy(15,11);writeln('réer               type d'' ntité.');
  gotoxy(15,12);writeln('upprimer           ttribut.');
  gotoxy(14,13);writeln('                   type d''as ociation.');
  gotoxy(13,18);writeln('etour au  enu principal.');
  gotoxy(12,19);writeln('votre choix ~',chr(26));
  textmode;
  gotoxy(14,10);writeln('P');  gotoxy(32,10);writeln('B');
  gotoxy(14,11);writeln('C');  gotoxy(39,11);writeln('E');
  gotoxy(14,12);writeln('S');  gotoxy(32,12);writeln('A');
  gotoxy(41,13);writeln('S');

      Line 78    Col 1    Insert     Indent   A:ATELIER.PAS
  gotoxy(12,18);writeln('R');  gotoxy(22,18);writeln('M');
  gotoxy(27,19);read(kbd,choix_operation);writeln(choix_operation);
  gotoxy(28,19);read(kbd,choix_article);writeln(choix_article);delay(150)
  case choix_operation of
      'P':begin
          case choix_article of
              'B':num_option:='G';
              'E':num_option:='H';
              'A':num_option:='I';
              'S':num_option:='J';
              else num_option:='X';
              end;
          end;
    'C':begin
          case choix_article of
              'B':num_option:='B';
              'E':num_option:='E';
              'A':num_option:='A';
              'S':num_option:='C';
              else num_option:='X';
              end;
          end;
    'S':begin
          case choix_article of
```

```
Line 102  Col 1    Insert     Indent  A:ATELIER.PAS
              'B':num_option:='S';
              'E':num_option:='U';
              'A':num_option:='P';
              'S':num_option:='M';
               else num_option:='X';
               end;
          end;
      'R':if choix_article='M' then num_option:='R' else num_option:='X';
      else num_option:='X';
      end;
end;
```

```
      Line 117  Col 1    Insert     Indent  A:ATELIER.PAS

procedure menu_traduction;
begin
  clrscr;
  cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179)
  cadre(13,15,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179)
  gotoxy(30,3);writeln('Traduction=et=Programme');
  cadre(31,10,4,8,chr(201),chr(187),chr(200),chr(188),chr(205),chr(186));
  textcolor(7);
  cadre(29,8,3,3,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179));
  gotoxy(15,11);writeln('raduction d''un schéma E/A.');
  gotoxy(15,12);writeln('crire d''un programme.');
  gotoxy(14,13);writeln('traduction d''un  rogramme LDA.');
  gotoxy(13,18);writeln('etour au menu principal.');
  gotoxy(12,19);writeln('votre choix -',chr(26));
  textmode;
  gotoxy(14,11);writeln('T');
  gotoxy(14,12);writeln('E');
  gotoxy(30,13);writeln('P');
  gotoxy(12,18);writeln('R');
  gotoxy(27,19);read(kbd,num_option);
  gotoxy(27,19);writeln(num_option);delay(100);
end;
```

```
       Line 140  Col 1    Insert     Indent  A:ATELIER.PAS

procedure requete_menu;
var chcorr:boolean;
begin
    chcorr:=false;
    mbd:=false;
    traduc:=false;
    case no_requete_menu of
      'G':begin
          repeat
             menu_gestion;
             if num_option in['G','H','I','J','B','E','A','C','S','U','P',
             if not chcorr then requete_message(17);
             until num_option in['G','H','I','J','B','E','A','C','S','U','
             mbd:=true;
        end;
      'R':num_option:='Z';
      'T':begin
          repeat
             menu_traduction;
             if num_option in['T','E','P','R'] then chcorr:=true;
             if not chcorr then requete_message(17);
             until num_option in['T','E','P','R'];
             traduc:=true;

       Line 186  Col 1    Insert     Indent  A:ATELIER.PAS
             end;
      'C':num_option:='D';
      'Q':num_option:='R';
  end;
 end;

procedure menu_principal;
begin
  clrscr;
  cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179)
  cadre(9,8,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179));
  gotoxy(33,3);writeln('Menu=Principal');
  cadre(30,12,4,8,chr(201),chr(187),chr(200),chr(188),chr(205),chr(186));
  textcolor(7);
  cadre(28,10,3,3,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179));
  gotoxy(16,11);writeln('estion de metabase de données.');
  gotoxy(16,12);writeln('apport sur une base de données.');
  gotoxy(16,13);writeln('raductions et programmes.');
  gotoxy(13,17);writeln('ommentaire.');
  gotoxy(13,18);writeln('uitter.');
  gotoxy(12,19);writeln('votre choix -',chr(26));
  textmode;
  gotoxy(15,11);writeln('G');
  gotoxy(15,12);writeln('R');
```

```
      Line 210  Col 1    Insert     Indent   A:ATELIER.PAS
  gotoxy(15,13);writeln('T');
  gotoxy(12,17);writeln('C');
  gotoxy(12,18);writeln('Q');
  gotoxy(27,19);read(kbd,no_requete_menu);
  gotoxy(27,19);writeln(no_requete_menu);delay(100);
end;

procedure en_tete;
begin
  clrscr;
  cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179)
  for color:=1 to 2 do begin
  textcolor(7*color);
  cadre(8,8,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179));
  gotoxy(37,3);writeln('*FuNDP*');
  cadre(22,22,4,4,chr(201),chr(187),chr(200),chr(188),chr(205),chr(186));
  gotoxy(80,24);
  if color=2 then begin
     gotoxy(21,10);writeln('atelier de développement d''application');
     gotoxy(36,12);writeln('dBASE III');
     gotoxy(80,24);
     delay(1200);
                  end;
                  delay(1000);

      Line 234  Col 1    Insert     Indent   A:ATELIER.PAS
                        end;
  textmode;
end;
```

```
Line 239  Col 21  Insert     Indent  A:ATELIER.PAS

                    (* DEBUT MODULE  OBTENIR ELEMENTS   *)

procedure obtenir_entite;
begin
  continue:=true;
  BD.NOMB:=nom_bd;
  dbid(BASE,BD);
  dbfpath(ENT,BD,BE);
  if ENT.NOME=nom_ent then continue:=false;
  while continue and dbfound do
    begin
     dbnpath(ENT,BD,BE);
     if ENT.NOME=nom_ent then continue:=false;
    end;
end;

procedure obtenir_attribut;
begin
  obtenir_entite;
  continue:=true;
  dbfpath(ATT,ENT,EA);

      Line 283  Col 1   Insert     Indent  A:ATELIER.PAS
  if ATT.NOMA=nom_att then continue:=false;
  while continue and dbfound do
    begin
     dbnpath(ATT,ENT,EA);
     if ATT.NOMA=nom_att then continue:=false;
    end;

end;

procedure obtenir_ass_via_origine;
begin
  obtenir_entite;
  continue:=true;
  dbfpath(ASS,ENT,ORIGINE);
  if ASS.NOMS=nom_ass then continue:=false;
  while continue and dbfound do
    begin
     dbnpath(ASS,ENT,ORIGINE);
     if ASS.NOMS=nom_ass then continue:=false;
    end;
end;
```

```
      Line 305  Col 1    Insert     Indent  A:ATELIER.PAS
procedure obtenir_ass_via_cible;
begin
  obtenir_entite;
  continue:=true;
  dbfpath(ASS,ENT,CIBLE);
  if ASS.NOMS=nom_ass then continue:=false;
  while continue and dbfound do
    begin
     dbnpath(ASS,ENT,CIBLE);
     if ASS.NOMS=nom_ass then continue:=false;
    end;
 end;
```

```
       Line 284  Col 53  Insert    Indent  A:ATELIER.PAS

                  (*DEBUT MODULE CONTROLE CONTRAINTES    *)

procedure control_non_existance_bd;
begin
BD.NOMB:=nom_bd;
     dbid(BASE,BD);
     if dbfound  then nom_bd_refuse:=true
     else nom_bd_refuse:=false;
end;
procedure control_existance_bd;
begin
     BD.NOMB:=nom_bd;
     dbid(BASE,BD);
     if dbfound  then nom_bd_refuse:=false
     else nom_bd_refuse:=true;
end;


procedure control_non_existance_ent;
begin
  obtenir_entite;
  if continue then nom_ent_refuse:=false;

       Line 328  Col 53  Insert    Indent  A:ATELIER.PAS
end;

procedure control_existance_ent;
begin
     obtenir_entite;
    if continue then nom_ent_refuse:=true else nom_ent_refuse:=false;
end;

procedure control_non_existance_att;
begin
     obtenir_attribut;
     if continue then nom_att_refuse:=false;
end;

procedure control_existance_att;
begin
     obtenir_attribut;
     if not continue then nom_att_refuse:=false;
end;
procedure control_non_existance_id;
begin
  obtenir_entite;
  etat_att_refuse:=false;
  dbfpath(ATT,ENT,EA);
```

```
       Line 352  Col 53  Insert     Indent  A:ATELIER.PAS
  while dbfound and not etat_att_refuse do
    begin
     if ATT.ETAT='o' then etat_att_refuse:=true;
     dbnpath(ATT,ENT,EA);
    end;
end;

procedure control_non_existance_ass;
var continuer:boolean;
begin
dbid(BASE,BD);
continuer:=true;
dbfpath(ENT,BD,BE);
  while dbfound and continuer do
    begin
      nom_ent:=ENT.NOME;
      obtenir_ass_via_origine;
      if not continue then continuer:=false;
      dbnpath(ENT,BD,BE);
    end;
 if continue then nom_ass_refuse:=false else nom_ass_refuse:=true;
end;

procedure control_existance_ass;

       Line 376  Col 53  Insert     Indent  A:ATELIER.PAS
var encore,continue:boolean;
    nom_ass_inter:chaine;
begin
  continue:=true;
  encore:=true;
  BD.NOMB:=nom_bd;
  dbid(BASE,BD);
  dbfpath(ENT,BD,BE);
  while dbfound and continue do
   begin
    dbfpath(ASS,ENT,ORIGINE);
    while dbfound and encore do
      begin
       if ASS.NOMS=nom_ass then
        begin
          encore:=false;
          continue:=false;
          nom_ent_or:=ENT.NOME;
          dbfpath(ENT,ASS,-CIBLE);
          nom_ent_cible:=ENT.NOME;
         end;
       dbnpath(ASS,ENT,ORIGINE);
      end;
   dbnpath(ENT,BD,BE);
```

```
       Line 352  Col 53  Insert     Indent  A:ATELIER.PAS
   while dbfound and not etat_att_refuse do
     begin
      if ATT.ETAT='o' then etat_att_refuse:=true;
      dbnpath(ATT,ENT,EA);
     end;
end;

procedure control_non_existance_ass;
var continuer:boolean;
begin
dbid(BASE,BD);
continuer:=true;
dbfpath(ENT,BD,BE);
   while dbfound and continuer do
     begin
       nom_ent:=ENT.NOME;
       obtenir_ass_via_origine;
       if not continue then continuer:=false;
       dbnpath(ENT,BD,BE);
     end;
  if continue then nom_ass_refuse:=false else nom_ass_refuse:=true;
end;

procedure control_existance_ass;

       Line 376  Col 53  Insert     Indent  A:ATELIER.PAS
var encore,continue:boolean;
    nom_ass_inter:chaine;
begin
   continue:=true;
   encore:=true;
   Mb.NOMM:=nom_bd;
   dbid(BASE,BD);
   dbfpath(ENT,BD,BE);
   while dbfound and continue do
    begin
     dbfpath(ASS,ENT,ORIGINE);
     while dbfound and encore do
       begin
        if ASS.NOMS=nom_ass then
         begin
           encore:=false;
           continue:=false;
           nom_ent_or:=ENT.NOME;
           dbfpath(ENT,ASS,-CIBLE);
           nom_ent_cible:=ENT.NOME;
         end;
        dbnpath(ASS,ENT,ORIGINE);
       end;
    dbnpath(ENT,BD,BE);
```

```
      Line 385  Col 49  Insert     Indent  A:ATELIER.PAS
  end;
  if continue then nom_ass_refuse:=true else nom_ass_refuse:=false;
end;



                  (*      DEBUT MODULE GESTION MBD          *)


procedure trt_supprimer_att;
var continuer:boolean;
begin
   obtenir_entite;
   obtenir_attribut;
   dbremove(ATT,ENT,EA);
   dbdelete(ATTRIBUT,ATT);
   requete_message(14);
 end;


procedure menu_supprimer_att;
var repeter_menu_att:boolean;
  begin

      Line 424  Col 49  Insert     Indent  A:ATELIER.PAS
    nom_att_refuse:=true;
    nom_ent_refuse:=true;
    nom_bd_refuse:=true;
    repeter_menu_att:=true;
    while repeter_menu_att do
    begin
       clrscr;
       cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       cadre(11,12,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       if menu_att_positionner then begin
             gotoxy(31,3);writeln('POSITIONNER=ATTRIBUT');
                                  end else begin
              gotoxy(32,3);writeln('SUPPRIMER=ATTRIBUT');
                                       end;
       cadre(30,13,4,0,chr(201),chr(187),chr(200),chr(188),chr(205),chr(1
       textcolor(7);
       cadre(30,18,-4,7,chr(201),chr(187),chr(200),chr(188),chr(205),chr
       gotoxy(12,9);writeln('Le nom de la base de données:');
       gotoxy(12,11);writeln('Le nom du type d''entité:');
       if menu_att_positionner then begin
                       gotoxy(12,18);writeln('Nom d''attribut à positionne
                                  end else begin
                       gotoxy(12,18);writeln('Le nom d''attribut à suppri
                                       end;
```

```
Line 448  Col 49  Insert     Indent  A:ATELIER.PAS
 sortir_saisir;
 textmode;

  if (not nom_bd_refuse and not nom_ent_refuse) or ent_positionne t
     begin
      gotoxy(42,9);writeln(nom_bd);
      gotoxy(37,11);writeln(nom_ent);
     end;
 if not nom_bd_refuse or (bd_positionnee and not ent_positionne)  t
     begin
       gotoxy(42,9);
       writeln(nom_bd);
     end;
 if nom_bd_refuse and not bd_positionnee then begin
     gotoxy(42,9);
     readln(nom_bd);
   end;

 if (nom_bd='') then repeter_menu_att:=false;
 if nom_bd<>'' then
    begin
       control_existance_bd;
                 if nom_bd_refuse then requete_message(3)
                  else                                 begin

Line 472  Col 49  Insert     Indent  A:ATELIER.PAS

     if nom_ent_refuse and not ent_positionne then
                          begin
                            gotoxy(37,11);readln(nom_ent);
                           end;
     if nom_ent='' then repeter_menu_att:=false;

       if nom_ent<>'' then
         begin
                  control_existance_ent;
                 if nom_ent_refuse  then requete_message(18)
                 else  begin
 textcolor(7);
 cadre(30,13,4,0,chr(201),chr(187),chr(200),chr(188),chr(205),chr(1
 textmode;
 cadre(30,18,-4,7,chr(201),chr(187),chr(200),chr(188),chr(205),chr(

        if menu_att_positionner then schema_aide;
        gotoxy(44,18);readln(nom_att);
        textmode;
        if (nom_att='') or (nom_att='H') then repeter_menu_att:=fal
        if (nom_att<>'') and (nom_att<>'H') then
             begin
                  control_existance_att;
```

```
      Line 496  Col 49  Insert     Indent  A:ATELIER.PAS
                            if nom_att_refuse then begin
                                                    requete_message(13);
                                                  end else repeter_menu_att:=
                                                                      end;
                                                                      end;
                        end;
                  end;
              end;
         end;
    end;


procedure supprimer_att;
var reponse:char;
 begin
   if not att_positionne then menu_supprimer_att;
   if (nom_bd<>'') and (nom_ent<>'')
   and (nom_att<>'') then
   begin
     textcolor(23);
     sortir_saisir;
     repeat
       gotoxy(12,22);
       writeln('il faut supprimer cet article?<o/n>:');

       Line 506  Col 1    Insert     Indent  A:ATELIER.PAS
       gotoxy(48,22);writeln('                    ');
       gotoxy(48,22);readln(reponse);
       until reponse in['n','o',char(26)];
       textmode;
       if reponse='o' then trt_supprimer_att;
    end;
end;


procedure trt_supprimer_ent;
begin
    obtenir_entite;
    dbfpath(ATT,ENT,EA);
    while dbfound do
      begin
        dbremove(ATT,ENT,EA);
        dbdelete(ATTRIBUT,ATT);
        dbnpath(ATT,ENT,EA);
       end;
    dbfpath(ASS,ENT,ORIGINE);
    while dbfound do
     begin
      dbremove(ASS,ENT,ORIGINE);
      dbfpath(ENT,ASS,-CIBLE);
```

```
      Line 544  Col 1    Insert     Indent  A:ATELIER.PAS
      if dbfound then dbremove(ASS,ENT,CIBLE);
      dbdelete(ASSOCIATION,ASS);
      obtenir_entite;
      dbnpath(ASS,ENT,ORIGINE);
     end;
  dbfpath(ASS,ENT,CIBLE);
  while dbfound do
   begin
    dbremove(ASS,ENT,CIBLE);
    dbfpath(ENT,ASS,-ORIGINE);
    if dbfound then dbremove(ASS,ENT,ORIGINE);
    dbdelete(ASSOCIATION,ASS);
    obtenir_entite;
    dbnpath(ASS,ENT,CIBLE);
   end;
  dbdelete(ENTITE,ENT);
  if not supprimer_base then requete_message(12);
 end;

procedure menu_supprimer_ent;
var  repeter_menu_ent:boolean;
  begin
    nom_ent_refuse:=true;
    nom_bd_refuse:=true;

      Line 568  Col 1    Insert     Indent  A:ATELIER.PAS
    repeter_menu_ent:=true;
    while repeter_menu_ent do
     begin
       clrscr;
       cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       cadre(11,10,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       if menu_ent_positionner then                              begin
             gotoxy(31,3);writeln('POSITIONNER=ENTITE'); end else   begir
             gotoxy(32,3);writeln('SUPPRIMER=ENTITE');              end;
       textcolor(7);
       cadre(30,20,-1,7,chr(201),chr(187),chr(200),chr(188),chr(205),chr(
       gotoxy(12,15);writeln('Le nom de la base de données:');
       if menu_ent_positionner then
         begin
             gotoxy(12,17);writeln('Nom du type d''entité à positionner/t
         end else
          begin
             gotoxy(12,17);writeln('Le nom du type d''entité à supprimer:
          end;
       sortir_saisir;
       textmode;
       if bd_positionnee then begin gotoxy(42,15);writeln(nom_bd); end
       else begin
       if nom_bd_refuse then begin gotoxy(42,15);readln(nom_bd); end
```

```
      Line 592  Col 1    Insert     Indent  A:ATELIER.PAS
       else begin gotoxy(42,15);writeln(nom_bd); end;
            end;
       textmode;
       if (nom_bd='') then repeter_menu_ent:=false;
       if (nom_bd<>'') then
         begin
            control_existance_bd;
                  if nom_bd_refuse then  requete_message(3)
         else
             begin
             if menu_ent_positionner then begin textcolor(7); schema_aide
             textmode; end;
             gotoxy(49,17);readln(nom_ent);
             if (nom_ent='') or (nom_ent='H') then repeter_menu_ent:=fals
             if (nom_ent<>'') and (nom_ent<>'H') then
                     begin
                control_existance_ent;
                if nom_ent_refuse then requete_message(10)
                 else repeter_menu_ent:=false;
                   end;
           end;
      end;
  end;
  end;

      Line 616  Col 1    Insert     Indent  A:ATELIER.PAS


procedure supprimer_ent;
var reponse:char;
 begin
     supprimer_base:=false;
     if not ent_positionne then menu_supprimer_ent;
     if (nom_bd<>'') then
     if (nom_ent<>'') then
        begin
           textcolor(23);
           sortir_saisir;
           repeat
               gotoxy(12,22);
               writeln('il faut supprimer cet article?<o/n>:');
               gotoxy(48,22);writeln('                       ');
               gotoxy(48,22);readln(reponse);
            until reponse in['n','o',char(26)];
            textmode;
            if reponse='o' then trt_supprimer_ent;
         end;
  end;
```

```
       Line 617  Col 1    Insert     Indent  A:ATELIER.PAS

procedure trt_supprimer_ass;
begin
   nom_ent:=nom_ent_or;
   obtenir_entite;
   obtenir_ass_via_origine;
   dbremove(ASS,ENT,ORIGINE);
   nom_ent:=nom_ent_cible;
   obtenir_entite;
   dbremove(ASS,ENT,CIBLE);
   dbdelete(ASSOCIATION,ASS);
   requete_message(16);
 end;


procedure menu_supprimer_ass;
var  repeter_menu_ass:boolean;
  begin
    nom_ass_refuse:=true;
    nom_bd_refuse:=true;
    repeter_menu_ass:=true;
    while repeter_menu_ass do
     begin

       Line 663  Col 1    Insert     Indent  A:ATELIER.PAS
       clrscr;
       cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       cadre(14,14,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       if menu_ass_positionner then begin
                     gotoxy(29,3);writeln('POSITIONNER=ASSOCIATION');
                                  end else begin
                     gotoxy(30,3);writeln('SUPPRIMER=ASSOCIATION');
                                         end;
       textcolor(7);
       cadre(30,25,-1,7,chr(201),chr(187),chr(200),chr(188),chr(205),chr(
       gotoxy(12,15);writeln('Le nom de la base de données:');
       if menu_ass_positionner then begin
              gotoxy(12,17);
              writeln('Nom du type d''association à positionner/H:');
                                    end else begin
              gotoxy(12,17);
              writeln('Le nom du type d''association à supprimer:');
                                         end;
       sortir_saisir;
       textmode;
       if nom_bd_refuse and not bd_positionnee then begin gotoxy(42,15);
                                                    readln(nom_bd); end
       else begin gotoxy(42,15);writeln(nom_bd); end;
       textmode;
```

```
      Line 687  Col 1    Insert     Indent  A:ATELIER.PAS
       if nom_bd='' then repeter_menu_ass:=false;
       if (nom_bd<>'') then
         begin
            control_existance_bd;
                  if nom_bd_refuse then  requete_message(3)
         else
             begin
               if menu_ass_positionner then begin textcolor(7); schema_ai
               textmode;
             end;
             gotoxy(54,17);readln(nom_ass);
             if (nom_ass='') or (nom_ass='H') then repeter_menu_ass:=fals
             if (nom_ass<>'') and (nom_ass<>'H') then
                    begin
                  control_existance_ass;
                  if nom_ass_refuse then requete_message(15)
                   else repeter_menu_ass:=false;
                    end;
           end;
      end;
  end;
  end;
```

```
      Line 710  Col 1    Insert     Indent  A:ATELIER.PAS

procedure supprimer_ass;
var reponse:char;
begin
       if not ass_positionnee then menu_supprimer_ass else control_existe
       if (nom_bd<>'') then
           if (nom_ass<>'') then
             begin
                 cadre(32,33,10,-8,chr(213),chr(184),chr(212),chr(190),ch
                 gotoxy(10,3);
                 writeln('TYPE D''ENTIE ORIGINE: ',nom_ent_or,'   TYPE D'
               textcolor(23);
               sortir_saisir;
               repeat
                 gotoxy(12,22);
                 writeln('il faut supprimer cet article?<o/n>:');
                 gotoxy(48,22);writeln('                    ');
                 gotoxy(48,22);readln(reponse);
               until reponse in['n','o',char(26)];
               textmode;
               if reponse='o' then trt_supprimer_ass;
             end;
end;
```

```
       Line 735  Col 1    Insert     Indent   A:ATELIER.PAS

procedure trt_supprimer_bd;
begin
    supprimer_base:=true;
    dbfpath(ENT,BD,BE);
    if dbfound then begin nom_ent:=ENT.NOME;
                          trt_supprimer_ent;
                    end;
    while dbfound do
      begin
        dbnpath(ENT,BD,BE);
        nom_ent:=ENT.NOME;
        trt_supprimer_ent;
      end;
      dbdelete(BASE,BD);
      requete_message(11);
      supprimer_base:=false;
 end;

procedure menu_supprimer_bd;
  begin
    nom_bd_refuse:=true;
    while nom_bd_refuse do
      begin

       Line 759  Col 1    Insert     Indent   A:ATELIER.PAS
        clrscr;
        cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr
        cadre(9,9,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr(1
        if menu_bd_positionner then                begin
        gotoxy(33,3);writeln('POSITIONNER=BD'); end else   begin
              gotoxy(34,3);writeln('SUPPRIMER=BD');         end;
        textcolor(7);
        cadre(30,26,-1,5,chr(201),chr(187),chr(200),chr(188),chr(205),chr(
        if menu_bd_positionner then
          begin
              gotoxy(12,15);writeln('Nom de la base de données à positionn
              schema_aide;
          end else
           begin
              gotoxy(12,15);writeln('Le nom de la base de données à suppri
           end;
        sortir_saisir;
        textmode;
        gotoxy(55,15);readln(nom_bd);
        if (nom_bd<>'') and (nom_bd<>'H') then
          begin
            control_existance_bd;
            if nom_bd_refuse then requete_message(3);
          end else nom_bd_refuse:=false;
```

```
      Line 783   Col 1    Insert     Indent   A:ATELIER.PAS
    end;
  end;


procedure supprimer_bd;
var reponse:char;
begin
   if not bd_positionnee then  menu_supprimer_bd;
   if nom_bd<>'' then
         begin
               textcolor(23);
               sortir_saisir;
               repeat
                 gotoxy(12,22);
                 writeln('il faut supprimer cet article?<o/n>:');
                 gotoxy(48,22);writeln('                      ');
                 gotoxy(48,22);readln(reponse);
               until reponse in['n','o',char(26)];
               textmode;
               if reponse='o' then trt_supprimer_bd;
          end;
end;


      Line 784   Col 1    Insert     Indent   A:ATELIER.PAS


procedure trt_creation_ass;
var continuer:boolean;
begin

   ASS.NOMS:=nom_ass;
   dbcreate(ASSOCIATION,ASS);
   nom_ent:=nom_ent_or;
   obtenir_entite;
   dbinsert(ASS,ENT,origine);
   nom_ent:=nom_ent_cible;
   obtenir_entite;
   dbinsert(ASS,ENT,cible);
   nom_bd_refuse:=false;
   nom_ent_or_refuse:=true;
   nom_ent_cible_refuse:=true;
   requete_message(8);
   ass_positionnee:=false;
 end;
```

```
      Line 807  Col 1    Insert     Indent  A:ATELIER.PAS

procedure menu_creation_ass;
var repeter_menu_ass:boolean;
  begin
    repeter_menu_ass:=true;
    while repeter_menu_ass do
    begin
       clrscr;
       cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       cadre(13,14,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       gotoxy(31,3);writeln('CREATION-ASSOCIATION');
       cadre(30,15,4,2,chr(201),chr(187),chr(200),chr(188),chr(205),chr(1
       textcolor(7);
       cadre(30,20,-4,7,chr(201),chr(187),chr(200),chr(188),chr(205),chr(
       gotoxy(12,9);writeln('Le nom de la base de données:');
       gotoxy(12,11);writeln('Le nom du type d''entité origine:');
       gotoxy(12,13);writeln('Le nom du type d''entité cible:');
       gotoxy(12,18);writeln('Le nom du type d''associatin à créer:');
       sortir_saisir;
       textmode;
        if not nom_bd_refuse and not nom_ent_or_refuse and
           not nom_ent_cible_refuse then
                    begin
                      gotoxy(42,9);writeln(nom_bd);

      Line 853  Col 1    Insert     Indent  A:ATELIER.PAS
                      gotoxy(47,11);writeln(nom_ent_or);
                      gotoxy(43,13);writeln(nom_ent_cible);
                     end;
        if not nom_bd_refuse and not nom_ent_or_refuse and
           nom_ent_cible_refuse then
                    begin
                      gotoxy(42,9);writeln(nom_bd);
                      gotoxy(45,11);writeln(nom_ent_or);
                     end;
        if bd_positionnee then
                    begin
                      gotoxy(42,9);writeln(nom_bd);
                     end  else
          begin
           if not nom_bd_refuse and nom_ent_or_refuse and
            nom_ent_cible_refuse then
                    begin
                      gotoxy(42,9);writeln(nom_bd);
                     end;
          end;
       if nom_bd_refuse and not bd_positionnee then
                     begin
                      gotoxy(42,9);readln(nom_bd);
                     end;
```

```
Line 877  Col 1    Insert     Indent  A:ATELIER.PAS
 if nom_bd='' then repeter_menu_ass:=false;
 if nom_bd<>'' then
    begin
       control_existance_bd;
                 if nom_bd_refuse then requete_message(3)
                  else                                  begin
     if nom_ent_or_refuse then
                                    begin
                                   gotoxy(45,11);
                                   readln(nom_ent);
                                   nom_ent_or:=nom_ent;
                                      end;
       if nom_ent_or='' then repeter_menu_ass:=false;
       if nom_ent_or<>'' then
         begin
                    nom_ent:=nom_ent_or;
                    control_existance_ent;
                    if nom_ent_refuse then nom_ent_or_refuse:=true
                       else nom_ent_or_refuse:=false;
                   if nom_ent_or_refuse then requete_message(18)
                                                        else begin
          if nom_ent_cible_refuse then
                                    begin
                                   gotoxy(43,13);

Line 901  Col 1    Insert     Indent  A:ATELIER.PAS
                                   readln(nom_ent);
                                   nom_ent_cible:=nom_ent;
                                      end;
       if nom_ent_cible='' then repeter_menu_ass:=false;
       if nom_ent_cible<>'' then
         begin
            if nom_ent_cible=nom_ent_or then requete_message(10)
            else begin
                    nom_ent:=nom_ent_cible;
                    control_existance_ent;
                    if nom_ent_refuse then nom_ent_cible_refuse:=true
                       else nom_ent_cible_refuse:=false;
                   if nom_ent_cible_refuse then requete_message(18)
                       else begin
 textcolor(7);
 cadre(30,15,4,2,chr(201),chr(187),chr(200),chr(188),chr(205),chr(1
 textmode;
 cadre(30,20,-4,7,chr(201),chr(187),chr(200),chr(188),chr(205),chr(
 gotoxy(50,18);readln(nom_ass);
 repeter_menu_ass:=false;
        if nom_ass<>'' then
              begin
                control_non_existance_ass;
                if nom_ass_refuse then begin
```

```
      Line 925  Col 1    Insert     Indent  A:ATELIER.PAS
                                                  requete_message(9);
                                                  repeter_menu_ass:=true;
                                                  end;
                        end;
                                    end;
                                end;
                            end;
                        end;
                end;
            end;
        end;
    end;

end;

procedure creation_ass;
var repeter_creation_ass:boolean;
begin
        nom_ass_refuse:=true;
        nom_ent_refuse:=true;
        nom_bd_refuse:=true;
        nom_ent_or_refuse:=true;
        nom_ent_cible_refuse:=true;
    repeter_creation_ass:=true;

       Line 939  Col 1    Insert     Indent  A:ATELIER.PAS
    while repeter_creation_ass do
      begin
        menu_creation_ass;
        if (nom_bd<>'') then
         if (nom_ent_or<>'') then
          if (nom_ent_cible<>'') then
            if (nom_ass<>'') then trt_creation_ass;
        if (nom_ass='') or (nom_ent_or='') or (nom_ent_cible='')
        or (nom_bd='') then repeter_creation_ass:=false;
      end;
end;



procedure trt_creation_att;
var continuer:boolean;
begin
    ATT.NOMA:=nom_att;
    ATT.ETAT:=etat_att;
    dbcreate(ATTRIBUT,ATT);
    obtenir_entite;
    dbinsert(ATT,ENT,EA);
    requete_message(5);
    att_positionne:=false;

       Line 951  Col 1    Insert     Indent  A:ATELIER.PAS
 end;
```

```
       Line 974  Col 1    Insert     Indent  A:ATELIER.PAS

procedure menu_creation_att;
var repeter_menu_att:boolean;
  begin
    repeter_menu_att:=true;
    etat_att_refuse:=true;
    while repeter_menu_att do
    begin
       clrscr;
       cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       cadre(11,11,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       gotoxy(32,3);writeln('CREATION=ATTRIBUT');
       cadre(30,15,4,0,chr(201),chr(187),chr(200),chr(188),chr(205),chr(1
       textcolor(7);
       cadre(30,20,-2,7,chr(201),chr(187),chr(200),chr(188),chr(205),chr(
       gotoxy(12,9);writeln('Le nom de la base de données:');
       gotoxy(12,11);writeln('Le nom du type d''entité:');
       gotoxy(12,16);writeln('Le nom d''attribut à créer:');
       gotoxy(12,18);writeln('Il est identifiant ? (o/n):');
       sortir_saisir;
       textmode;
        if (creation_attribut_direct and not nom_ent_refuse)
        or (not creation_attribut_direct)
        or ent_positionne then

      Line 1020 Col 1    Insert     Indent  A:ATELIER.PAS
                    begin
                      gotoxy(42,9);writeln(nom_bd);
                      gotoxy(37,11);writeln(nom_ent);
                     end;
      if (creation_attribut_direct and not nom_bd_refuse and nom_ent_refus
      or (bd_positionnee and not ent_positionne)  then
                               begin
                                 gotoxy(42,9);
                                 writeln(nom_bd);
                               end;
       if (creation_attribut_direct and not bd_positionnee and premiere_p
       or (nom_bd_refuse and not premiere_passage_bd and not bd_positionr
                                    then begin
                                           gotoxy(42,9);
                                           readln(nom_bd);
                                         end;

       if (nom_bd='') then repeter_menu_att:=false;
       if nom_bd<>'' then
          begin
             control_existance_bd;
                      if nom_bd_refuse then begin
                                              requete_message(3);
                                            end
```

```
Line 1044 Col 1    Insert     Indent  A:ATELIER.PAS
                   else                                  begin
      if not ent_positionne
      and ((creation_attribut_direct and premiere_passage_ent)or (no
       then
                                    begin
                                   gotoxy(37,11);readln(nom_ent);
                                     end;
        if nom_ent='' then repeter_menu_att:=false;

        if nom_ent<>'' then
          begin
                    control_existance_ent;
                   if nom_ent_refuse  then
                              begin
                                  requete_message(18);
                               end                                  else begi
 textcolor(7);
 cadre(30,15,4,0,chr(201),chr(187),chr(200),chr(188),chr(205),chr(1
 textmode;
 cadre(30,20,-2,7,chr(201),chr(187),chr(200),chr(188),chr(205),chr(
           gotoxy(39,16);readln(nom_att);

                                    textcolor(23);
                                    sortir_saisir;

Line 1068 Col 1    Insert     Indent  A:ATELIER.PAS

             repeat
                gotoxy(40,18);writeln('                         ');
                gotoxy(40,18);readln(etat_att);
                textmode;
                if (nom_att='') then etat_att:='n';
              until etat_att in['n','o',char(26)];
          if nom_att='' then repeter_menu_att:=false;
          if etat_att='o' then
                           begin
                             control_non_existance_id;
                             if etat_att_refuse then requete_message(
                           end;
          if (etat_att<>'o') or ((etat_att='o') and (not etat_att_ref
          then    begin
          if etat_att=char(26) then creation_attribut_direct:=false;
          if (nom_att<>'') and (etat_att<>char(26)) then
              begin
                   control_non_existance_att;
                   if nom_att_refuse then begin
                                            requete_message(7);
                                          end else repeter_menu_att:=
                                                              end;
                                                              end;
```

```
      Line 1092 Col 1    Insert     Indent  A:ATELIER.PAS
                                    end;
                                 end;
                              end;
                              premiere_passage_ent:=false;
                        end;
                         premiere_passage_bd:=false;
      end;

  end;


procedure creation_att;
var repeter_creation_att:boolean;
 begin
        premiere_passage_bd:=true;
        premiere_passage_ent:=true;
        nom_att_refuse:=true;
        nom_ent_refuse:=true;
        nom_bd_refuse:=true;
        repeter_creation_att:=true;
        while repeter_creation_att do
          begin
            menu_creation_att;
            if (nom_bd<>'') and (nom_ent<>'') and (nom_att<>'') then

      Line 1112 Col 1    Insert     Indent  A:ATELIER.PAS
            if (nom_bd<>'') and (nom_ent<>'') and (nom_att<>'') then
                                                    trt_creation_att;
            if (nom_bd='') or (nom_ent='') or (nom_att='')
                           then repeter_creation_att:=false;
            creation_entite_direct:=false;
          end;
end;


procedure trt_creation_ent;
begin
   ENT.NOME:=nom_ent;
   dbcreate(ENTITE,ENT);
   dbinsert(ENT,BD,BE);
   requete_message(2);
   ent_positionne:=false;
 end;

procedure menu_creation_ent;
var  repeter_menu_ent:boolean;
  begin
    nom_ent_refuse:=true;
    nom_bd_refuse:=true;
    repeter_menu_ent:=true;
```

```
  Line 1139 Col 1    Insert     Indent  A:ATELIER.PAS
    while repeter_menu_ent do
     begin
       clrscr;
       cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       cadre(10,8,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr(
       gotoxy(32,3);writeln('CREATION=ENTITE');
       textcolor(7);
       cadre(30,20,-1,7,chr(201),chr(187),chr(200),chr(188),chr(205),chr(
       gotoxy(12,15);writeln('Le nom de la base de données:');
       gotoxy(12,17);writeln('Le nom du type d''entité à créer:');
       sortir_saisir;
       textmode;
       if bd_positionnee then begin gotoxy(42,15);writeln(nom_bd); end
       else begin
       if creation_entite_direct then begin gotoxy(42,15);readln(nom_bd);
       else begin gotoxy(42,15);writeln(nom_bd); end;
            end;
       textmode;
       if nom_bd='' then repeter_menu_ent:=false;
       if (nom_bd<>'') then
         begin
            control_existance_bd;
                  if nom_bd_refuse then begin
                                        requete_message(3);

  Line 1163 Col 1    Insert     Indent  A:ATELIER.PAS
                                        creation_entite_direct:=true;
                                        end else
              begin
              gotoxy(45,17);readln(nom_ent);
              if nom_ent='' then repeter_menu_ent:=false;
              if nom_ent<>'' then
                    begin
                 control_non_existance_ent;
                 if nom_ent_refuse then   begin
                                             requete_message(4);
                                             creation_entite_direct:=fa
                                          end else repeter_menu_ent:=f
                    end;
            end;
         end;
   end;
   end;


procedure creation_ent;
var repeter_creation_ent:boolean;
 begin
    repeter_creation_ent:=true;
    while repeter_creation_ent do
```

```
       Line 1187 Col 1    Insert     Indent  A:ATELIER.PAS
      begin
        menu_creation_ent;
        BD.NOMB:=nom_bd;
        if (BD.NOMB<>'') then
         if (nom_ent<>'') then
          begin
            trt_creation_ent;
            creation_attribut_direct:=false;
            creation_att;
          end;
        if (nom_ent='') or (nom_bd='') then repeter_creation_ent:=false;
      end;
end;


procedure trt_creation_bd;
begin
   dbcreate(BASE,BD);
   requete_message(1);
   bd_positionnee:=false;
 end;

procedure menu_creation_bd;
  begin

       Line 1211 Col 1    Insert     Indent  A:ATELIER.PAS
     nom_bd_refuse:=true;
     while nom_bd_refuse do
      begin
        clrscr;
        cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr
        cadre(9,7,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr(1
        gotoxy(34,3);writeln('CREATION=BD');
        textcolor(7);
        cadre(30,20,-1,5,chr(201),chr(187),chr(200),chr(188),chr(205),chr(
        gotoxy(12,15);writeln('Le nom de la base de données à créer:');
        sortir_saisir;
        textmode;
        gotoxy(50,15);readln(nom_bd);
        if nom_bd<>'' then
          begin
            BD.NOMB:=nom_bd;
            control_non_existance_bd;
            if nom_bd_refuse then requete_message(0);
          end else nom_bd_refuse:=false;
      end;
  end;
```

```
       Line 1226 Col 1    Insert     Indent   A:ATELIER.PAS

procedure creation_bd;
begin
    menu_creation_bd;
    if nom_bd<>'' then
    begin
      BD.NOMB:=nom_bd;
      trt_creation_bd;
      creation_entite_direct:=false;
      creation_ent;
    end;
end;



procedure affichage_bd;
begin
  efface_schema_aide;
  cadre(-14,26,6,-1,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179)
  window(55,7,65,10);
  dbfirst(BASE,BD);
  while dbfound do
    begin
      writeln(BD.NOMB);

       Line 1258 Col 1    Insert     Indent   A:ATELIER.PAS
      dbnext(BASE,BD);
      delay(1000);
    end;
    normvideo;
    readln(x);
    window(1,1,80,40);
end;

procedure positionner_bd;
var repeter_menu:boolean;
begin
  repeter_menu:=true;
  while repeter_menu do
    begin
      repeter_menu:=false;
      bd_positionnee:=false;
      ent_positionne:=false;
      att_positionne:=false;
      ass_positionnee:=false;
      menu_bd_positionner:=true;
      menu_supprimer_bd;
      if (nom_bd<>'') and (nom_bd<>'H') then begin
                                              bd_positionnee:=true;
                                              ent_positionne:=false;
```

```
      Line 1282 Col 1    Insert     Indent   A:ATELIER.PAS
                                                  att_positionne:=false;
                                                  ass_positionnee:=false;
                                                 end;
      menu_bd_positionner:=false;
      if nom_bd='H' then begin
                           affichage_bd;
                           repeter_menu:=true;
                          end;
    end;
end;
procedure affichage_ent;
begin
  efface_schema_aide;
  cadre(-14,26,6,-1,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179)
  window(55,7,65,10);
  BD.NOMB:=nom_bd;
  dbid(BASE,BD);
  dbfpath(ENT,BD,BE);
  while dbfound do
    begin
      writeln(ENT.NOME);
      dbnpath(ENT,BD,BE);
      delay(1000);
    end;

      Line 1306 Col 1    Insert     Indent   A:ATELIER.PAS
    normvideo;
    readln(x);
    window(1,1,80,40);
end;


procedure positionner_ent;
var repeter_menu:boolean;
begin
 repeter_menu:=true;
 while  repeter_menu do
  begin
    repeter_menu:=false;
    menu_ent_positionner:=true;
    ent_positionne:=false;
    menu_supprimer_ent;
    if (nom_ent<>'') and (nom_ent<>'H') and (nom_bd<>'') then
      begin
       ent_positionne:=true;
       bd_positionnee:=true;
       att_positionne:=false;
       ass_positionnee:=false;
      end;
    if nom_bd<>'' then bd_positionnee:=true;
```

```
      Line 1330 Col 1    Insert     Indent  A:ATELIER.PAS
    menu_ent_positionner:=false;
    if nom_ent='H' then
       begin
          repeter_menu:=true;
          affichage_ent;
       end;
  end;
end;

procedure affichage_att;
begin
  efface_schema_aide;
  cadre(-19,31,9,-4,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179)
  window(60,4,70,7);
  obtenir_entite;
  dbfpath(ATT,ENT,EA);
  while dbfound do
    begin
      writeln(ATT.NOMA);
      dbnpath(ATT,ENT,EA);
      delay(1000);
    end;
    normvideo;
    readln(x);

      Line 1354 Col 1    Insert     Indent  A:ATELIER.PAS
    window(1,1,80,40);
end;


procedure positionner_att;
var repeter_menu:boolean;
begin
repeter_menu:=true;
  while repeter_menu do
  begin
    repeter_menu:=false;
    menu_att_positionner:=true;
    att_positionne:=false;
    menu_supprimer_att;
    if (nom_bd<>'') and (nom_ent<>'') and (nom_att<>'') and (nom_att<>'H'
    then begin
           att_positionne:=true;
           ent_positionne:=true;
           bd_positionnee:=true;
           end;

   if nom_bd<>'' then
                begin
                   bd_positionnee:=true;
```

```
        Line 1378 Col 1    Insert     Indent  A:ATELIER.PAS
                        ent_positionne:=false;
                      end;
      if (nom_ent<>'') and (nom_bd<>'') then
        begin
         ent_positionne:=true;
         bd_positionnee:=true;
        end;

      menu_att_positionner:=false;
      if (nom_bd='') or (nom_ent='') then nom_att:='';
      if nom_att='H' then
         begin
            repeter_menu:=true;
            affichage_att;
         end;
    end;
end;

procedure affichage_ass;
begin
   efface_schema_aide;
   cadre(-19,31,9,-4,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179)
   window(60,4,70,7);
   BD.NOMB:=nom_bd;

        Line 1402 Col 1    Insert     Indent  A:ATELIER.PAS
   dbid(BASE,BD);
   dbfpath(ENT,BD,BE);
   while dbfound do
     begin
       dbfpath(ASS,ENT,ORIGINE);
       while dbfound do
         begin
           writeln(ASS.NOMS);
           dbnpath(ASS,ENT,ORIGINE);
         end;
       dbnpath(ENT,BD,BE);
       delay(1000);
     end;
     normvideo;
     readln(x);
     window(1,1,80,40);
end;



procedure positionner_ass;
var repeter_menu:boolean;
begin
   repeter_menu:=true;
```

```
      Line 1403 Col 1    Insert     Indent   A:ATELIER.PAS
  repeter_menu:=true;
  while  repeter_menu do
  begin
    repeter_menu:=false;
    menu_ass_positionner:=true;
    ass_positionnee:=false;
    menu_supprimer_ass;
    if (nom_bd<>'') and (nom_ass<>'') and (nom_ass<>'H')
    then begin
           ass_positionnee:=true;
           bd_positionnee:=true;
         end;
    if nom_bd<>'' then bd_positionnee:=true;
    menu_ass_positionner:=false;
   if nom_ass='H' then
       begin
          repeter_menu:=true;
          affichage_ass;
       end;
  end;
end;
```

```
      Line 1426 Col 1    Insert     Indent   A:ATELIER.PAS

procedure trt_traduction_schema;
var chercher:boolean;
    nom_att_id:chaine;
begin
   clrscr;
   textcolor(7);
   BD.NOMB:=nom_bd;
   dbid(BASE,BD);
   dbfpath(ENT,BD,BE);
   while dbfound do
     begin
       y:=0;
       clrscr;
        cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),ch
        cadre(12,12,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),ch
        gotoxy(32,3);writeln('TRADUCTION=SCHEMA');
       gotoxy(12,6);writeln('Fichier dBase.........',ENT.NOME);
       gotoxy(12,7);writeln('Champs (Fields):');
       dbfpath(ATT,ENT,EA);
       while dbfound do
         begin
           y:=y+1;
           gotoxy(14,7+y);writeln('... ',ATT.NOMA);
```

```
Line 1472 Col 1    Insert     Indent   A:ATELIER.PAS
          dbnpath(ATT,ENT,EA);
        end;
      nom_ent:=ENT.NOME;
      dbfpath(ASS,ENT,CIBLE);
        while dbfound do
          begin
            dbfpath(ENT,ASS,-ORIGINE);
            if dbfound then
              begin
                  nom_att_id:='pas_encore';
                  chercher:=true;
                  dbfpath(ATT,ENT,EA);
                  while dbfound do
                      begin
                       if ATT.ETAT='o' then
                              begin
                                chercher:=false;
                                nom_att_id:=ATT.NOMA;
                              end;
                        dbnpath(ATT,ENT,EA);
                      end;
                  if chercher then
                                    begin
                                      gotoxy(12,20);writeln('Il faut un i

Line 1496 Col 1    Insert     Indent   A:ATELIER.PAS
                                     end else
                                       begin
                                          y:=y+1;
                                          gotoxy(12,7+y);writeln('att.étran
                                          gotoxy(12,8+y);writeln('Contraint
    gotoxy(12,9+y);writeln(nom_att_id,'e (:',nom_ent,') in ',nom_att_id,'
                                       end;
            end;
          obtenir_entite;
          dbnpath(ASS,ENT,CIBLE);
       end;
       dbnpath(ENT,BD,BE);
       sortir_saisir;
       readln(x);
    end;
    textmode;
    requete_message(26);
end;


procedure  menu_traduction_schema;
  begin
    nom_bd_refuse:=true;
```

```
  Line 1497 Col 1    Insert     Indent  A:ATELIER.PAS
    nom_bd_refuse:=true;
    while nom_bd_refuse do
     begin
       clrscr;
       cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       cadre(12,12,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       gotoxy(32,3);writeln('TRADUCTION=SCHEMA');
       textcolor(7);
       cadre(30,26,-1,5,chr(201),chr(187),chr(200),chr(188),chr(205),chr(
       gotoxy(12,15);writeln('Le nom de la base de données à traduire:');
       sortir_saisir;
       textmode;
       gotoxy(54,15);readln(nom_bd);
       if nom_bd<>'' then
         begin
           control_existance_bd;
           if nom_bd_refuse then requete_message(3);
         end else nom_bd_refuse:=false;
     end;
  end;
```

```
  Line 1520 Col 1    Insert     Indent  A:ATELIER.PAS

procedure traduction_schema;
begin
  if not bd_positionnee then  menu_traduction_schema;
  if nom_bd<>'' then trt_traduction_schema;
end;
```

```
      Line 1592 Col 1    Insert     Indent  A:ATELIER.PAS

                    (*  DEBUT  MODULE      RAPPORT     *)

procedure menu_rapport;
var repeter_menu_rapport_bd:boolean;
begin
    repeter_menu_rapport_bd:=true;
    while  repeter_menu_rapport_bd do
     begin
       clrscr;
       cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr
       cadre(9,8,10,-9,chr(213),chr(184),chr(212),chr(190),chr(205),chr(1
       gotoxy(35,3);writeln('*RAPPORT*');
       textcolor(7);
       cadre(30,21,-1,7,chr(201),chr(187),chr(200),chr(188),chr(205),chr(
       gotoxy(12,15);writeln('Le nom de la base de données:');
       gotoxy(12,17);writeln('Faut-t-il imprimmer le rapport?(o/n):');
       sortir_saisir;
       textmode;
       if bd_positionnee then begin gotoxy(42,15); writeln(nom_bd);
                                end else begin
                                          gotoxy(42,15);readln(nom_bd);
                                         end;
       if nom_bd<>'' then

      Line 1616 Col 1    Insert     Indent  A:ATELIER.PAS
         begin
           control_existance_bd;
           if nom_bd_refuse then requete_message(3)
                  else    begin
                            repeter_menu_rapport_bd:=false;

                                    textcolor(23);
                                    sortir_saisir;

                            repeat
                                gotoxy(50,17);writeln('      ');
                                gotoxy(50,17);readln(imprime);
                                textmode;
                            until imprime in['n','o',chr(26)];
                            if imprime=chr(26) then
                            repeter_menu_rapport_bd:=false;
                           end;

                 textmode;

         end      else   repeter_menu_rapport_bd:=false;
    end;
end;
```

```
       Line 1639 Col 1    Insert     Indent  A:ATELIER.PAS

procedure reste_att;
begin
writeln(ATT.NOMA);
delay(2000);
end;

procedure schema_ass;
var i,j:integer;
begin
   clrscr;
   gotoxy(30,3);writeln('RAPPORT SUR ',nom_bd);
   cadre(12,11,10,-8,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179
   cadre(39,39,11,12,chr(201),chr(187),chr(200),chr(188),chr(205),chr(186
   textcolor(7);
   gotoxy(12,3);writeln('TE origine');
   gotoxy(60,3);writeln('TE cible');
   for i:=0 to 2 do
   begin
     cadre(30,-18,7-i*6,-5+i*6,chr(213),chr(184),chr(212),chr(190),chr(20
     cadre(30,-18,7-i*6,-3+i*6,chr(213),chr(184),chr(212),chr(190),chr(20
     cadre(-18,30,7-i*6,-5+i*6,chr(213),chr(184),chr(212),chr(190),chr(20
     cadre(-18,30,7-i*6,-3+i*6,chr(213),chr(184),chr(212),chr(190),chr(20


       Line 1663 Col 1    Insert     Indent  A:ATELIER.PAS
     cadre(8,8,6-i*6,-4+i*6,chr(213),chr(184),chr(212),chr(190),chr(205),
     for j:=23 to 31 do begin
                          gotoxy(i,7+i*6);writeln('-');
                          gotoxy(j+26,7+i*6);writeln('-');
                        end;
   end;
end;

procedure rapport;
var lignes,ligne,colonne,nb_att,toure:integer;
begin
 menu_rapport;
 if (nom_bd<>'') and (imprime<>chr(26)) then
 begin
  ligne:=5;
  BD.NOMB:=nom_bd;
  dbid(BASE,BD);
  clrscr;
  colonne:=0;toure:=0;
  gotoxy(30,3);writeln('RAPPORT SUR ',nom_bd);
  cadre(12,11,10,-8,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179
  dbfpath(ENT,BD,BE);
  while dbfound do
    begin
```

```
Line 1687 Col 1    Insert     Indent  A:ATELIER.PAS
 if toure=0 then begin
 gotoxy(30,3);writeln('RAPPORT SUR ',nom_bd);
 cadre(12,11,10,-8,chr(213),chr(184),chr(212),chr(190),chr(205),chr
 cadre(39,39,11,12,chr(201),chr(187),chr(200),chr(188),chr(205),chr
                  end;
                   lignes:=0;
                   dbfpath(ATT,ENT,EA);
                     while dbfound do
                      begin
                         lignes:=lignes+1;
                         dbnpath(ATT,ENT,EA);
                      end;
textcolor(7);
if lignes<11 then ligne:=lignes else ligne:=10;
cadre(35-colonne,-21+colonne,3,-1,chr(213),chr(184),chr(212),chr(19
cadre(35-colonne,-21+colonne,3,ligne,chr(213),chr(184),chr(212),chr
gotoxy(7+colonne,10);writeln(ENT.NOME);
dbfpath(ATT,ENT,EA);
nb_att:=0;
while dbfound do
  begin
      nb_att:=nb_att+1;
      if nb_att<11 then
         begin

Line 1711 Col 1    Insert     Indent  A:ATELIER.PAS
            gotoxy(6+colonne,nb_att+11); if(ATT.ETAT='o') then writel
            gotoxy(7+colonne,nb_att+11);writeln(ATT.NOMA);
        end else begin
                   if nb_att=11 then begin read(nom_ass); clrscr;end
                   reste_att;
                 end;
     dbnpath(ATT,ENT,EA);

  end;
  colonne:=colonne+19;
  toure:=toure+1;
  if toure=4 then begin
                    toure:=0; colonne:=0;
                    requete_message(21);
                    clrscr;
                    end;
 dbnpath(ENT,BD,BE);
 textmode;
end;
cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179)
requete_message(22);

toure:=0;
dbfirst(ASSOCIATION,ASS);
```

```
      Line 1735 Col 1    Insert     Indent   A:ATELIER.PAS
  while dbfound do
   begin
    dbfpath(ENT,ASS,-origine);
    if dbfound then
       begin
        dbfpath(BD,ENT,-BE);
        if dbfound and (BD.NOMB=nom_bd) then
           begin
            if toure=0 then schema_ass;
            gotoxy(12,6+toure*6);writeln(ENT.NOME);
            gotoxy(34,7+toure*6);writeln(ASS.NOMS);
            dbfpath(ENT,ASS,-cible);
            if dbfound then  begin
                gotoxy(60,6+toure*6);
                writeln(ENT.NOME);
                                end;
            toure:=toure+1;
           end;
       end;
  if toure=3 then begin
       toure:=0;
       requete_message(21);
       end;
   dbnext(ASSOCIATION,ASS);

      Line 1759 Col 1    Insert     Indent   A:ATELIER.PAS
 end;
 end;
 cadre(39,39,11,12,chr(213),chr(184),chr(212),chr(190),chr(205),chr(179))
 requete_message(20);
end;

            (*     FIN                RAPPORT                          *)

                    (*    PROGRAMME PRINCIPAL     *)
begin
bd_positionnee:=false;
ent_positionne:=false;
att_positionne:=false;
ass_positionnee:=false;
menu_bd_positionner:=false;
menu_ent_positionner:=false;
menu_att_positionner:=false;
menu_ass_positionner:=false;
dbopen('metabd');
retour_menu_principal:=true;
en_tete;
while retour_menu_principal do
  begin
       repeat
```

```
Line 1783 Col 1    Insert     Indent  A:ATELIER.PAS
            control_menu_principal:=false;
            menu_principal;
            if no_requete_menu in['G','R','T','C','Q']
                then control_menu_principal:=true;
            if not control_menu_principal  then requete_message(17);
     until no_requete_menu in['G','R','T','C','Q'];
     if no_requete_menu='Q' then retour_menu_principal:=false;
     retour_sous_menu:=true;
     while retour_sous_menu do
     begin
      requete_menu;
      case num_option of
        'Z':begin
            rapport;
            retour_sous_menu:=false;
              end;
        'B':creation_bd;
        'T':traduction_schema;
        'A':creation_att;
        'C':creation_ass;
        'S':supprimer_bd;
        'U':supprimer_ent;
        'M':supprimer_ass;
        'E':begin

Line 1803 Col 1    Insert     Indent  A:ATELIER.PAS
        'E':begin
              if mbd then creation_ent
              else ecrire_programme;
            end;
         'P':begin
              if mbd then supprimer_att
              else traduction_programme;
            end;
         'G':positionner_bd;
         'H':positionner_ent;
         'I':positionner_att;
         'J':positionner_ass;
         'D':begin commentaire; retour_sous_menu:=false; end;
         'R':retour_sous_menu:=false;
       end;
      end;
  end;
  clrscr;
  dbclose;
end.
```