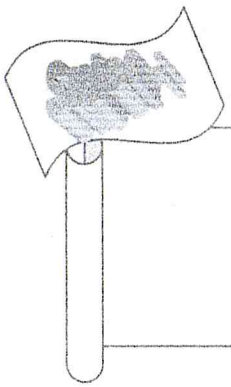
République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université « SAAD DAHLEB» de Blida

Faculté des sciences
Département de l'informatique

Mémoire du projet de fin d'étude
pour l'obtention du diplôme d'ingénieur d'état

Spécialité : informatique
Option : Système d'information avancé



Système flexible de gestion et d'aide a la décision

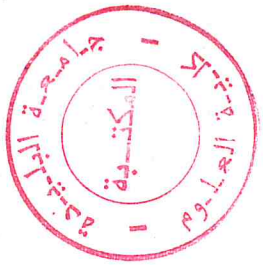
- proposé et encadré par : Mr A. Melaab
- Etudié par : **M^eLLe Refik Nadia**
  M r Hantabli Hamza

- Promotrice : Mme Oukid

- Organisme d'accueil : Caisse National de la Mutualité agricole (C.N.M.A) d'Alger

Soutenu le 1er octobre 2003 devant les jurés :
Mlle Benbldia : Présidente.
Mr Mahieddine : Examinateur.
Mme Benstiti : Examinatrice.

Promotion : 2002-2003



بسم الله الرحمن الرحيم

الحمد لله الذي أفاض على عباده النعمة ، و كتب على نفسه الرحمة ،و أشهد أن لا إله إلا الله عليه توكلت و عليه أنيب ، لا غنى لأحد عن فضله و رحمته ، و لا طمع في الفوز بجنته إلا بعفوه و مغفرته ، و أشهد أن سيدنا محمدا عبده و رسوله أرسله رحمة للعالمين ، و قدوة للعاملين ، و محجة للسالكين ، و حجة على العباد أجمعين ، بعثه للإيمان مناديا ، و إلى دار السلام داعيا ، و للخليقة هاديا ، و لكتابه تاليا و مبينا ، و في مرضاته ساعيا ، و بالمعروف آمرا و المنكر ناهيا ، أرسله على حين فترة من الرسل فهدى به إلى أوضح السبل ، و افترض طاعته و محبته ، و سد إلى الجنة كل طريق إلا طريقه ، و فهي موصدة إلا على من كانوا له تابعين .و دعا إلى الله سرا و جهارا و آذن بذلك بين أظهر الأمة ليلا و نهارا إلى أن أشرقت شمس الإيمان و علت كلمة الرحمن ، و بطلت دعوة الشيطان و اهتدى كل حيران .فصلوات الله و تسليماته عليه و على آله أصحاب الصراط السوي ، ومن اهتدى .

Remerciements

Tous d'abord nous remercions le bon ***Dieu*** pour nous avoir guidés vers le bon chemin de la lumière et de savoir, pour nous avoir donner du courage et de la volonté afin de pouvoir réaliser ce mémoire.

Nous exprimons nos sincères remerciements à nos ***parents*** pour tout ce qu`ils ont pu nous apporter.

Nous tenons à remercie les membres du jury pour avoir eu l`obligeance d'accepter et d`apprécier notre travail.

Nous remercions vivement notre encadreur ***Mr A.Melaab*** qui nous a suivi, guidé et conseillé avec enthousiasme tout au long de cette année.

Nous tenons également à exprimer notre gratitude à notre promotrice ***MeOukid*** pour sa préoccupation et ces remarques pertinentes concernant le travail.

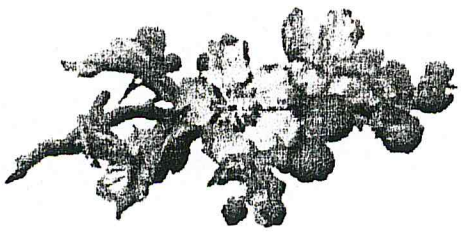
Nous remercions ceux qui ont bien voulu nous accueillir au sein de la **C.N.M.A** d`Alger et surtout le personnel de la **D.T.O.I** .

A tous les enseignants de la faculté des sciences exacts de Blida et surtout les enseignants du département de l`informatique.

Un grand merci aussi a **Mr F. Rabahi** pour sa disponibilité et son aide.

A Chikhaoui, Fouad, Mr Sellal, Mr Samir, Idir, Nafissa et Mme Dahmani.

Enfin, tous ceux qui nous ont aidé de prés ou de loin à la réalisation de ce mémoire.



Dédicaces

Je dédie mon travail à :

Mon cher papa et ma chère maman à qui je souhaite une longue vie.

Mes frères ***Samir*** et ***Mohamed***.

Mes soeurs ***Samia***, ***Lynda*** et ***warda.***

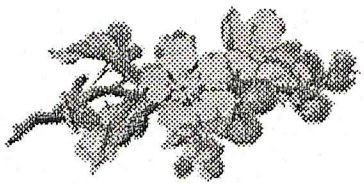
Toute ma famille.

Tous mes amis sans exception.

Et surtout à mon binôme ***Hamza.***

Tous ceux qui m`ont aider de près ou de loin.

Nadia



Je dédie mon travail à :

Mon cher papa et ma chère maman à qui je souhaite une longue vie.

Mes frères **Sid Ali ,Mohamed** et **Abderrezak** .

Mes sœurs.

Toute ma famille.

Tous mes amis sans exception.

Et surtout à mon binôme *Nadia*.

Tous ceux qui m`ont aider de près ou de loin.

Hamza

TABLE DES MATIERES

LISTE DES TABLEAUX

CHAPITRE III :

LISTE DES FIGURES

CHAPITRE III :

CHAPITRE IV :

Préface

Chaque matin, en Afrique, une gazelle se réveille,

elle sait qu'elle devra courir plus vite que le lion ;

ou elle se fera tuer.

Chaque matin, en Afrique, un lion se réveille ;

Il sait qu'il devra courir plus vite que la gazelle ;

ou il mourra de faim .

Quand le soleil se lève,

L'important n'est pas d'être un lion ou une gazelle.

Le mieux :

C'est de commencer à courir.

Savoir n'est pas assez, il faut agir.

Avoir de bonnes dispositions n'est pas assez ;

Il faut les mètres aux pratiques.

Bruce Lee.

Système flexible de gestion et d'aide à la décision

Résumé. L'introduction d'une nouvelle règle de gestion, remet nécessairement en cause le système d'information. De ce fait, le changement doit être répercuter sur le logiciel informatique, ce qui implicitement induit des modifications dans les sources du logiciel. Ce qui n'est pas toujours possible. D'où la nécessité d'un système qui aide le programmeur à développer le nouveau logiciel, avec le moins de temps possible. Pour cela, nous avons choisi une stratégie simple consiste à interfacer un systèmes de gestion de bases de données relationnel avec un système expert afin de permettre de tenir compte en particulier des 'contraints dynamiques'.

Le système a été conçu par UML et réalisé sous Delphi.

Mots clés. Système de gestion de base de données, Système expert, UML, modélisation orienté objet.

System flexible from management and aid of decision

Abstract. The introduction of a new management rule, make the user to built a new system of information. Therefore, he must remake the software, and change the program, but it's not already possible. So, it's necessary to have a system who aid it programmer at develop a new software, with less of time. For this reason, we have choose a simple strategy consist at combine a data base system management with a expert system, which in particular to consider a dynamic constraints.

The system we have developed, is designed with UML, and implemented with Delphi.

Key words. Data base system management, expert system, UML, modeling oriented object.

Introduction générale

Les entreprises et les établissements doivent évoluer constamment dans un environnement très changeant et un marché très concurrentiel, ce qui nécessite de faire évoluer rapidement les règles de gestion de l'entreprise. La refonte du système d'information (SI) n'exclut pas le risque de rejet du logiciel correspondant. Cette situation exige la conduite d'études et d'actions de promotions particulières pour maîtriser les changements, car si le système ne répond pas aux attentes de l'utilisateur, celui-ci sera tenté de créer un système parallèle, plus adapté à ses besoins.

La Caisse Nationale de la Mutualité Agricole (CNMA) est une institution professionnelle, chargée de réaliser, au profit des agriculteurs et professions assimilées, la couverture de l'ensemble des risques qui peuvent les menacer dans leur vie. La CNMA dispose d'une direction technique d'organisation informatique (D.T.O.I), son but est de faciliter et d'automatiser les tâches de la CNMA.

La Caisse Nationale de la Mutualité Agricole (CNMA), nous a confié ce projet qui consiste à réaliser un système flexible de gestion et d'aide à la décision. C'est un système qui pourra évoluer afin de s'adapter aux changements de règles de gestion et aux nouveaux besoins des utilisateurs.

Pour cela, nous avons choisi une stratégie simple qui consiste à interfacer un systèmes de gestion de bases de données relationnel (SGBDR) avec un système expert (SE).

Ce choix est basé sur le fait que les SGBDRs sont devenus l'outil de base de l'informatisation des entreprises. Ces systèmes garantissent plusieurs fonctionnalités pour la gestion des données. Mais l'insuffisance de leur capacité de représentation sémantique les rendant en général peu efficaces pour la manipulation de connaissances. A l'inverse, les systèmes experts n'offrent pas toujours des capacités suffisantes de gestion de données. Ils fournissent par contre des mécanismes de représentation des connaissances et surtout d'aide à la décision.

Un système d'information (SI) a toujours un cycle de développement, il n'est jamais définitivement figé, son déroulement peut être perturbé par l'évolution de la réglementation externe et le changement imprévu des règles de gestion internes. L'expression de nouveaux besoins relative aux utilisateurs ou aux nécessités techniques est un phénomène fréquent. C'est pourquoi, dans toute

entreprise qu'elle soit publique ou privé, les grands programmes de modernisation ou les grands changements structurels sont aujourd'hui accompagnés par une refonte complète des SI. Il s'agit alors de refaire complètement le logiciel informatique ce qui implique une grande perte du temps et d'argent, ou de modifier la source du logiciel existant (si elle est disponible), ce qui reste une tâche fastidieuse (source non compréhensive, incohérence d'instructions,...). D'où la nécessité d'un système flexible qui s'adaptera au changement des règles de gestion et aux nouveaux besoins.

Le but de notre étude est de réaliser un système qui assure la gestion et l'aide à la décision, susceptible de s'adapter aux changements de l'environnement et qui assure :

- La communication administrateur ⇔ système ;
- L'introduction des nouvelles règles et procédures de gestion ;
- La génération des structures de données et les masques de saisie des transactions ;
- Agrégation des informations saisies ;
- Introduction et maintenance des systèmes d'information.

Pour présenter nos travaux, nous avons organisé le présent mémoire en quatre chapitres :

Le premier chapitre est consacré aux SGBDRs. En insistant particulièrement sur les contraintes prises en charge par un SGBDR et les contraintes non prises en charge, afin de mieux situer les limitations des SGBDRs.

Dans le deuxième chapitre, nous avons introduit le nouveau système qui représente une combinaison d'un SGBDR et un système expert. Après avoir décrit les concepts de base des SEs, nous avons met l'accent sur la capacité que possède un SE de gérer les contraintes qu'un SGBDR ne peut gérer.

Le troisième chapitre, permet d'expliciter un certain nombre de concept de la modélisation. Puis les fonctionnalités que le système doit offrir aux usagés. Enfin une conception détaillée du système, suivant le cycle de vie en cascade, en utilisant le langage de modélisation UML (unified modeling language).

Dans le quatrième chapitre nous présentons à l'implémentation de la solution retenue au niveau de la conception, ainsi que la validation.

Nous achevons notre mémoire par une conclusion générale, dans laquelle nous montrons l'apport du système et de l'approche considérée.

I- Systèmes de gestion de base de données relationnels

I- 1-Introduction

Dans le cadre d'un projet d'informatisation, la conception d'une base de données relationnelle passe d'abord par l'identification des objets de gestion et des règles de gestion du domaine modélisé (interviews des utilisateurs, étude des documents manipulés, des fichiers existants, ...). Une fois énoncées et validées, ces règles nous conduisent automatiquement à la structure du modèle relationnel correspondant.

Après construction de la base de données, il nous faut un système capable de définir, de manipuler et de garantir la sécurité des données. Ce système et le système de gestion de base de données relationnel (SGBDR).

I-2- Définitions du système de gestion de base de données relationnel

Le système de gestion de base de données (SGBD) est un logiciel permettant de gérer de manière efficace, un volume important de données structurées, accessible par des utilisateurs simultanés locaux ou non locaux [Chri 90] . Un SGBDR relationnel est un SGBD basé sur le modèle relationnel.

I-3- Architecture du SGBDR

Les SGBDRs présentent la base de donnée sous trois niveaux d'abstraction :

Le niveau externe: Offre à l'utilisateur final un schéma externe particulier, à travers lequel il voit la base.

Le niveau conceptuel (logique) : Il s'agit d'une vision tabulaire où la sémantique de l'information est exprimée en utilisant les concepts de relation, attributs et de contraintes d'intégrité. Le niveau conceptuel est défini au travers du schéma conceptuel [Mor 92].

Le niveau interne : Il regroupe les services de gestion de la mémoire secondaire. Le niveau physique est responsable du choix de l'organisation physique des fichiers ainsi que de l'utilisation de telle ou telle méthode d'accès en fonction de la requête. Ce niveau doit également assurer le partage des ressources, la gestion de la concurrence et des pannes. La personne responsable de ce niveau est un administrateur de bases de données [Mor 92].

I-4- Les langages utilisés par le SGBDR

Le SGBDR comporte un langage de définition de données (DDL) et un langage de manipulation de données (DML) pour peupler la base de données (création, suppression, modification et interrogation).

Finalement, il faut pouvoir gérer les utilisateurs et leur droit d'accès aux données. A cette fin, l'administrateur de la base de données utilise un langage de contrôle des données (DCL).

Dans la plupart des SGBDR, un seul langage regroupe tous les sous-langages mentionnés ci-dessus. Le plus répandu est le SQL (Stuctured Query Language).

Lorsque les programmes d'application accèdent à une base de données, le langage de programmation, doit être couplé à des instructions DML. Ces instructions font partie du sous langage de données.

I-5- Les composants de base

- **Cardinalités :** On appelle cardinalité d'une entité dans une association le nombre minimum de fois et le nombre maximum de fois où une occurrence de l'entité est susceptible de participer à l'association [Gard 89].

  La cardinalité d'un lien peut prendre les valeurs suivantes :

  0.1 = Participation unique et optionnelle à l'association

  0.n = Participation multiple et optionnelle à l'association

  1.1 = Participation unique et obligatoire à l'association

  1.n = Participation multiple et obligatoire à l'association.

Les cardinalités de type 0-1 ou 1-1 peuvent être exprimées grâce aux contraintes d'intégrité.

- **Domaine :** Un domaine est un ensemble de valeurs caractérisé par un nom. Il a pour équivalence une déclaration de type dans un langage de programmation [Gard 89].

Un domaine peut déclarer des checks (des contrôles), des valeurs par défaut ou des tests, qui seront appliqués en temps réel sur les données.

- **Relation :** Une relation est un sous ensemble de données caractérisé par des attributs et visualisable sous forme de table [Mor 92].
- **Attribut :** Colonne d'une relation caractérisée par un nom et correspond à un domaine [Mor 92].
- **n-uplets :** ligne d'une relation (où n est le degré de la relation, c'est à dire le nombre d'attributs de la relation) [Mor 92].
- **Schéma de relation :** Nom de la relation, suivi de la liste des attributs avec leurs domaines [Mor 92].
- **Clé primaire :** Ensemble minimum d'attributs qui permet de distinguer chaque n-uplet de la Table par rapport à tous les autres. Chaque Table doit avoir une clé primaire [Gard 89].
- **Clé candidate :** Ensemble minimum d'attributs susceptibles de jouer le rôle de clé primaire [Gard 89].
- **Clé étrangère :** Elle fait référence à la clé primaire d'une autre Table.
- **Valeur nulle :** Si, pour un tuple donné, il n'existe pas de valeur pour un attribut, alors cet attribut reçoit une valeur nulle (NULL). La nullité d'une valeur est ambiguë, puisqu'elle peut signifier que :

  - la donnée n'est pas pertinente pour le tuple.

  - la donnée n'est pas connue du système, bien qu'elle soit pertinente.

- **Intégrité référentielle :** L'intégrité référentielle spécifie qu'un tuple d'une relation qui fait référence à une autre relation doit faire référence à un tuple existant dans cette relation. Par conséquent, le SGBDR doit vérifier, lors d'insertions ou de modifications, l'existence du tuple référencé [Bowm 02].

- **règles de gestion :** Une règle de gestion est une phrase qui permet d'expliquer la cardinalité qui existe entre une entité et une association, elle spécifie ce que le système d'information doit faire ou comment il doit être structuré [1]. Plus précisément pour l'aspect informationnel les règles de gestion décrivent les interactions entre les données, selon un classement en quatre types :

- ***Définition*** : Caractéristiques ou propriétés d'un objet dans le système d'information. Exemple : Un prêt est identifié par le numéro de livre, le matricule du lecteur et la date d'emprunt.

- ***Fait*** : Expression d'une certitude ou d'une existence dans le système d'information. Exemple : Une ligne d'abonné dépend d'un seul commutateur.

- ***Formule*** : Calcul utilisé dans le système d'information.

Exemple : Un salarié ne peut participer à plus de quatre formations par an.

-***Validation*** : Contrainte sur une valeur dans le système d'information.

Exemple : la date de divorce est supérieure à la date de mariage.

- **Contraintes d'intégrités:** Les contraintes d'intégrité sont des règles de gestion particulières exprimée de manière déclarative. Elles permettent d'assurer, à l'exécution, la cohérence des éléments d'information dupliqués dans plusieurs tables [Bowm 02].

I-6- Contraintes prises en charge par un SGBDR : Les contraintes d'intégrités

Le modèle d'une base de données relationnelle implique, par sa conception, un certain nombre de contraintes d'intégrité qui traduisent les propriétés sémantiques des données :

- ***Contrainte de clé*** **:** La contrainte de clé spécifie les clés candidates de chaque schéma de relation. Les valeurs des clés candidates doivent être uniques pour chaque tuple dans n'importe quel exemplaire de relation de la base de données.

- ***Contrainte d'intégrité d'entité :*** L'intégrité d'entité exige que chaque relation possède une seule clé primaire déclarée et qu'aucun composant de la clé primaire d'une relation ne puisse prendre la valeur nulle.

- ***Contrainte d'intégrité de référence :*** L'intégrité de référence exige que chaque valeur non nulle d'une clé externe doit exister ailleurs dans la base de données comme valeur clé primaire cible. Elle est spécifiée pour maintenir la cohérence entre les tuples de deux relations.

- ***Contrainte de non-*****nullité :** On peut ajouter à une colonne la contrainte de non-nullité qui implique que cette colonne ne peut pas avoir de valeur nulle.

- ***Contrainte de la valeur nulle des clés étrangères :*** Dans une relation d'entité, la valeur nulle est autorisée seulement pour les clés étrangères des types d'entité à participation facultative dans un type d'association.

  - Dans une relation d'entité, la valeur nulle n'est pas autorisée pour les clés étrangères des types d'entité à participation obligatoire dans un type d'association.

  - Dans une relation d'association la valeur nulle n'est pas autorisée pour les clés étrangères des types d'entité participant dans le type d'association.

- ***Contrainte d'unicité :*** UNIQUE garantit également le caractère distinct de chaque valeur de la colonne, mais permet que la colonne soit définie par NULL. UNIQUE est aussi souvent implémentée en tant qu'index.

- ***Contrainte de valeur automatique :*** DEFAULT définit une valeur fournie automatiquement par le système lorsque l'utilisateur n'en a pas saisie explicitement.

- ***Contrainte de domaine :*** CHECK indique que les données peuvent être saisies dans une colonne particulière : il s'agit d'une façon de définir le domaine de la colonne. Pour cela on utilise les opérateurs et les mots clés suivants :

  - **Opérateurs de comparaison :** Pour déterminer laquelle est la plus *grande*, la plus *petite*, *inférieure*, ou *égale* à une valeur quelconque de la base de donnée. SQL propose à cette fin un jeu d'opérateurs de comparaison, il s'agit de : = , < , > , <= , >= , <>.

  - **Combinaisons ou négations logiques :** On utilise les opérateurs logiques AND, OR, et NOT lorsque on traite plusieurs conditions. AND relie plusieurs conditions et ne renvoie un résultat que lorsque toutes les conditions sont vraies. OR relie également plusieurs conditions, mais renvoie un résultat lorsque n'importe laquelle des conditions est vrai. L'opérateur NOT rend négative une expression.

  - **Fourchette :** La fourchette est une autre condition de recherche courante. Il existe deux façon d'indiquer les fourchettes : avec les opérateurs de comparaison > et < ; avec le mot clé BETWEEN. On utilise BETWEEN pour rechercher la valeur inférieure et la valeur supérieure, ainsi que les intermédiaires. Pour trouver toutes les lignes extérieures à la fourchette, on utilise NOT BETWEEN.

  - **Listes :** Le mot clé IN nous permet de sélectionner des valeurs correspondant à une liste de valeurs. On utilise le NOT IN pour trouver les valeurs qui n'appartiennent pas à la liste.

- **Valeurs inconnues :** IS NULL est un emplacement d'information inconnue. Il ne signifie pas zéro ou vide. IS NOT NULL signifie qu'une valeur doit être saisie.
- **Correspondance de chaînes de caractères :** on utilise LIKE pour trouver des valeurs en comparant une partie de chaîne de caractères avec tous les valeurs. SQL propose deux caractères génériques utilisables avec LIKE :
  - le pourcentage « % » : Remplace une chaîne de caractères de longueur quelconque ou nulle.
  - Le caractère de soulignement « _ » : Remplace exactement un caractère.

I-7- Contraintes non prises en charge par le SGBDR

I-7.1. Contrainte conditionnelle

Une contrainte conditionnelle est une contrainte qui signifie que chaque valeur d'un champ (colonne) est remplie selon une condition qui dépend de la valeur d'un autre champ.

Exemple : Soit la table CLIENT (Code, Nom, Prénom, Forme Juridique, Raison Social).

La Raison Sociale est non nulle quand la Forme Juridique est égale à personne physique, et nulle si la Forme Juridique est égale à personne morale.

I-7.2. Contrainte d'ordre d'affectation de valeur

Certains champs ne peuvent être remplies qu'après le remplissage des valeurs du champ prioritaire.

Exemple : On ne peut affecter la valeur du champ commune qu'après le choix de la valeur du champ wilaya.

I-7.3. Contrainte du champ obligatoire

La contrainte du champ obligatoire spécifie les champs qui sont obligatoires lors de la saisie. Par exemple : le Code du client est obligatoire.

I-7.4. Contrainte de visibilité

La contrainte de visibilité spécifie les champs visibles lors de la saisie.

I-7.5. Contrainte de la lecture seule

La contrainte de la lecture seule spécifie les champs qui ne sont pas modifiable, et qui sont conçu pour la lecture seulement.

I-8- L'interface utilisateur

L'interface utilisateur est sans aucun doute la composante la plus importante du n'importe quel système aux yeux de l'utilisateur final, et un SGBDR doit s'ouvrir à l'extérieur avec une interface adaptée aux différentes catégories d'utilisateurs et d'applications. Or le SGBDR n'est pas charger de définir cette dernière, il est nécessaire de le coupler avec un programme qui peut définir une interface qui doit se révéler la plus conviviale possible et aisée à l'emploi.

I-9- Conclusions

Les fonctionnalités du SGBDRs sont nombreuses et offrent des possibilités puissantes et variées. Ils sont cependant très limités pour trois raisons essentielles :

- Ils sont conçus pour gérer des données et des règles de gestion, c'est -à-dire structurées de manière régulière sous forme de tables. L'introduction d'une nouvelle règle ou d'un nouvel objet implique la reconstruction du MCD et par conséquent tout le système d'information.
- Les SGBDRs ne permettent que de retrouver des données stockées dans la base. Ils ne sont guère capables de raisonner afin de prendre des décisions, ou de comparer des situations nouvelles par rapport à la mémoire des faits enregistrés.
- Les SGBDRs sont loin de présenter des interfaces simples, accessibles par des non informaticiens. Sans compter qu'il faut arriver à offrir des possibilités d'interface plus adaptées aux applications.
- Ils ne sont pas capable de prendre en charge toutes les contraintes de la BD.

Chapitre II

II-1- Introduction

Les SGBDR représentent une avancée considérable dans le domaine de base de données (BD), leurs performances générales n'ont jamais cessé de croître et ces systèmes règnent aujourd'hui en maîtres sur le marché des applications de BD [Del, 91]. Il n'en reste pas moins que ces système se révèlent inadaptés aux nouvelles applications. En particulier, le SGBDR est incapable de prendre en compte les contraintes dynamiques.

En utilisant les systèmes expert, la qualité du services se verra par conséquent améliorée par des décision rapides toujours cohérentes ainsi que par la réduction du temps, d'effort et surtout du coût.

Les systèmes experts (SEs) et les SGBDRs se sont, jusqu'à très récemment, développées de façon quasi indépendante. Il est maintenant classique de chercher parmi les outils d'aide à la décision des idées pour améliorer les performances interactifs des SGBDRs.

II-2- Système intégré

Le système intégré est un système qui combine les caractéristiques du couple SGBDR et SE . Autrement dit, c'est un SE qui repose sur un SGBDR, il a pour but de pallier la relative faiblesse de ces deux derniers.

Ce système est tout d'abord un SE. En ce sens, il doit fournir à l'utilisateur des interfaces conviviales, traiter les contraintes qu'un SGBDR ne peut pas traiter et le décharger d'autres contraintes. Le SGBDR doit prendre en charge les contraintes non effectuées par le SE, et offrir un langage de requête permettant de poser des questions et d'effectuer des mises à jour.

II-3- Les systèmes expert

Les systèmes experts apportent un renouveau fondamental dans les techniques d'aide à la décision qui sont utilisées dans les entreprises. Par leur effet de démultiplication de la capacité de raisonnement des décideurs, ils sont appelés à être de véritables assistants de gestion dont les entreprises devront inévitablement se doter à l'avenir pour accroître leurs performances et rester compétitives [Ern, 88].

II-3.1.Définition

Un système expert est un programme ou ensemble de programmes informatiques mettant en œuvre, d'une part, un ensemble de connaissances relatives à un domaine donné, et, d'autre part, des règles d'interprétation de ces connaissances, afin de fournir à l'humain, des solutions ou une aide à la prise de décision dans la résolution de problèmes dans ce domaine [Kar, 93].

II-3.2. Architecture générale d'un système expert

Un système expert est constitué de trois composantes :

- ✓ Une base de connaissances ;
- ✓ Un moteur d'inférences ;
- ✓ Une interface.

A- La base de connaissances

La base de connaissances contient toute l'information dont un expert humain aurait besoin pour s'acquitter de son travail, dans un domaine donné. C'est la seule composante qui contient les connaissances propres au domaine que le système est censé recouvrir. Cette base de connaissances est elle-même divisée en deux composantes la *base des faits* et la *base des règles*.

❖ **La base des faits**

La *base des faits* est la mémoire de travail du système expert. Elle est variable au cours de l'exécution et elle est vidée lorsque l'exécution se termine. Au début de la session elle est vide, puis elle est complétée par des faits temporaires relatifs au problème en cours de traitement , fournis soit par l'utilisateur ou déduits par le système.

➢ **Les faits**

Les faits sont la façon la plus élémentaire de représenter la connaissance. Il y a deux façons pour un système de connaître ces éléments de connaissance : ils peuvent être fournis par l'utilisateur, une base de données, des capteurs..., ou déduits par le système grâce à un raisonnement automatique.

➢ **Représentation des faits**

Les faits peuvent s'exprimes de différentes façons :

- *Logique d'ordre 0 (logique des propositions pures)*
- *Logique d'ordre 0+ (logique des propositions avec variable globales)*
- *Logique d'ordre 1 (logique des prédicats)*
- *Logique d'ordre supérieur à 1*
- *Objets structurés (schémas)*

. Logique d'ordre 0 (logique des propositions pures)

Le fait est une proposition (symbole) logique dont la valeur de vérité peut dépendre du lieu, de l'époque, etc.

Exemple : P1 : « Il_pense »

. Logique d'ordre 0+ *(logique des propositions avec variable globales)*

Le fait est représenté par un triplet de la forme :

<Attribut> <Comparateur> <Valeur>

Exemple : P1 : « valeur < 2 »

. Logique d'ordre 1 *(logique des prédicats)*

La logique d'ordre 1 permet d'utiliser des variables et des quantificateurs dans définition des faits.

Exemple : P1 : Cercle(X, Y)

. Logique d'ordre supérieur à 1

Les variables peuvent représenter des prédicats, les prédicats donc peuvent porter sur des prédicats.

Exemple : Soit le prédicat cercle(X , Y) on peut définir le prédicat 'fonction(R)'.

. Objets structurés (Schémas, « frames »)

La notion d'objet structuré traduit une démarche pour diviser la connaissance en sous structures définissant des *micro-mondes*. A chaque objet sont attachés différents types de connaissances et de propriétés.

Exemple : voiture ((Couleur rouge)

(Propriétaire Michel))

➢ **Métafaits et métavaleurs**

Il n'est pas suffisant que le système ait des connaissances, il faut aussi qu'il ait des *métaconnaissances*. Il faut par exemple qu'un système expert puisse savoir si une valeur a été attribuée à un fait. Dans la négative, cette valeur pourra être demandée à l'utilisateur. Mais si l'utilisateur ne peut pas répondre, il faudra que le système puisse le savoir afin de ne pas poser éternellement la même question. La seule manière d'attribuer une valeur à un tel fait sera alors de la déduire d'autres faits.

❖ **La base des règles**

La base des règles contient des principes plus généraux, des heuristiques de résolution de problèmes qui représentent les modes de raisonnement propres au domaine considéré. Ce sont les connaissances déductives souvent représentées par ce qu'on appelle des *règles de production*. La connaissance de ces heuristiques peut provenir, de la part de l'expert humain, soit d'une accumulation d'observations empiriques, soit de connaissances techniques propres au domaine.

➢ **Les règles**

Les règles indiquent quelles conséquences tirer lorsque telle situation est établie.

Une règle est de la forme :

Si < conditions> alors <action>

B- Le moteur d'inférence

Le moteur d'inférence permet de manipuler les données stockées dans la base de connaissances de façon à pouvoir résoudre les problèmes posés au système. Le moteur d'inférence reste séparé des connaissances du domaine et généralement il n'est pas dépendant de son application à un problème particulier.

❖ **Cycle de base d'un moteur d'inférence**

Le cycle de base d'un moteur d'inférence est généralement divisé en trois étapes :

✓ Le filtrage.

✓ La résolution de conflits.

✓ L'exécution.

➢ **Le filtrage**

Le moteur d'inférence recherche dans la base des règles celles qui sont déclenchables en comparant les prémisses de chaque règle à la base des faits. Les règles retenues lors de cette étape forme *l'ensemble des conflits.*

➢ **La résolution de conflits**

Dans cette phase le moteur choisit dans l'ensemble des conflits les règles qui vont effectivement être déclenchées.

➢ **L'exécution**

Dans cette phase le moteur d'inférence effectue la mise en œuvre des parties <action> des règles rétenues lors de la résolution des conflits.

Généralement, cette phase consiste à ajouter de nouveaux faits à la base des faits.

❖ **Mode d'invocation des règles**

On distingue essentiellement trois modes principaux de fonctionnement des moteurs d'inférences :

- ✓ Le *chaînage avant (mode déductif).*
- ✓ Le *chaînage arrière (mode inductif).*
- ✓ Le *chaînage mixte.*

➢ **Le chaînage avant**

L'utilisateur du système expert introduit des faits. A partir de ces faits, le système va essayer de déduire toutes les conclusions possibles. Parmi toutes conclusions déduites, certaines intéressent l'utilisateur, d'autres qui ne lui signifient rien.

➢ **Le chaînage arrière**

La méthode de chaînage arrière part des conclusions. Elle cherche à trouver, à partir des conclusions, les prémisses nécessaires à leur déclenchement.

➢ **Chaînage mixte**

C'est une combinaison d'un chaînage arrière à du chaînage avant.

C- Interface utilisateur

La troisième composante importante d'un système expert contient des interfaces utilisateur. Les deux plus importantes sont:

1) une interface grâce à laquelle les utilisateurs pourront obtenir une consultation du système expert. Ce peut être un formulaire dans le cas d'un système fonctionnant en chaînage avant ou un module de questions posées à l'utilisateur pour un système fonctionnant en chaînage arrière ou mixte.

2) Et une interface permettant l'acquisition des connaissances. Cette dernière est utilisée par l'expert humain pour mettre à jour et vérifier ses connaissances.

II-3.3. Caractéristiques des systèmes experts

Les systèmes experts ont des caractéristiques particulières qui les distinguent des autres systèmes informatiques. Parmi ces caractéristiques, on trouve :

- L'architecture particulière de ses modules ;
- La codification des connaissances dans une base de connaissances ;
- La séparation des connaissances de la partie contrôle ;
- La capacité à raisonner sur des données incomplètes, inexactes et floues (sur certains systèmes experts) ;
- La possibilité d'explication des résultats.

II-3.4. Générateurs de systèmes experts

Un système expert concerne un domaine précis d'application. Un *système expert général* (ou *générateur de systèmes experts*) est un outil de développement de systèmes experts particuliers. Un système expert général contient un moteur d'inférences et un système de représentation des connaissances.

II-3.5. Contraintes prises en charge par un système expert

Un systèmes expert utilise, dans ça programmation, un langage procédural ce qui lui permet de traiter plusieurs cas et de résoudre beaucoup de problème. Tel que, pour prendre en charge une contraintes il suffit d'ajouter une procédure au système qui traite cette dernière.

Toutes les contraintes peuvent être considérées par un SE, mais sûrement après une très longue programmation et un temps de réponse non négligeable pour certaines contraintes (par rapport aux SGBDRs). Puisque le SGBDR prend en charge la plus part des contraintes, ça ne va pas être intéressant de refaire avec le SE ce que un SGBDR peut faire. Car la concurrence acharnée découlée de l'ouverture du marché mondial exige le gain du temps. De ce fait, il sera intéressant d'utiliser les performances du SE pour compléter celles du SGBDR et traiter les contraintes non prises en charge par ce dernier.

II-4- Rôle de système intégré

Si un système expert assure une fonction très différente d'un SGBDR il n'en reste pas moins qu'il a besoin de celui-ci car il est nécessaire de connaître la situation pour prendre la décision [Ernst 88].

On peut résumer le rôle respectif de ces deux types de systèmes de la manière suivante :

SE : Aide à la décision.

Problèmes imprécis. Il n'y a pas qu'une solution.

Proposer de façon interactive des solutions voisines [Ernst 88].

SGBDR : Connaître la situation.

Répondre à des questions précises [Ernst 88].

Le rôle de notre système intégré est la combinaison des deux rôles.

II-5- Contraintes prises en charge par le système intégré

Le système hybride peut prendre en charge tous les contraintes de la BD grâce au SE. Certains de ces contraintes exigent l'accès à la base de donnée à chaque fois ; il est plus intéressant d'éviter de reprogrammer ces derniers, et les confier au SGBDR pour les prendre en charge. D'autres contraintes n' ont pas besoin d'accéder à la BD et un SE est mieux placé pour les prendre en charge.

En général, un SE prend en charge les contraintes dynamiques et le SGBDR ne prend en charge que les contraintes statiques.

On peut classer les contraintes en deux classes, qui sont :

II-5.1. Contraintes prises en charge par le SGBDR

Un SGBDR par sa nature prend en charge plusieurs contraintes, parmi ces contraintes nous choisissons celles qui exigent le retour à la BD à chaque utilisation, ces contraintes sont :

- *Contrainte de clé et d'index ;*
- *Contrainte d'intégrité d'entité ;*
- *Contrainte d'intégrité de référence ;*
- *Contrainte d'unicité ;*
- *Contrainte de domaine dans le cas de correspondance de chaînes de caractères ;*

II-5.2. Contraintes prises en charge par le système expert

Un SE peut prendre en charge tous les contraintes dynamiques et statiques. Dans notre cas, nous ne confions au SE que les contraintes dynamiques et les contraintes qui n'exigent pas un accès à la BD . Afin que le système soit plus efficace. Ces contraintes sont :

- *Contrainte de non*-nullité ;
- *Contrainte de la valeur nulle des clés étrangères ;*
- *Contrainte de valeur automatique* ;
- *Contrainte de domaine* : parmi les opérateurs et les mots clés prise en charge nous citons :
  - ✓ *Opérateurs de comparaison ;*
  - ✓ *Combinaisons ou négations logiques ;*
  - ✓ *Fourchette ;*
  - ✓ *Listes ;*
  - ✓ *Valeurs inconnues .*

- *Contrainte conditionnelle ;*
- *Contrainte d'ordre d'affectation de valeur ;*
- *Contrainte du champ obligatoire ;*
- *Contrainte de visibilité ;*
- *Contrainte de la lecture seule .*

II-5- Fonctionnalités du système intégré

Après avoir interfacer les deux système SGBDR et SE notre nouveau système et devenu très riche puisque il combine les fonctionnalités des deux systèmes, parmi ces fonctionnalités on cite les suivantes :

- La possibilité d'effectuer les opérations classiques du calcul relationnel (union, restriction, projection, jointure, différence) ;
- Les fonctions d'agrégats traditionnelles des langages relationnel ;
- Les fonctions arithmétiques et plus généralement celles définies par les utilisateurs ;
- Les mises à jour des faits au travers des règles ;
- La négation, qui permet de référencer des faits non existants dans la base ;
- La récursivité ;
- La modularité avec la gestion de base de données et de méta-règles.

En bref, toutes les fonctionnalités offertes par les SEs et les SGBDRs devont pouvoir figurer dans le système intégré.

II-6- Conclusion

Les SGBDRs ne permettent pas de programmer entièrement une application, ils doivent être combinés avec des langages généraux comme les SEs. Cette combinaison d'un langage avec un autre nous donne notre système.

« La réalité est toujours plus complexe que la théorie qui tente de la décrire.

En effet, pour appréhender cette réalité, on est bien obligé de la simplifier et de la schématiser ;

Car, elle s'en venge quelque fois avec malice…..

Mais souvent, elle se prête à ce jeu, accepte de donner à l'analyste

La joie d'avoir participer à une création… »

Fontelliet.

« On ne peut pas avoir le simple sans passer par le complexe. »

I- Introduction

La conception de logiciels sophistiqués alourdit considérablement les contraintes imposées aux développeurs. Les informaticiens sont confrontés à une complexité croissante, à la fois du fait de la nature des applications, des environnement distribués et hétérogènes, de la taille des logiciels, et des attentes des utilisateurs en matière d'ergonomie. Pour surmonter ces difficultés, les informaticiens doivent apprendre à faire, à expliquer, et à comprendre [Mul, 01]. C'est pour quoi qu'il est nécessaire de considérer un modèle pour la démarche de développement, afin de réaliser tout système.

Le modèle en cascade est le modèle le plus classique, il décrit le cycle de vie d'un logiciel par la succession des étapes suivantes : la spécification des besoins, la conception, l'implémentation, le test et la validation [Bru, 01].

Le contenu du modèle dépend du problème. Un langage de modélisation comme UML (Unified Modeling Language) est suffisamment général pour être employé dans tous les domaines informatiques. UML est un langage de modélisation objet. En tant que tel, il facilite l'expression et la communication de modèles en fournissant un ensemble de symboles (la notation) et des règles qui régissent l'assemblage de ces symboles (la syntaxe et la sémantique) [Mul, 01].

Dans ce contexte, Nous proposons dans ce chapitre d'adapter les techniques et les outils de l'UML afin que les développements s'appuyant sur cette notation puissent en bénéficier et pour modéliser toutes les facettes du système, depuis la phase d'analyse des besoins jusqu'au test. De plus, l'UML est meilleure pour représenter les notions d'orienté objet utilisées dans notre conception.

Dans ce chapitre nous présentons le langage de modélisation UML. Puis, nous spécifions les besoins des utilisateurs du système et en fin, nous expliquons la conception de l'application implémentée.

II- Problématique

Les utilisateurs des logiciels de gestion, dans le milieux professionnel ont constaté certaines anomalies, des logiciels conçus pour être des outils de travail au service de l'utilisateur ne repondent plus à tous leur besoins qui ne cessent de croître.

Exemple 1 :

Le service d'assurance assure des différents objets, tels que : personnes, récoltes, apicoles et avicoles...

L'assurance d'un nouvel objet comme un réacteur nucléaire ou un métro, non pris en compte par le système actuel, entraîne la réfection du système d'information (SI), car la description de cet objet est complètement différente de la description des autres. De ce fait, le changement doit être répercuté sur le logiciel informatique de la production d'assurances, ce qui implique que des modification dans les sources du logiciel ce qui n'est pas toujours possible car les sources ne sont pas disponible dans la plus part des cas, sans compter un risque d'apparitions d'erreurs dans le fonctionnement courant du logiciel. D'où la nécessité d'avoir un système qui aide le programmeur à développer le nouveau logiciel, en perdant le moins de temps possible, et en prenant le moins de risque d'erreurs.

Exemple 2 :

Chaque commune appartient à une wilaya, avant de saisir la commune il faut sélectionner sa wilaya. Dans ce cas le champ wilaya est un champ obligatoire, son ordre d'affectation de valeur est prioritaire que celui du champ commune.

L'ajout de nouvelles règles peut aider l'utilisateur à décider lors de la saisie. Cependant, il oblige le programmeur à écrire un programme ce qui entraîne le perte de temps. D'où la nécessité d'avoir un système qui décharge le programmeur d'écrire ce genre de programmes, à chaque fois qu'une contrainte verrait à être considérée.

III- Langage de modélisation UML

III-1- Avantages de l'UML

Le choix du langage de modélisation porte sur le langage UML de part le fait qu'il constitue une unification des méthodes objets, tirant donc profit des avantages de chacune de ces méthodes de l'origine de l'unification et qu'il constitue un standard pour la modélisation orientée objet. Ce choix est également justifié par plusieurs autres avantages qu'offre UML par rapport à d'autres méthodes objet et par rapport à nos besoins. Nous les décrivons dans ce qui suit :

- UML est fondé sur un métamodèle. Ce métamodèle donne une vision plus formalisée du langage et définit toutes les possibilités d'extensibilité de celui-ci.
- UML se veut un langage non fermé : les éléments de modélisation qu'il propose sont génériques, extensibles et configurable par l'utilisateur [Mul, 01].
- La simplicité et la capacité d'expression visuelle qu'offre l' UML, et qui permettent de faciliter la communication autour des besoins entre les différents acteurs du système. UML offre une bonne communication avec les utilisateurs. Les concepts manipulés en UML correspondent à des réalités concrètes pour l'utilisateur.
- La diversité des diagrammes qu'offre UML (au nombre de neuf) et qui permet de présenter plusieurs perspectives ou vues de l'architecture du [illegible] développer.

III-2- Concepts et notations

Nous décrivons dans cette partie les concept de base du langage UML, nécessaire pour notre étude. UML est la forme contractée de « Unified Modeling [illegible], il représente l'état de l'art des langages de modélisation objets.

UML s'appuie sur des concepts, des relations et des diagrammes :

- Des concepts (structurels, comportementaux, annotationnels, groupements).
- Des relations (association, généralisation, agrégations, compositions... etc.).
- Des diagrammes (statiques et dynamiques).

III-1- Les concepts

UML supportent quatre (04) types de concepts : [ket & al, 01]

- *Les concepts structurels* : représentés par les classes, les interfaces, les collaborations...
- *Les concepts comportementaux*: représentés par les interactions et les états des objets.
- *Les concepts annotationnels*: représentés par les notes. Une note est un commentaire attaché à un ou plusieurs éléments de modélisation [Mul, 97].

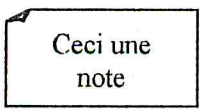


Figure III-1: Représentation d'une note

- *Les concepts de groupement*: représentés par les sous-systèmes et les paquetages.

Les paquetage : Le concept de paquetage unifié les concepts de catégorie et de sous système [Fan &al, 00]. Un paquetage regroupe des éléments de la modélisation [Gab, 98]: cas d'utilisation, classes, objets, modules ou composant. L'importation permet aux éléments d'un paquetage d'accéder aux éléments d'un autre paquetage. Cette relation est à sens unique et est représentée par une relation de dépendance associée à un stéréotype « import ». l'accès d'un paquetage à un autre paquetage est présenté par le stéréotype « accède ».

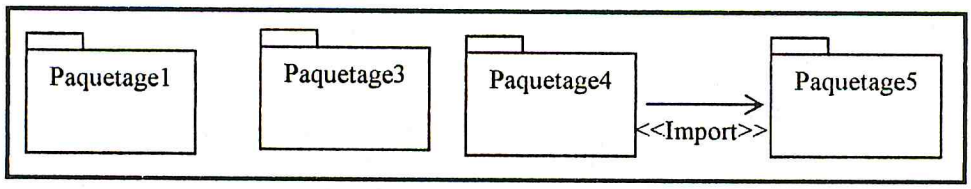


Figure III-2: Réprésentation graphique d'un paquetage

III-2- Les relations

Elles permettent de relier les concepts entre eux. On distingue quatre types de relations, les associations, les généralisations, les agrégations, les compositions ... etc [Ber, 02].

III-2.1. L'association

Relation structurelle précisant que les objets d'un élément sont reliés aux objets d'un autre élément.

III-2.2. La généralisation

Relation entre un élément général et un élément dérivé de celui-ci, mais plus spécifique (désigné par sous-élément ou élément fils).

Le plus souvent, la relation de généralisation est utilisée pour représenter une relation d'héritage.

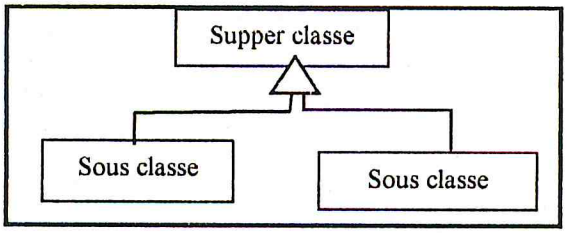


Figure III-2: Représentation graphique d'une généralisation

III-2. 3. L'agrégation

Représente une association non symétrique dans laquelle une extrémité joue un rôle prédominant par rapport à l'autre extrémité [Mul, 97].

III-2. 4. La composition

Relation d'agrégation mettant en avant une notion de propriété forte et de coïncidence des cycles de vie. Les parties sont créées et détruites en même temps que [illegible 02]. Elle est distinguée par un losange plein.

III-3- Les diagrammes d'UML

UML définit neufs types de diagrammes qui sont présentés dans l'extrait du méta modèle suivant :

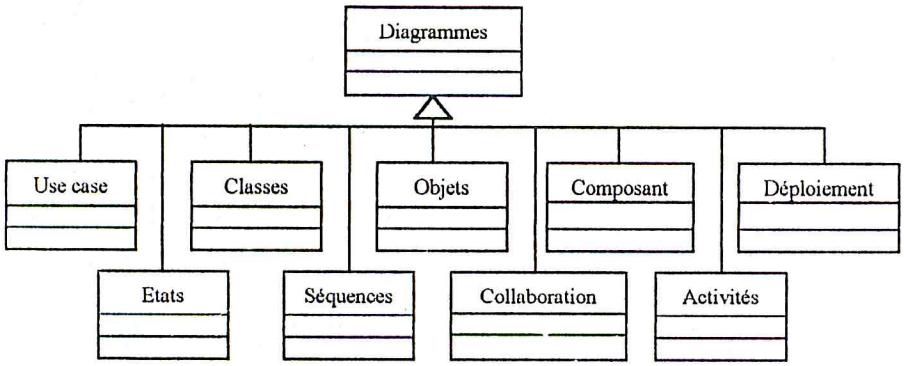


Figure III-3: Les diagrammes d'UML[Ket& al, 01].

Dans ce document nous utilisons sept diagrammes parmi les neuf diagrammes de l'UML. Pour spécifier les besoins de l'utilisateur nous utilisons les diagrammes de cas d'utilisation et les diagrammes de séquence.

Les diagrammes de classes, de collaboration, d'états-transitions, d'activités et de composants sont utilisés pour la conception du système.

Pour l'implémentation du système nous utilisons les diagrammes de séquence.

III-3.1.Diagramme de cas d'utilisation :

Un diagramme de cas d'utilisation permet de décrire les interactions entre les acteurs de l'organisation et le système dans chacun des cas d'utilisation envisagés. Il décrit le comportement d'un système au point de vue de l'utilisateur et fixe les limites du système et les relations entre le système et l'environnement [Mul, 97].

Les divers cas d'utilisation du système vont être présentés dans les diagrammes de cas d'utilisation. Les cas d'utilisation peuvent être liés les uns aux autres par trois types de relations.

- *La relation d'utilisation :* lorsque une ou plusieurs tâches sont utilisées régulièrement, on peut les factoriser dans un même use case et faire de telle sorte que d'autres use cases l'utilisent en le pointant par une flèche [Gab, 98].

Cet use case est en fait une sous partie de chaque use case qui l'utilise. Ce qui permet de décomposer un use case complexe en plusieurs uses cases.

- *La relation d'inclusion :* le cas d'utilisation source comprend également le comportement de son cas d'utilisation destination. Cette relation a un caractère obligatoire (à la différence de la généralisation) et permet ainsi de décomposer des comportements partageables entre plusieurs cas d'utilisation différents [Gab 98].

- *Le relation d'extension :* le cas d'utilisation source ajoute son comportement au cas d'utilisation destination. L'extension peut-être soumise à condition. Cette relation permet de modéliser des variantes de comportement d'un cas d'utilisation [Gab 98].

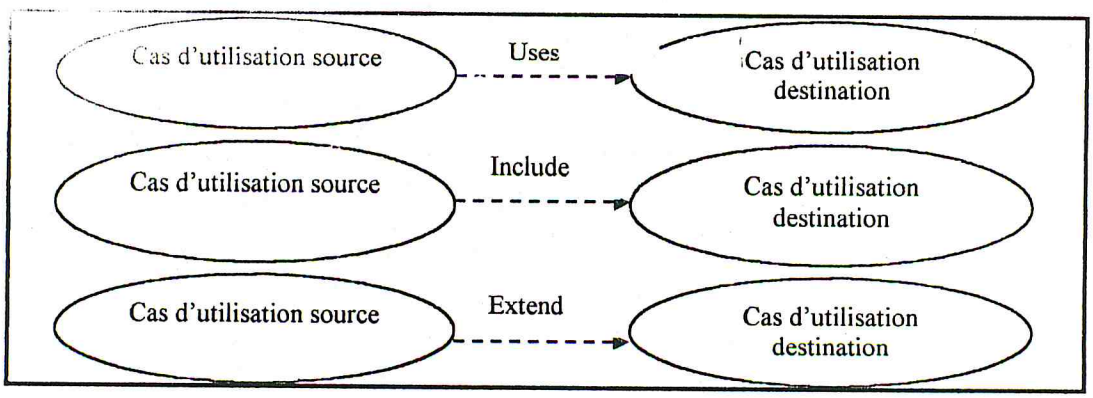


Figure III -4: Les trois relations entre cas d'utilisation

III-3.2. Diagramme de classes :

Les diagrammes de classes montre la structure du système et les éléments des classes tels que: les classes, les relations d'héritages entre classes, les associations, dont les agrégations, les attributs, les opérations et la spécification d'opérations et contraintes au niveau des entités [Ber, 02].

➢ **Stéréotype :** permet de définir une utilisation particulière d'éléments de modélisation existants ou de modifier la signification d'un élément [Ber, 02].

Les stréotypes de base d'UML sont :

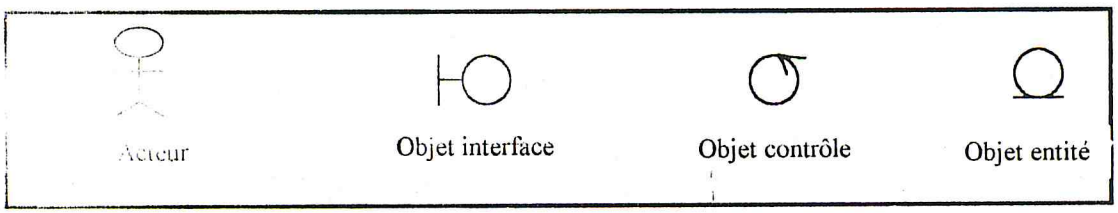


Figure III-4: Les stréotypes d'UML

- *Acteur* : représente les rôles des interlocuteurs u système.
- *Objet interface* : représente les limites du système.
- *Objet contrôle* : représente les classes effectuant des traitements internes au système.
- *Objet entité* : représenté les objet de domaine [Fan & al, 00].

III-3. 3. Diagramme des activités :

Ce diagramme permet de décrire le déroulement d'un cas d'utilisation particulier. Il est possible de décrire les acteurs responsables de chaque activités par l'utilisation des «couloirs d'activités» qui permettent de répartir graphiquement les différentes activités entre les acteurs opérationnels [Mul, 01]. Chaque activité est placée dans le «couloir» correspondant à l'acteur qui assume cette activité.

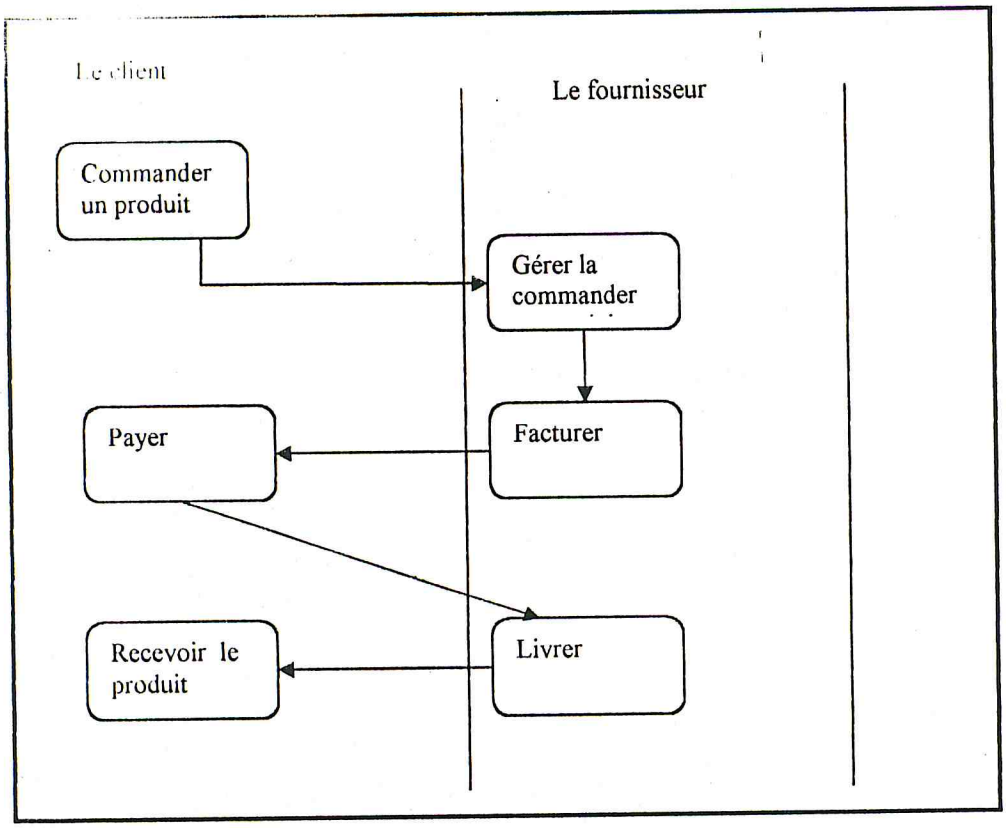


Figure III-5: Exemple d'un diagramme d'activité [Mul, 97].

III-3.4. Diagrammes de composants :

➢ **Composant**: élément physique qui représente une partie implémentée d'un système. Il peut être du code, un script, un fichier de commandes, etc. les composants présentent un ensemble d'interfaces [Ber, 02].

Les diagrammes de composants permettent de décrire l'architecture physique et statique d'une application en terme de modules : fichiers sources, librairie, exécutables etc. [Mul 01]. Ils décrivent les éléments physiques et leurs relations dans l'environnement de réalisation, ils montrent les choix de réalisation [Mul 97].

Les composants du systèmes : le sous système, le module, le programme et le sous programme, le processus et la tâche.

➢ **Module** : un système peut être décomposé en modules, chaque module correspond à un ensemble d'éléments physiques (fichiers, bibliothèques, sous ensembles de logiciel.) [Gab, 98]. La figure suivante donne le formalisme d'un module.

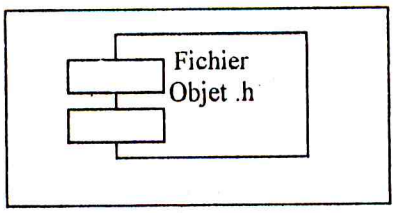


Figure III-6: Représentation d'un composant (fichier)

III-1-4 Diagramme de séquences :

Les diagrammes de séquences permettent de représenter les interactions entre objets en précisant la chronologie des échanges de messages. Ils peuvent être utilisés pour représenter les scénarios d'un cas d'utilisation donnée [Gab, 98].

➢ **Interactions**: modélisent un comportement dynamique entre objets [Mul 97]. Elles se traduisent par l'envoi de messages entre objets. Un diagramme de séquence représente une interaction entre objets, en insistant sur la chronologie des envois de messages [Ber, 02].

➢ **Les messages :** Les messages échangés sont représentés au moyen de flèches horizontales partant de l'émetteur vers le récepteur. L'ordre de l'envoi est donné par la positions sur l'axe vertical [Fan &al, 00].

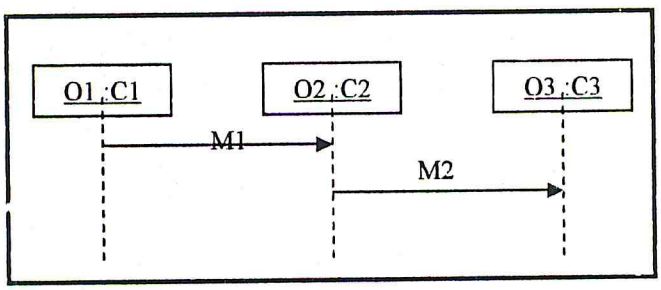


Figure III-7: Agencement de messages

Le diagramme de séquence distingue 5 types de messages prédéfinis qui sont présentés dans le tableau suivant :

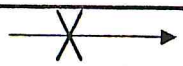
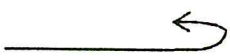
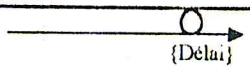
Type de message	La signification
	Message simple, exemple : à une passation de contrôle en mono tâche.
	Message asynchrone, pas de réponse attendue par l'émetteur.
	Message synchrone, réponse nécessaire du destinataire
	Message dérobant, le destinataire doit être à l'écoute
	Message minuté, l'émetteur est bloque pendant un laps de temps.

Tableau III-1 : Les différents types de messages

➢ **Période d'activation**

Correspond au temps pendant lequel un objet effectue une action, soit directement, soit par l'intermédiaire d'un autre objet.

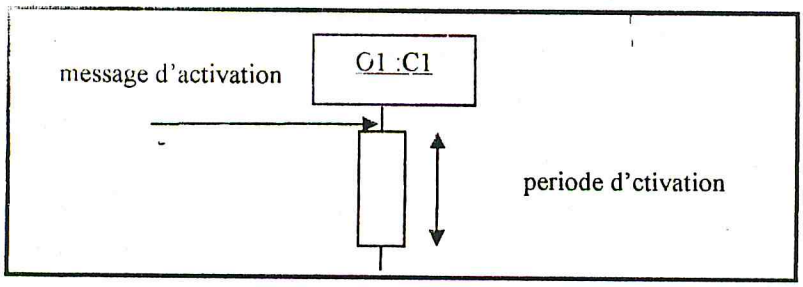


Figure III-8: Activation d'un objet de manière simple

III-3.6. Diagrammes de collaboration :

Les Diagrammes de collaboration montrent des interactions entre les objets et les acteurs. Ils permettent de représenter le contexte d'une interaction, car on peut y préciser les états des objets qui interagissent et peuvent être utilisé pour identifier les principaux objets [Gab 98]. La figure suivante présente le formalisme de base d'un diagramme de collaboration : un échange de message entre deux objets.

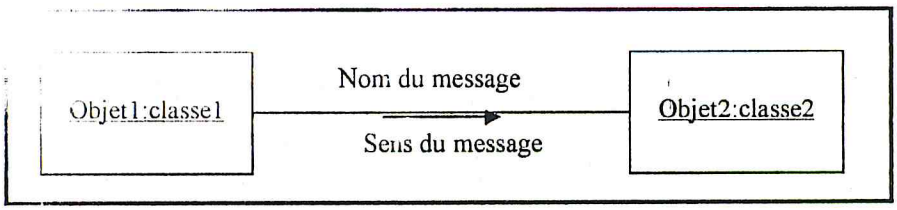


Figure III-9: Formalisme de base du diagramme de collaboration

III-3.7. Diagramme d'état-transition :

Un diagramme d'état-transition est un graphe constitué de nœuds représentants des états ainsi que des flèches représentant des transitions, portants des paramètres et des noms d'événements [Fan &al, 00]. Les diagrammes d'états permettent de définir le comportement d'un objet particulier vis-à-vis des sollicitations internes ou externes auxquelles il peut être soumis [Gab 98]. Ils permettent aussi de décrire l'évolution dans le temps les états des objet d'une certaine classe, les événements auxquels ils réagissent et les transitions qu'ils effectuent.

Les diagrammes d'états visualisent des automates (Figure suivante) d'états finis, du point de vue des états et des transitions [Mul, 97].

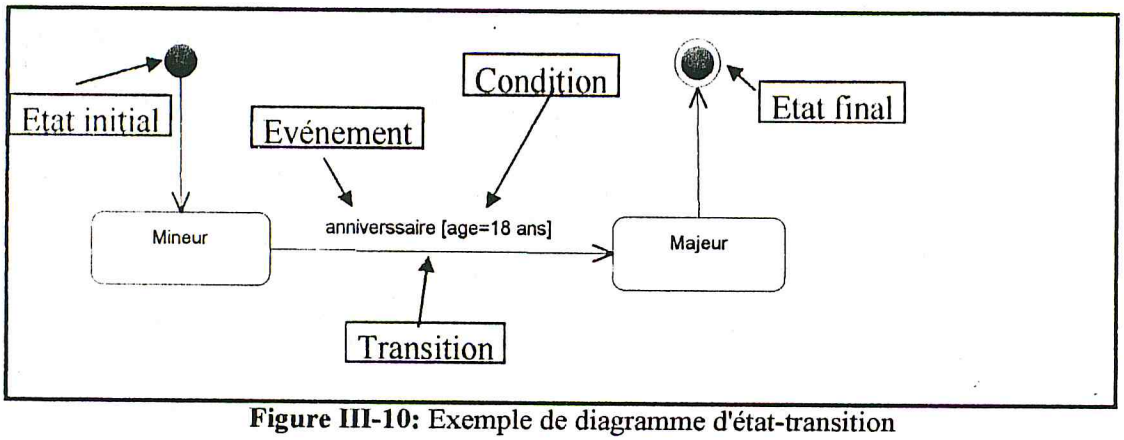


Figure III-10: Exemple de diagramme d'état-transition

➢ Points d'exécution des opérations

Il existe six points pour spécifier les opérations qui doivent être exécutées. Ces points sont dans l'ordre d'exécution :

- L'action associé à la transition d'entrée (op1).
- L'action d'entrée d'état (op2).
- L'activité dans l'état (op3).
- L'action de sorite d'état (op4).
- L'action associée aux événements internes (op5).
- L'action associée à la transition de sortie de l'état (op6).

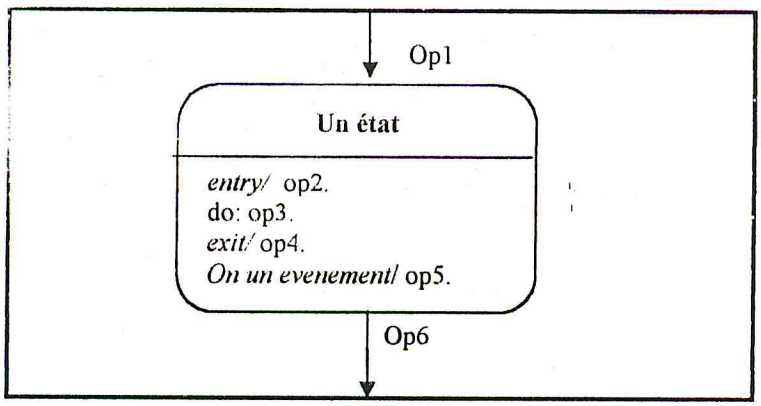


Figure III-11: Les points d'exécution pour un état [Mul, 97]

IV- Spécification des besoins

La première étape de la modélisation consiste généralement à déterminer précisément les besoins des utilisateurs du système. Il est ainsi particulièrement important de comprendre leurs besoins de manière globale, car une bonne informatisation passe de plus en plus par une vision intégrante, non parcellaire, qui s'optimise lorsqu'elle dépasse le point de vue d'une seule catégorie d'utilisateur [Mul, 01].

Pour ce faire, UML propose les cas d'utilisations, les diagrammes de séquences et les diagrammes de collaborations.

IV-1- Les cas d'utilisation

Les cas d'utilisation ont été formalisés pour déterminer les besoins des utilisateurs. Ils décrivent sous la forme d'actions et de réactions, le comportement d'un système du point de vue d'un utilisateur, ils permettent de définir les limites du système et les relations entre le système et l'environnement.

Un cas d'utilisation est une manière spécifique d'utiliser un système. C'est l'image d'une fonctionnalité du système, déclenchée en réponse à la stimulation d'un acteur externe [Mul, 01], l'ensemble des fonctionnalités d'un système est déterminé en examinant les besoins fonctionnels de chaque acteur.

a- Détermination des acteurs :

Les spécification débute par la recherche des acteurs (catégories d'utilisateurs) du notre système. Un acteur représente un rôle joué par une personne, un groupe de personnes ou par une chose qui interagit avec le système. par définition, les acteurs sont à l'extérieur de système [Mul, 01].

Les acteurs se recrutent parmi les utilisateurs du système et aussi parmi les responsables de sa configuration et de sa maintenance. Ils se répartissent dans les deux catégories suivantes :

- Concepteur ;
- Utilisateur.

Voici leur présentation graphique :

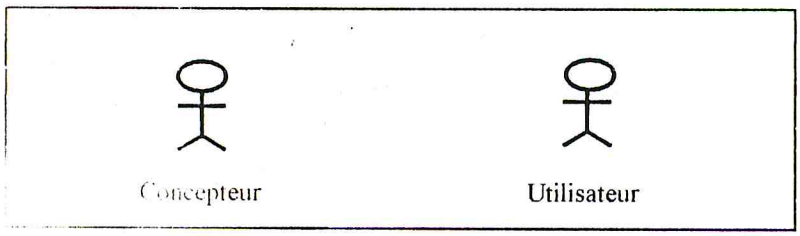


Figure III-14 : Représentation des catégories d'utilisateurs

La description de ces acteurs est la suivante :

- Un concepteur est une personne chargée de l'administration du système, de la conception et de la réalisation des logiciels produit par le système.
- Un utilisateur est la personne qui désire utiliser les logiciels fournis par le système.

b- Les cas d'utilisation des différents acteurs :

Les acteurs interagissent avec le système. L'étude des cas d'utilisation a pour objectif de déterminer ce que chaque acteur attend du système. La détermination des besoins est basée sur la représentation de l'interaction entre l'acteur et le système.

Un cas d'utilisation est une abstraction d'une partie du comportement du système. Il est instancié à chaque utilisation de système par une instance d'un acteur.

Après la description des acteurs il ressort que les catégories de besoins fonctionnels des acteurs se décomposent de la manière suivante :

➢ ***Cas d'utilisation des concepteurs :***

Le concepteur est très fortement impliquée dans le fonctionnent de notre système, il est considéré comme l'acteur principal du système. Les cas d'utilisation du concepteur sont :

- La conception qui consiste à concevoir des logiciels, les exécuter puis les faire passer à l'utilisateur pour les utiliser ;
- L'introduction des nouvelles règles et procédures de gestion ;
- Génération des interfaces, les structures de données et les masques de saisie ;
- Agrégation des informations saisies ;
- La consultation des données stockées dans une base de données ;
- La mise a jour des données, la mise a jour peut être un ajout, une suppression ou une modification des données.

On peut présenter les cas d'utilisation du concepteur par le diagramme suivant :

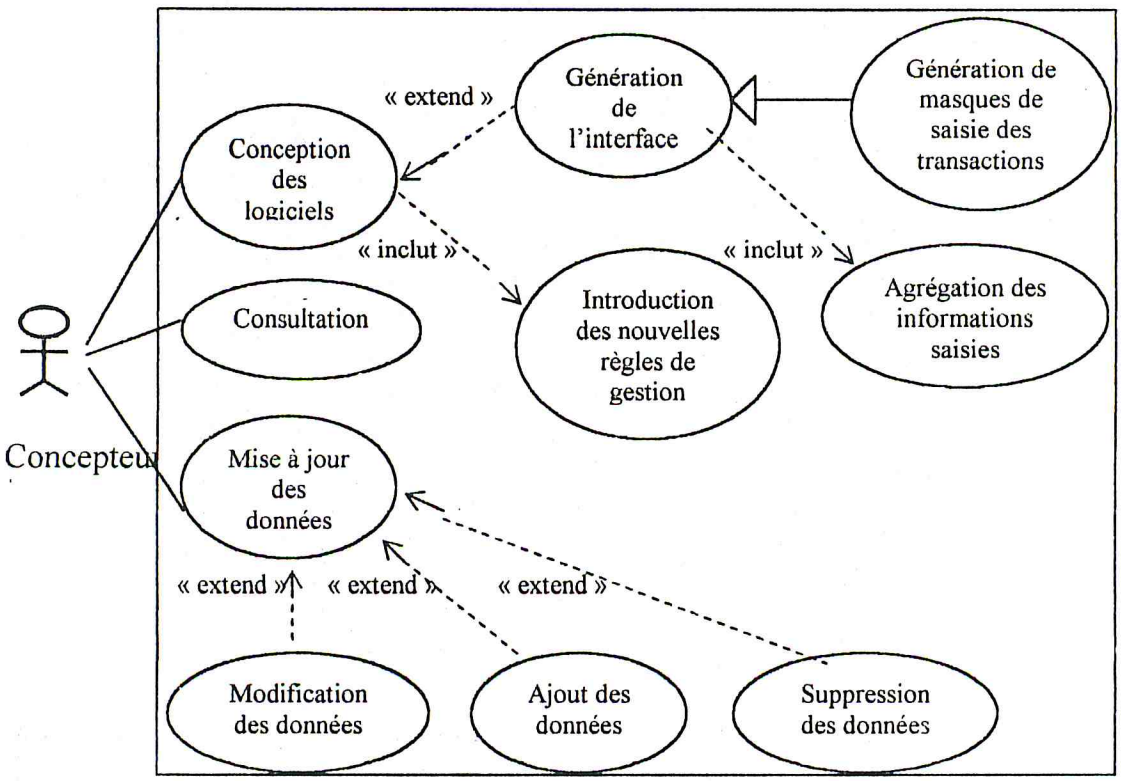


Figure III-15 : Diagramme des cas d'utilisation du concepteur

- *Cas d'utilisation de l'utilisateur :*

L'utilisateur final joue un rôle secondaire pour notre système. Les cas d'utilisation de l'utilisateur final sont :

La consultation des données stockées dans une base de données.

La mise a jour (ajout, suppression et modification) des données stockées dans une base de données. La présentation de ces cas d'utilisation et la suivante :

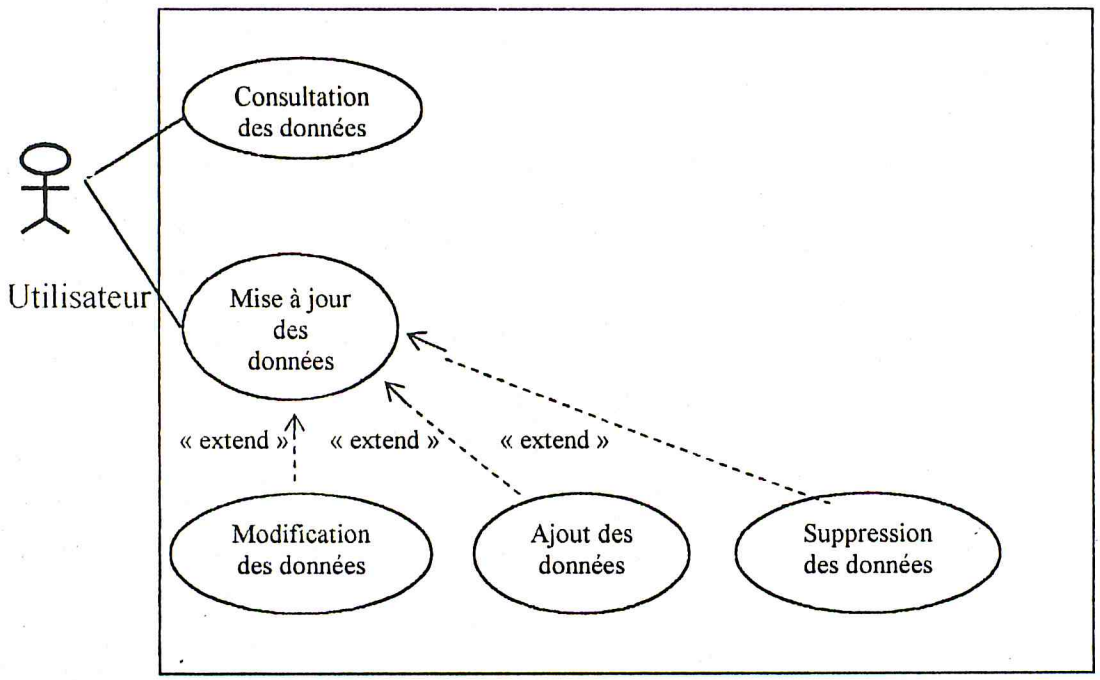


Figure III-16 : Diagramme des cas d'utilisation de l'utilisateur

c-Diagramme des cas d'utilisation

Chaque utilisation possible du système se traduit par une ellipse étiquetée avec le nom du cas d'utilisation, à laquelle est connecté le ou les acteurs concernés par cette interaction avec le système. Il est possible de raffiner les cas d'utilisation : une action complexe peut être réalisée par une combinaison d'actions plus simples (on dit que le cas d'utilisation complexe ((inclut)) les cas plus simples) [Bru, 01]. Pour notre système les actions :'conception' et 'consultation' sont des actions simples. L'action 'mise a jour' est une action complexe qui se décompose en trois actions : ajout, suppression et modification des données.

L'action 'conception des logiciels' inclut l'action 'introduction des nouvelles règles et procédures de gestion'.

Un acteur peut également participer à des relations de généralisation avec d'autres acteurs. Les acteurs 'fils' seront alors capables de communiquer avec les mêmes cas d'utilisation que le ou les acteurs parents.

L'acteur concepteur peut jouer le rôle d'un 'utilisateur' puisqu'ils partagent les cas d'utilisation : consultation et mise a jour, dans ce cas le concepteur est le fils et 'l'utilisateur' est le parent.

De même, nous trouvons des cas d'utilisation parents avec des cas d'utilisation fils. C'est le cas du cas d'utilisation 'génération de l'interface' qui est le parent du fils (cas d'utilisation fils)'génération de masques de saisie des transactions'.

Le diagramme des cas d'utilisation de notre système est présenter par la figure suivante

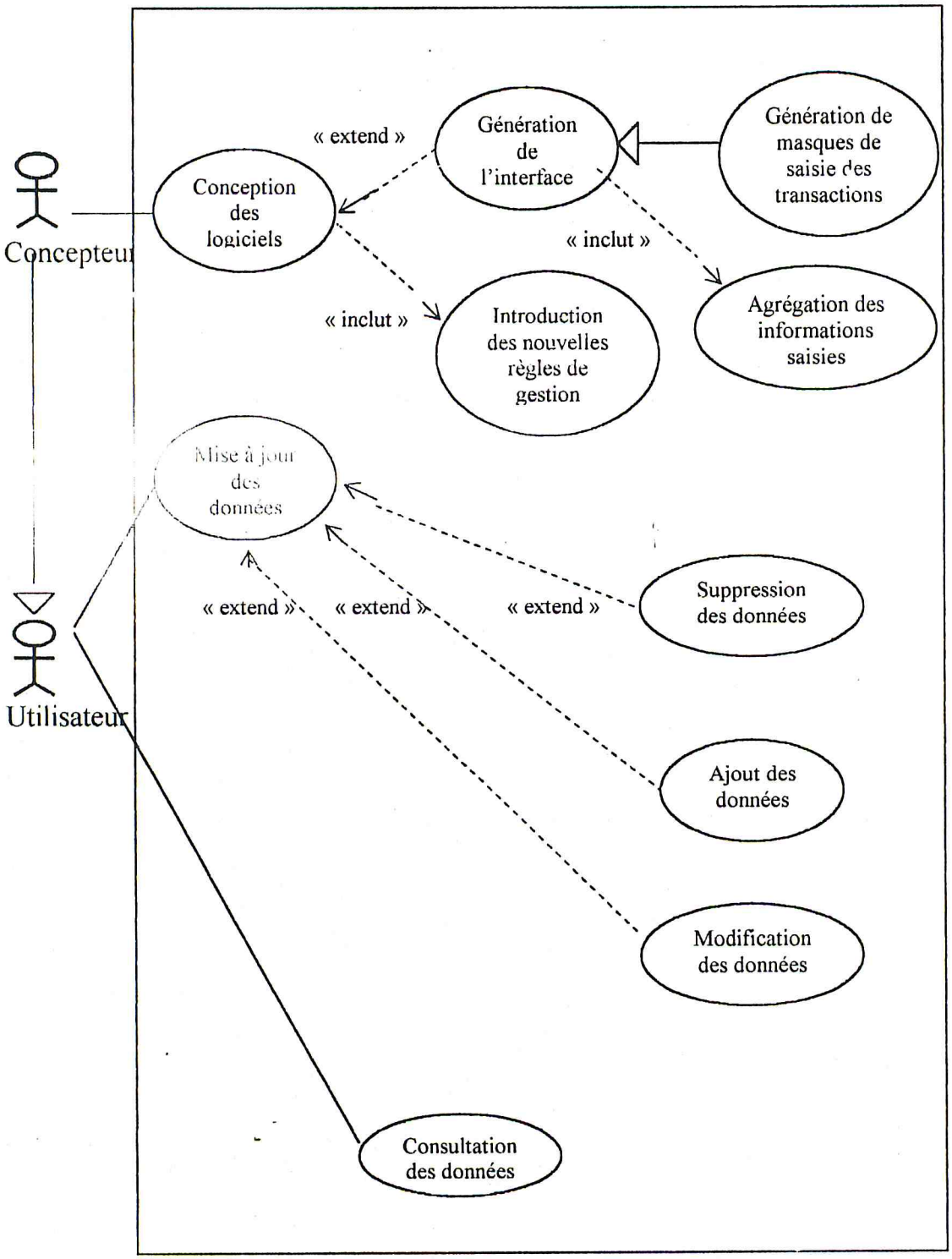


Figure III-16 : Diagramme des cas d'utilisation

IV-2- Les diagrammes de séquence

Les diagrammes de séquence montrent des interactions entre objets, ils nous permettent de bien schématiser les scénarios des cas d'utilisation.

La représentation se concentre sur la séquence des interactions selon un point de vue temporel. Ils sont en général, plus aptes à modéliser les aspects dynamiques des systèmes temps réel et des scénarios complexes mettant en ouvre peu d'objets [Mul, 01].

Une interaction modélise un comportement dynamique entre objets. Elle se traduit par l'envoi de message entre objets [Mul, 01] .

Un objet et matérialisé par un rectangle et une barre verticale, appelée ligne de vie des objets [Mul, 01]. Les objets communiquent en échangeant des messages représentés au moyen de flèches horizontales, orientées de l'émetteur du message vers le destinataire .

- **identification du concepteur:**

Scénario :

- Le concepteur se connecte au système et donne son mot de passe ;

- Le système vérifie l'identité du concepteur et autorise la connexion.

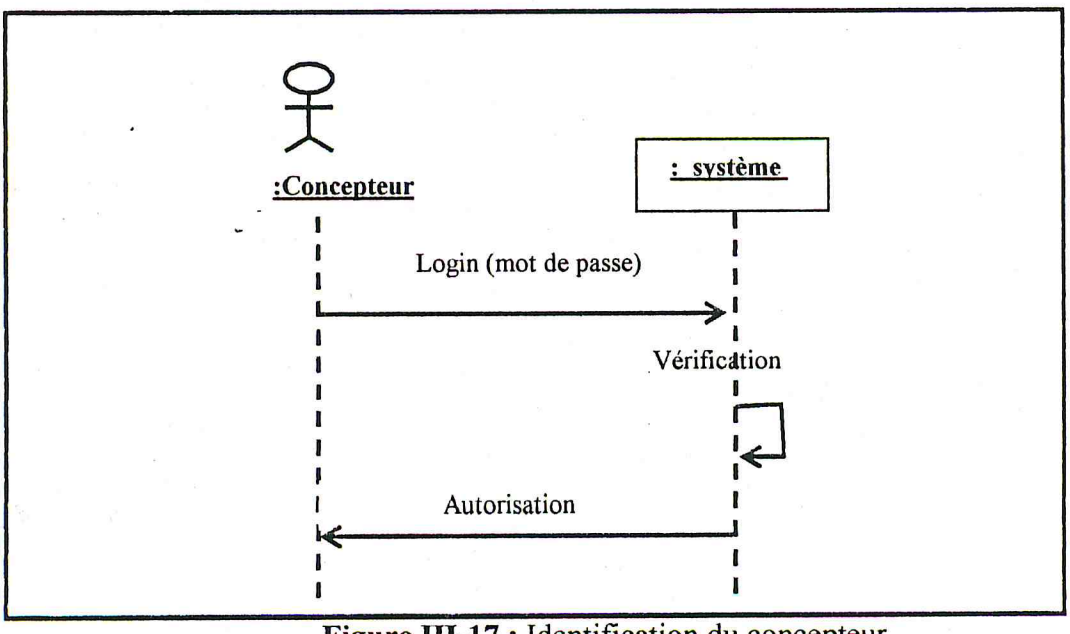


Figure III-17 : Identification du concepteur

➢ **Conception d'un logiciel :**

Un projet est un ensemble de fiches conçues par le concepteur, puis, exécutées.

Scénario :

- Un nouveau projet est crée par le concepteur;
- Le système ouvre des nouvelles Fiches de conception;
- Le concepteur génère l'interface selon les besoins spécifiés par l'utilisateur;
- Le concepteur relie les masques de saisie avec la base de données;
- Le concepteur demande l'enregistrement de son travail;
- Le système demande les noms des Fiches de conception;
- Le conception donne les noms des Fiches de conception et confirme leur enregistrement;
- Le système enregistre les Fiches de conception;
- Le système demande le nom du projet pour l'enregistrement;
- Le concepteur donne le nom du projet;
- Le système enregistre le projet;
- Le concepteur demande l'exécution du projet;
- Le système exécute le projet;
- Le système génère le projet exécutable;
- Le concepteur demande la sauvegarde de l'exécutable;
- Le système enregistre l'exécutable;
- Le concepteur ferme le projet .

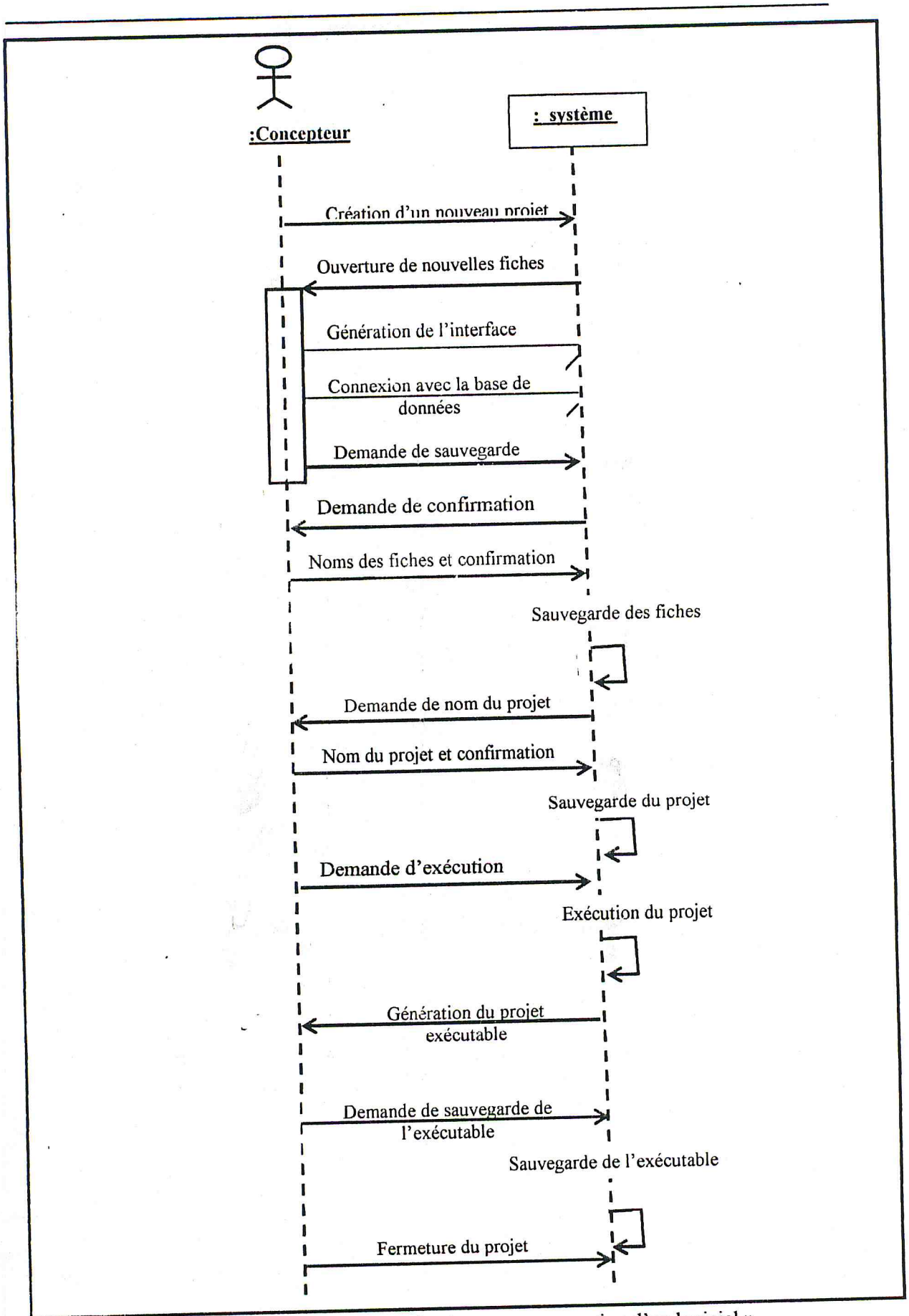


Figure III-18 : Diagramme de séquence « conception d'un logiciel »

➢ **Utilisation du logiciel :**

Scénario :

- L'utilisateur demande la liste des projets (logiciels) existants ;
- Le système fait la recherche de la liste des logiciels ;
- Le système affiche la liste des projets ;
- L'utilisateur choisit le logiciel du travail ;
- Le système ouvre le logiciel ;
- L'utilisateur consulte et fait la mise à jour des donnés ;
- Le système valide les modifications apportées à la base de données ;
- L'utilisateur ferme le logiciel.

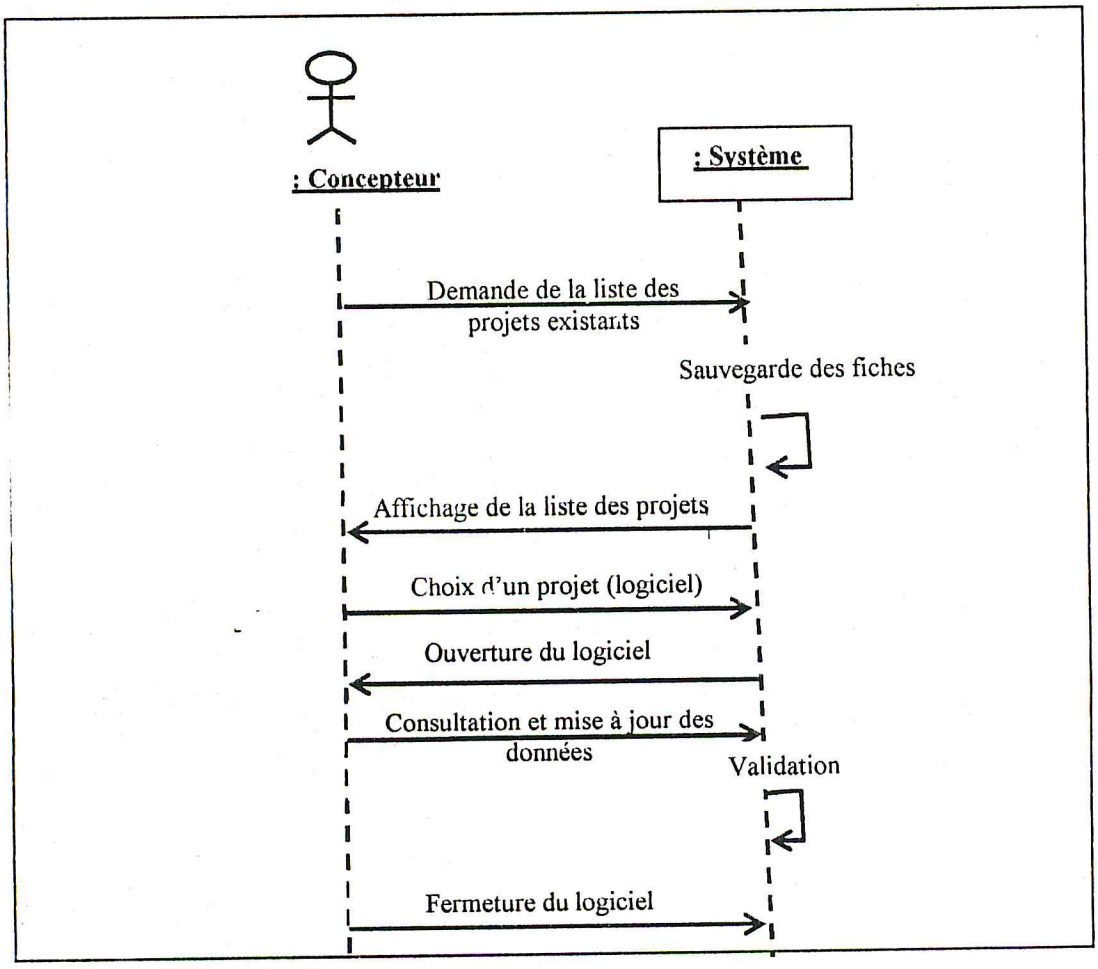


Figure III-19: Diagramme de séquence « Utilisation d'un logiciel »

➢ **Consultation des données :**

- L'utilisateur demande la consultation des données ;
- Le système recherche les données ;
- Le système affiche les données trouvées ;
- L'utilisateur demande l'impression ;
- Le système imprime les états.

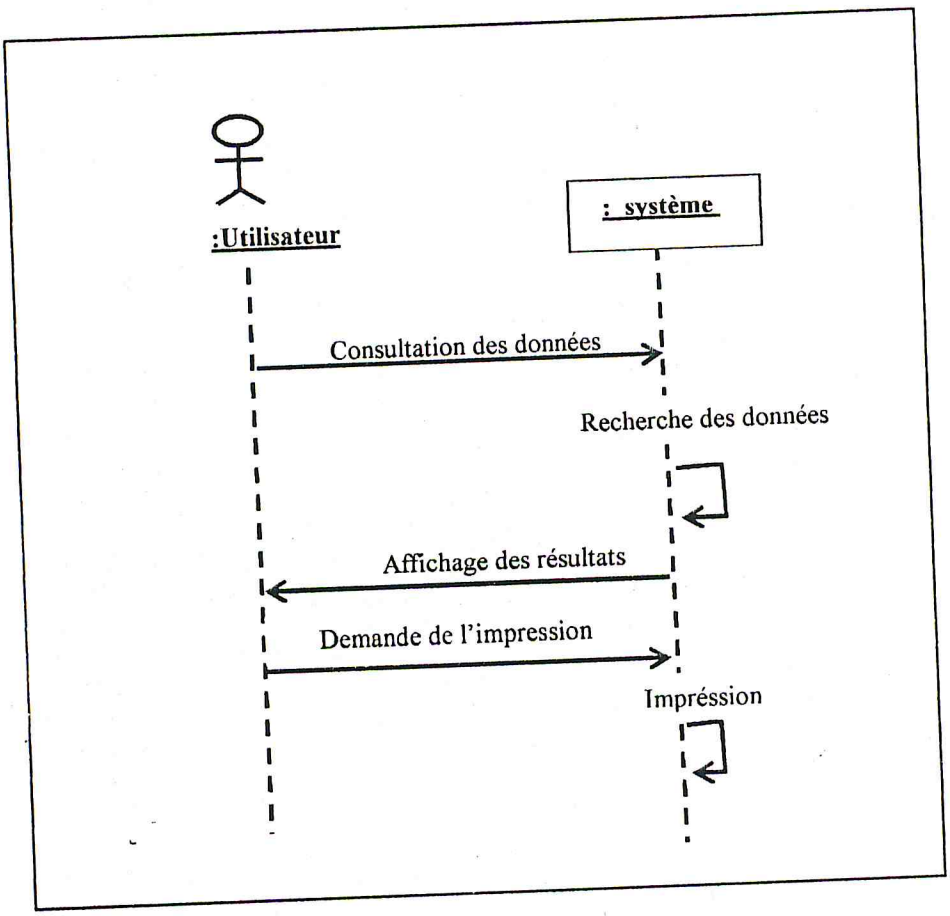


Figure III-20 : Diagramme de séquence « Consultation des données »

V- Conception

C'est la phase la plus importante du processus de développement d'un logiciel. Elle consiste à enrichir la description du logiciel de détails d'implémentation afin d'aboutir à une description très proche d'un programme.

Elle se déroule sur deux étapes : la conception générale et la conception détaillée.

La conception générale a pour but de décomposer le logiciel en modules et de préciser les interfaces et les fonctions de chaque module. A l'issue de cette étape, on obtient une description de l'architecture du logiciel et un ensemble de spécifications de ses divers composants. La conception détaillée fournit pour chaque module une description détaillée de la manière dont les fonctions du composant sont réalisées : algorithmes, représentation des données.

V- 1- La conception globale

V -1.1. Présentation du système proposé

La conception d'un outil doit d'abord déterminer la solution choisie à partir de l'étude des besoins de l'utilisateur.

La solution que nous avons choisit pour résoudre le problème posé dans le début de ce travail, consiste à combiner un SGBDR avec un système expert, ce choix est basé sur le fait qu'un SGBDR malgré sa puissance, ne peut prendre en charge toutes les contraintes des bases des données.

Ce qui oblige le concepteur d'un logiciel de programmer les contraintes non prises en charge par le SGBDR. Par contre un système expert peut grâce à la programmation prendre en charge toutes les contraintes même celles prises en charge par le SGBDR. Notre système est intégré car, le SGBDR garde la prise en charge des contraintes qui oblige le système à consulter la BD et le SE prend en charge les autres contraintes; ce qui décharge le concepteur des logiciels de la programmation de ces contraintes.

Pour concevoir notre système nous supposons que la BD existe déjà et nous fournissons à l'utilisateur une interface pour créer un nouveau projet, qui est un ensemble de fenêtres ; il peut alors déplacer des objets (masques de saisie, masques d'affichage…) pour réaliser l'interface de son logiciel et à l'aide d'un inspecteur d'objet (défini plus loin) il relie ces objets avec la BD.

Une interrogation avec la BD est possible en utilisant le SQL (défini plus loin).

A l'aide des interfaces spécifiques nous donnons la possibilité au concepteur pour faire entrer les données ou choisir des foncions qui seront traitées par notre système.

Après avoir exécuter le projet par le concepteur le système génère le logiciel qui sera utilisé par l'utilisateur final.

V -1.2. Caractéristiques du système

Un système est un objet dont les liens entre ses éléments conditionnent le fonctionnement. Notre système est un système flexible de gestion et d'aide à la décision, dont chaque terme correspond à une caractéristique à part entière. Ces termes sont les suivants:

- *Flexible:* La flexibilité du système est due au fait que l'utilisateur du système (concepteur) peut concevoir des logiciels sans avoir à réécrire les programmes, cette qualité vaut dans la mesure où les logiciels conçus ont le même modèle de spécification. Il s'agit donc de la capacité à concevoir des différents logiciels à l'aide d'un même système, capacité qui peut aller au delà de la simple conception d'un logiciel statique.
- *Gestion:* Notre système est spécialisé dans le domaine de la gestion des entreprises dont les principales fonctionnalités réalisées par le système sont: la consultation, la mise à jour et l'agrégation des données.
- *Aide à la décision:* Dans la vie d'une organisation, nombreuses sont les situations dont la complexité et/ou les enjeux invitent à rechercher une aide à la décision allant au-delà de l'utilisateur du "bon sens", de l'expérience ou de la mise en oeuvre de techniques de calcul élémentaire.

On présente souvent la décision comme le fait d'un individu isolé (le "décideur») exerçant librement un choix entre plusieurs possibilités d'actions à un moment donnée dans le temps [Roy, 93].

Dans ce contexte, décider ou, de façon plus large intervenir dans un processus de décision n'est qu'exceptionnellement trouver une solution à un problème. L'aide à la décision est l'activité de celui qui aide à obtenir des éléments de réponse aux questions que se pose un intervenant dans un processus de décision [Lev, 93].

C'est cette conception de l'aide à la décision qui nous a conduit à adopter notre solution tel que lors de la conception d'un logiciel de la part de l'utilisateur de notre système (concepteur), il doit trouver un moyen (interface) qui a comme but d'aider l'ordonnateur (utilisateur final) à prendre des décisions. Car les concepteurs sont des managers qui disposent de peu de temps à consacrer à l'apprentissage et à la programmation de nouveaux procédés de travail. Ils doivent pour cela disposer sans difficulté de l'information synthétique et fiable dont ils ont besoin. Le logiciel qu'ils manipuleront à cette fin doit être facile à utiliser.

V-1.3. Approche modulaire de conception

La consultation d'un logiciel est une suite d'itérations du genre division-réunion; il faut décomposer -diviser- pour comprendre et il faut composer -réunir- pour construire. Cela conduit à une situation paradoxale: Il faut diviser pour réunir [Mul, 01] !

Face à ce paradoxe, le processus de décomposition a été dirigé traditionnellement par un critère fonctionnel, il convient alors d'adopter une approche modulaire, approche basée sur le concept de modules. Un module est formé par une ou plusieurs tâches complémentaires ayant des caractéristiques communes.

L'étude des objectifs à atteindre et des tâches à accomplir nous permet de faire ressortir les modules suivants:

- **Module générateur d'interface :** Module dont les tâches primordiales sont: La génération des structures de données, la génération des masques de manipulation des données et la génération des masques d'affichage des transactions.

- **Module constructeur de requête SQL :** Ce module a comme tâche principale la construction de requêtes SQL à partir d'une BD existante.

- **Module système expert :** c'est le module le plus important de notre système, son rôle est la prise en charge des contraintes de la BD spécifiées dans le chapitre II et d'imposer les règles de gestion souhaitées.

- **Module interface utilisateur :** Ce module présente l'interface qui donne à l'utilisateur la possibilité de choisir et d'accéder au logiciel du travail et au concepteur la possibilité d'accéder à la partie du système consacrée à la conception des logiciels.

V -1.4. Architecture du système

L'architecture logicielle est le reflet des fonctions du système, l'architecture de notre système est présentée par le schéma décrit par la « figure III-21 ».

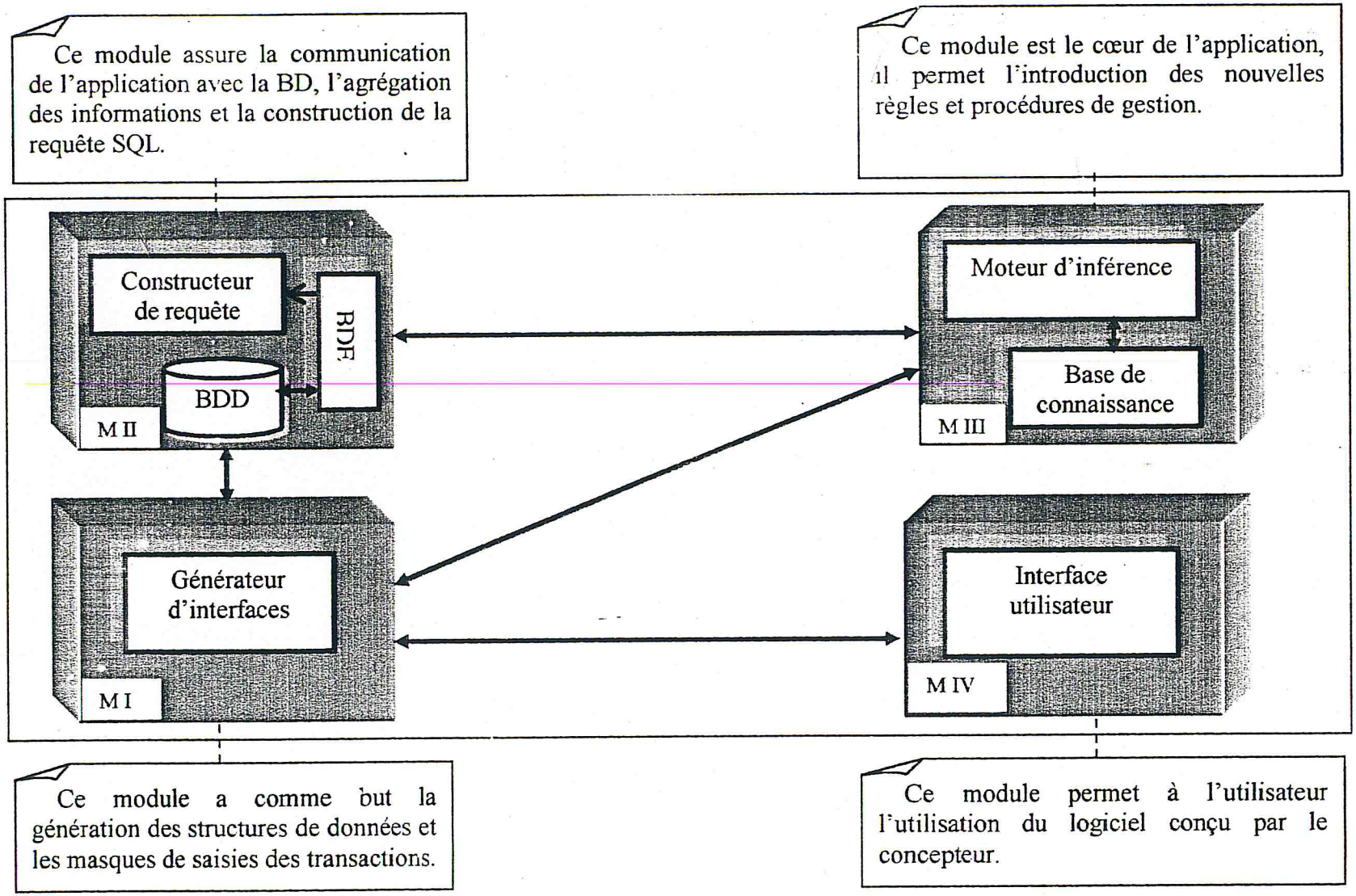


Figure III-21 : Architecture du système

V -1.5. Interaction entre modules

L'interaction n'est pas réservée au niveau d'un même module, mais s'étend à l'interaction entre différents modules.

Pour permettre à ces modules d'interagir, ils seront menus d'une interface qui relie les différents modules. Dans notre cas deux interfaces sont conçues pour répondre à ce besoin. La première interface est celle du module interface utilisateur, c'est via cette interface que l'utilisateur peut accéder au module générateur d'interface ou au logiciel conçu par le système.

La seconde interface appartient au module générateur d'interface, cette interface nommée « inspecteur d'objet » relie les trois modules : Générateur d'interface, constructeur SQL et système expert. Tel que après que le concepteur construit le SQL à l'aide du module « constructeur SQL », il relie les objets (masques de saisie, masques d'affichage,...), générée par le module générateur d'interface, avec le SQL construit. Puis à l'aide de l'inspecteur d'objet toujours, il relie ces objets avec le système expert et le système expert avec le SQL pour que le système interprète cette relation.

V -1.6. Diagramme de collaboration

Les fonctionnalités décrites par les cas d'utilisation sont réalisées tout d'abord par des collaborations d'objets du domaine. Par la suite, la réalisation des collaborations peut également faire intervenir des objets supplémentaires qui n'appartiennent pas au domaine de l'application mais qui sont nécessaires à son fonctionnement. Ces objets effectuent généralement l'interface entre le système et l'utilisateur, ou entre le système et un autre système.

Les diagrammes de collaboration sont des diagrammes qui représentent une vue dynamique du système. La notation permet de faire figurer un acteur dans un diagramme de collaboration afin de représenter des interactions déclenchées par un élément extérieur au système. Grâce à cette artifice, l'interaction est décrite de manière plus abstraite, sans entrer dans les détails des objets de l'interface utilisateur [Mul, 01].

Le premier message de l'interaction est envoyé par l'acteur, représenté soit par le symbole graphique des acteurs du modèle des cas d'utilisation, soit par un objet muni d'un stréotype qui précise sa qualité d'acteur. Un message se présente par une flèche placée à proximité d'un lien et dirigée vers l'objet destinataire du message au sein de l'interaction. En général, une séquence dans un diagramme de collaboration commence par le chiffre 1. Les numéros de séquence suivants sont incrémentés (2, 3...). Le diagramme de colaboration du système est représenté par la figure suivante :

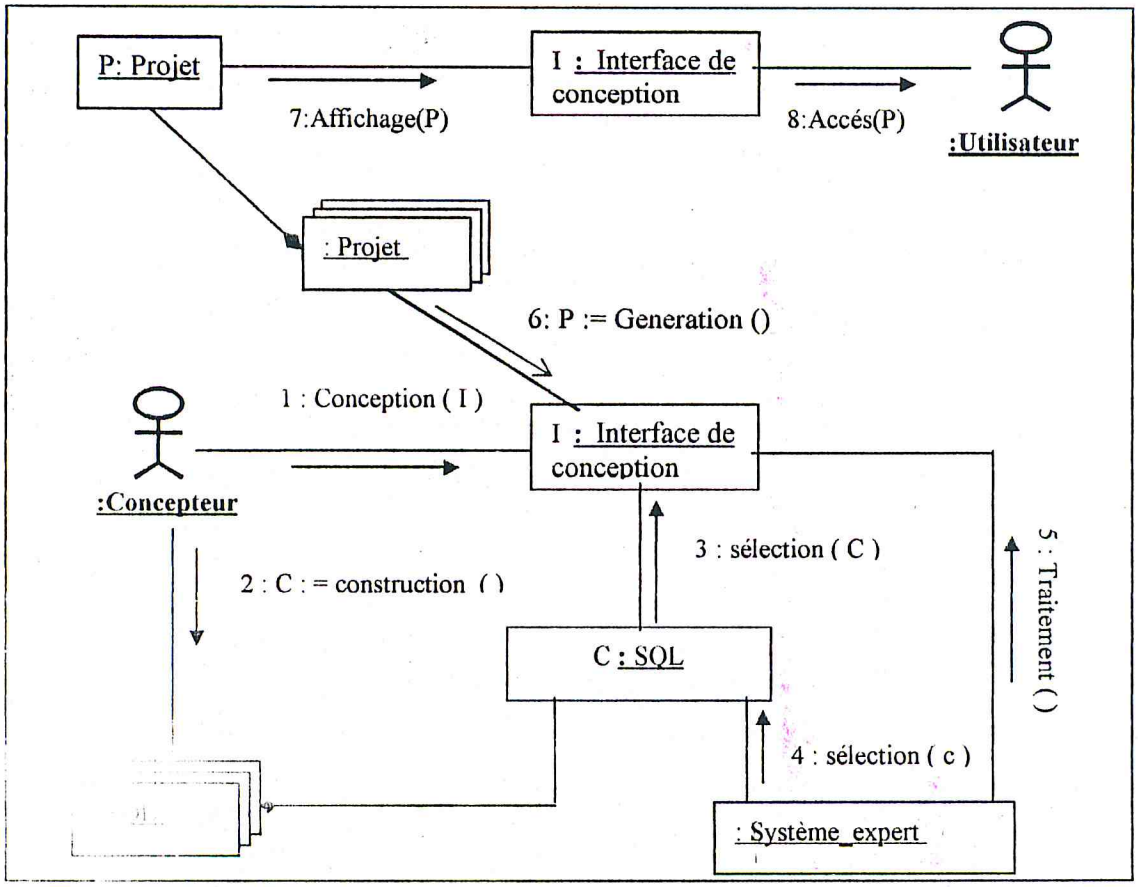


Figure III-22 : Diagramme de collaboration du système.

V -1.7. Les paquetages

Les modules représentés dans l'architecture du système peuvent être représentés par un diagramme de paquetage. Les paquetages offrent un mécanisme général pour la partition des modèles et le regroupement des éléments de modélisation. Chaque paquetage est représenté graphiquement par un dossier. Le paquetage de système est présenté par la figure suivante :

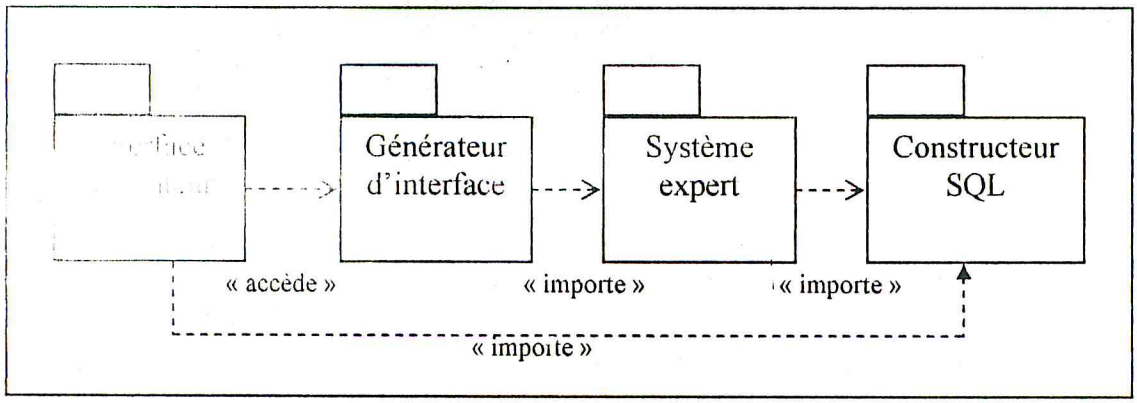


Figure III-23 : Diagramme de paquetage du système.

La dépendance « accède » Signifie que le paquetage source « Interface utilisateur » peut accéder au paquetage destinataire « Générateur d'interface ».

La dépendance « importe » signifie qu'un élément du paquetage source peut importer des élément du paquetage destinataire c-à-d le « Générateur d'interface » utilise les données fournies par le « Système expert » et ce dernier utilise le fichier SQL construit par le « Concepteur SQL ».

V -1.8. Les composants implémentés

Les diagrammes de composants décrivent les composants et leurs dépendances dans l'environnement de réalisation. Les diagrammes de composants sont des vues statiques de l'implémentation des systèmes qui montrent les choix de réalisation.

Dans le diagramme de composant chaque classe est réalisée par deux composants: la spécification et le corps, la spécification contient l'interface de la classe alors que le corps contient la réalisation de cette même classe. En Delphi, la spécification correspond à un fichier avec un suffixe (.dfm) et le corps à un fichier avec un suffixe (.pas).

*Exemples de classes implémentées :*Inspecteur, composant, field, champ,BD…

Un composant est un élément physique qui représente une partie implémentée d'un système. Les éléments qui sont utilisés pour construire le système sont représentés dans le paquetage suivant :

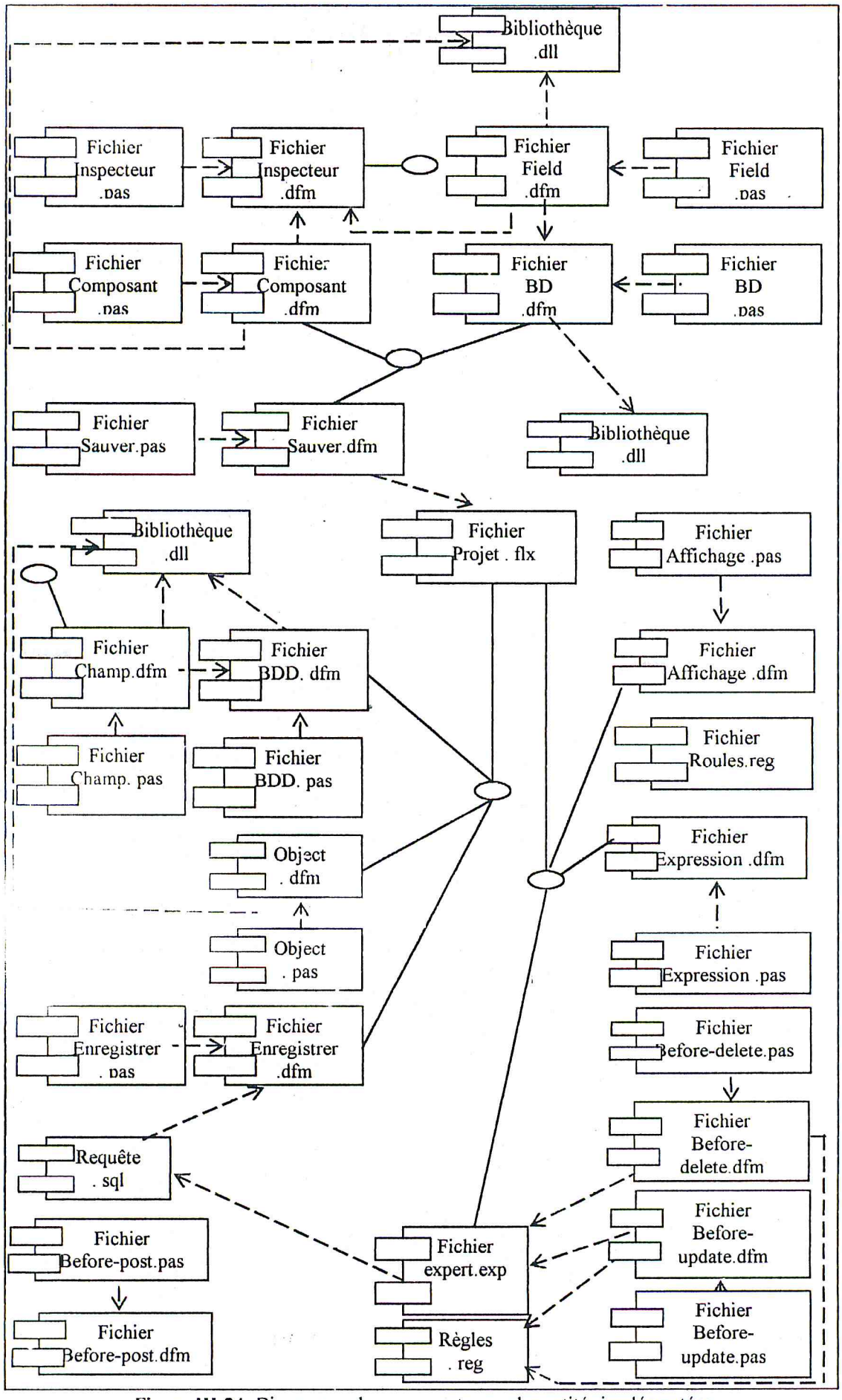


Figure III-24: Diagramme de composants pour les entités implémentées

V -2- Conception détaillée

C'est la partie la plus importante de ce document, elle décrit comment les cas d'utilisation et les opérations globales sont réalisés au sein du module.

En général, un module est réalisé par un ensemble d'objets coopérants entre eux, et toute opération sur le module se traduit par un ensemble d'actions impliquant les objets qui les composent. La partie détaillée comporte donc aussi des classes et les machines d'états associées, ainsi que des collaborations et interactions explicitant leur comportement.

V -2-.1.Module générateur d'interface

La part du code d'une application réservée à l'interface n'a cessé de prendre de l'importance comparée à celle destinée à implémenter les fonctionnalités proprement dites de l'application. Les concepteurs cherchent de plus en plus des moyens qui permettent une production rapide d'interfaces et surtout qui soit la plus indépendante possible de la plate-forme pour faciliter la portabilité et évolutive pour faciliter sa mise à jour et son amélioration. La diminution du coût de développement des interfaces et la rapidité de leur production sont devenues une exigence du marché.

Pour ces raisons, le premier module de notre système est un générateur d'interface qui a comme rôle la génération des interfaces du projet selon ce qui est définie au préalable par le concepteur.

❖ Architecture du module générateur d'interfaces

L'architecture du module générateur d'interfaces est présentée par le schéma décrit par la « figure III-25 » suivante :

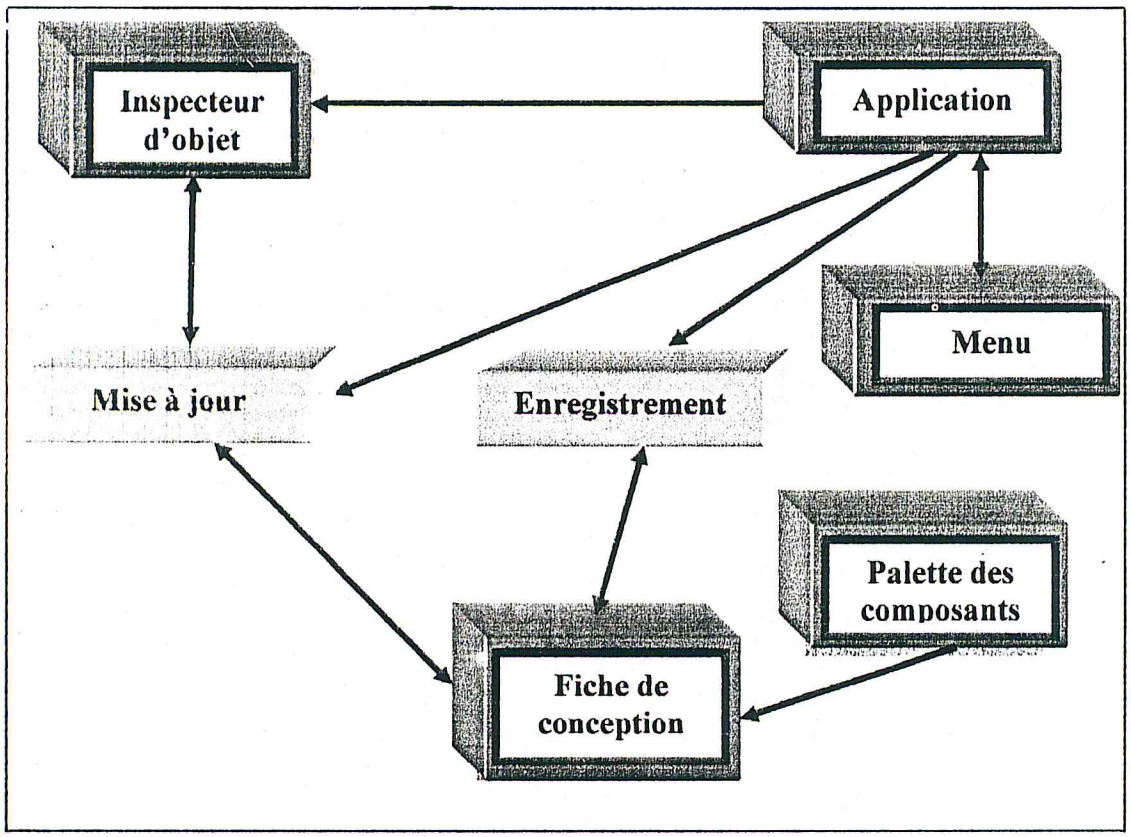


Figure III-23 : Architecture du module générateur d'interfaces

➢Palette des composants

Le but principal de la palette des composants est de permettre la construction visuelle d'interface, en permettant au concepteur de placer directement les objets de représentation (composants) pour composer l'apparence de son interface. La palette des composants est un ensemble de boutons avec les images des composants correspondants. Pour chaque bouton, nous affectons une classe de création d'un composant, son rôle est de créer un composant et de lui affecter un nom. Il faut donc pouvoir déterminer leur apparence et le comportement que cet interface aura, suite aux actions du concepteur ou aux sollicitations du [illegible] channel de l'application.

Un composant est crée juste après un simple click sur le bouton correspondant.

➢**Fiche de conception**

Après avoir crée le composant en cliquant sur le bouton de la palette des composant, le concepteur clicque sur la fiche de conception pour l'affichage de ce composant. Le concepteur voit un objet affiché sur la surface de la fiche et doit donc pouvoir le désigner directement et sans ambiguïté par le pointeur.

La sélection ou la désignation des composants est l'une des interactions qui jouent un grand rôle dans l'appropriation du composant par le concepteur. En plus de son rôle qui consiste à sélectionner des composants pour leur appliquer des opérations ou des transformation, la sélection amène l'inspecteur d'objets à prendre en compte le composant sélectionné. Un composant est sélectionné par une désignation directe de sa représentation graphique visible ou par son nom.

Dans le cas de la sélection, il arrive souvent que les représentations graphiques de certains composants se chevauchent sur d'autres composants, créant ainsi une ambiguïté quand le concepteur essaie de sélectionner l'un d'entre eux. Pour enlever les ambiguïté lors de la sélection, les composants sont sélectionnés uniquement en désignant leurs quatre angles et non par les surfaces des composants.

Si les composants non visuels de la BD ne peuvent être redimensionnés les autres composants eux peuvent être déplacés et redimensionnés par le concepteur.

En manipulation directe, le curseur est utilisé comme un pointeur pour désigner des composants ou sélectionner d'autres, mais il est aussi utilisé comme un outil de réalisation de l'action du concepteur. Il peut devenir un prolongement de la main du concepteur pour déplacer des composants ou les redimensionner.

L'un des principes d'une interface de manipulation directe indique que les composants doivent avoir une représentation permanente à la surface de l'écran. La présence d'une représentation est donc synonyme de l'existence d'un composant [Kol, 97], sauf si le concepteur l'a délibérément caché ou le composant est non visuel après l'exécution.

➢ **Inspecteur d'objets**

Concevoir une interface homme –machine n'est pas le simple habillage d'une application. Cela implique aussi bien la manipulation de cette interface.

Quand nous sélectionnons un composant par désignation directe, nous nous posons la question de savoir comment le relier avec les autres composants. Tel que, un composant du domaine doit avoir une connaissance du domaine embarquée qui lui permet de réagir aux différentes sollicitations de manière cohérente avec le domaine. Cette réaction est déterminée par l'ensemble des relations et des contraintes qui le lient aux autres composants.

Exemple : Le composant TTable est relié avec la BD via l'inspecteur d'objet.

L'inspecteur d'objets est une interface qui joue le rôle de l'intermédiaire entre les composants et le système, il permet au concepteur de comprendre l'état de son système et d'obtenir suffisamment d'informations sur les composants et leurs relations entre eux ou avec la BD. La réaction de l'inspecteur d'objets aux actions directes du concepteur sur les composants ou la fiche constitue une partie importante des fonctionnalités de l'inspecteur d'objets. De même, quand une action est exécutée sur l'inspecteur d'objet, un changement de la représentation graphique des composants doit refléter le nouvel état des composants et mettre en évidence le résultat de l'action.

Quand le concepteur affiche un composant, il peut lui donner un nouveau nom, le référencer par ce nom comme argument pour le relier avec d'autre composants.

Dans certaines tâches, pour produire un comportement cohérent des composants, le système doit pouvoir déterminer si un objet vérifie certains propriétés, même quand celle-ci n'a pas été directement précisée. Il doit être capable de déduire la présence ou non d'une propriété à partir de l'ensemble des relations du composant avec les autres composants.

Les principales propriétés qui nous avons traitées pour la génération d'interfaces sont représentées par le tableau suivant :

Propriétés	Composants	Rôle
Left, Top	Tous les composant	Déterminent la position du composant sur la fiche.
With, Hieght	Les composants visuels	Déterminent la hauteur et la largeur du composant.
Color	Panel	Définit la couleur de fond du composant.
Font	Memo, DBLookupListBox, DBLookupComboBox, DBListBox, DBComboBox, DBCheckBox, DBRadioGoup	Définit la fonte.
Hint	Tous les composants	Spécifier le texte de conseil du composant.
Caption	Label, Panel, Button	Libellé du composant.
Mask edit	MaskEdit	Contient le masque représentant le texte valide d'un contrôle de saisie masqué.
Font de titre	DBGrid	Décrit la fonte utilisée pour dessiner les titres de colonne.
Lecteur seul	Les composants visuels d'accès à la base e données	Détermine si l'utilisateur peut modifier la valeur du champ.
Data source	Les composants d'accès à la base de données	Identifie la liaison avec l'ensemble de données.
Pas[illegible] Char	DBEdit	Indique le caractère à afficher à la place des caractères saisies.
DataField	Les composants visuels d'accès à la base e données	Spécifie le champ dont le contrôle de saisie affiche les données.
Key field	DBLookupListBox, DBLookupComboBox	Identifie le champ de l'ensemble de données qui doit correspondre à la valeur du champ DataField.

Nom	Tous les composants	Indique le nom du composant
List source	DBLookupListBox, DBLookupComboBox	Identifie une source de données pour les données affichées dans le contrôle de référence.
List field	DBLookupListBox, DBLookupComboBox	Identifie les champs dont les valeurs sont affichées dans le contrôle de référence.
Items	DBComboBox, DBRadioGoup	Donne la liste des boutons radio.
Value checked	DBCheckBox	Spécifie la valeur de champ qui correspond à l'état activé de la case à cocher.
Values	DBRadioGoup	Détermine la valeur des boutons.
Data set	DataSource	Spécifie l'ensemble de données qui utilise le composant source de données comme canal vers les contrôles orientés données.
Active	Table, Query	Indique si un ensemble de données est ouvert.
Master fields	Table	Spécifie les champs de la table maître.
MasterSource	Table	Spécifie le nom de la source de données de la table maître.
IndexName	Table	Identifie un index secondaire de la table.
SQL	Query	Contient le texte de l'instruction SQL exécutée par la requête.
Etat de sortie	FastRepport	Définie les états de sortie.
Filter	Table, Query	Spécifie le texte du filtre actif de l'ensemble de données.

Tableau III-2 : Les principales propriétés des composants.

➢ Menu de manipulation des projets

Quand le concepteur veut créer un nouveau projet, il doit passer par le menu de manipulation des projets. Après la conception du projet une sauvegarde de ce dernier aura lieu. Ainsi, il doit exécuter le projet et sauvegarder l'exécutable. Pour modifier un projet une ouverture de ce dernier est possible.

❖ Paquetage du module générateur d'interfaces

Un paquetage est un regroupement d'éléments selon un critère purement logique. L'objectif de la décomposition en paquetage est d'avoir une cohérence forte entre éléments d'un même paquetage. En général, le contenu d'un même paquetage est constitué d'éléments qui forment un tout cohérent et qui sont proches sémantiquement [Mul, 01]. Les éléments du paquetage du module générateur d'interfaces sont représentés par la figure suivante :

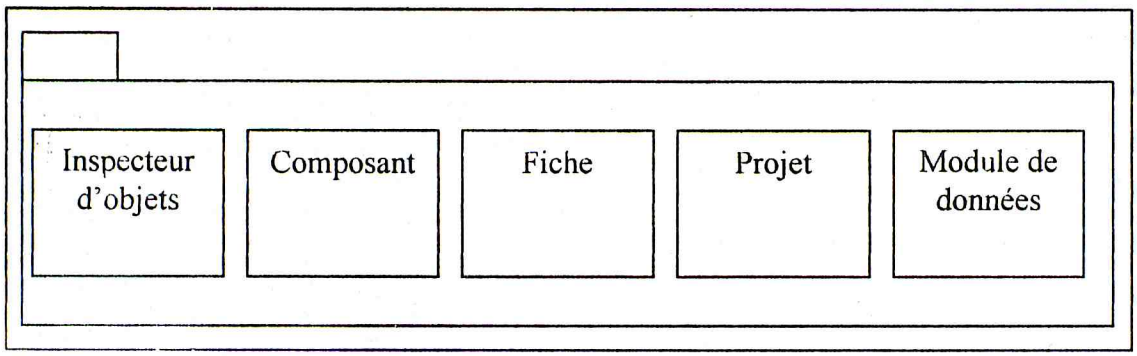


Figure III-24 : Paquetage du générateur d'interfaces.

❖ Diagrammes de collaboration

Les diagrammes de collaboration représentent un ensemble de rôles joués par des objets dans un contexte particulier, ainsi que les liens entre ces objets. Ils montrent également des interactions entre ces objets à travers la représentation d'envois de messages.

Les diagrammes de collaboration du module générateur d'interface sont représentés par les figures 25, 26, 27.

➢Nouveau projet :

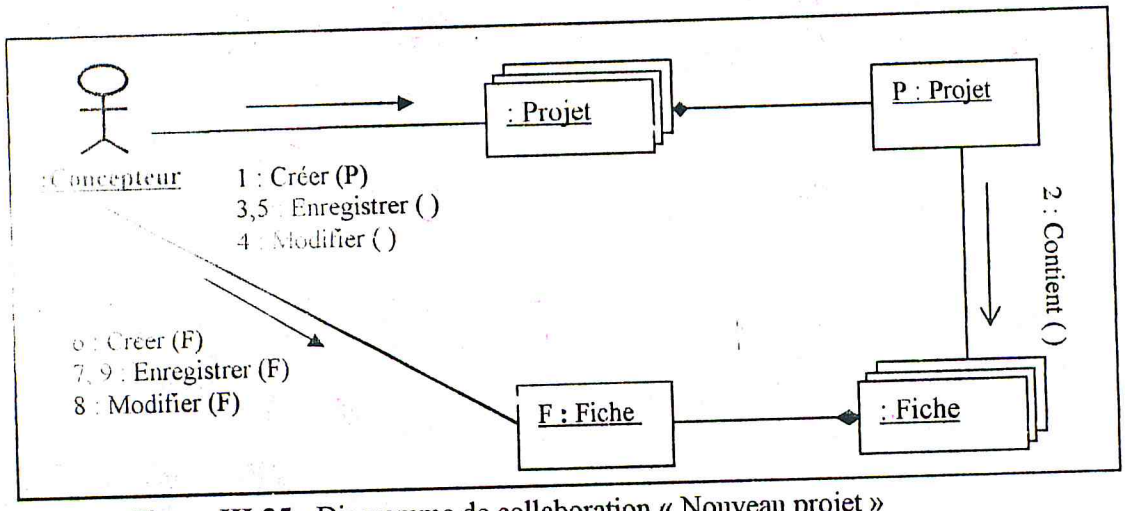


Figure III-25 : Diagramme de collaboration « Nouveau projet »

➢Opérations sur un composant

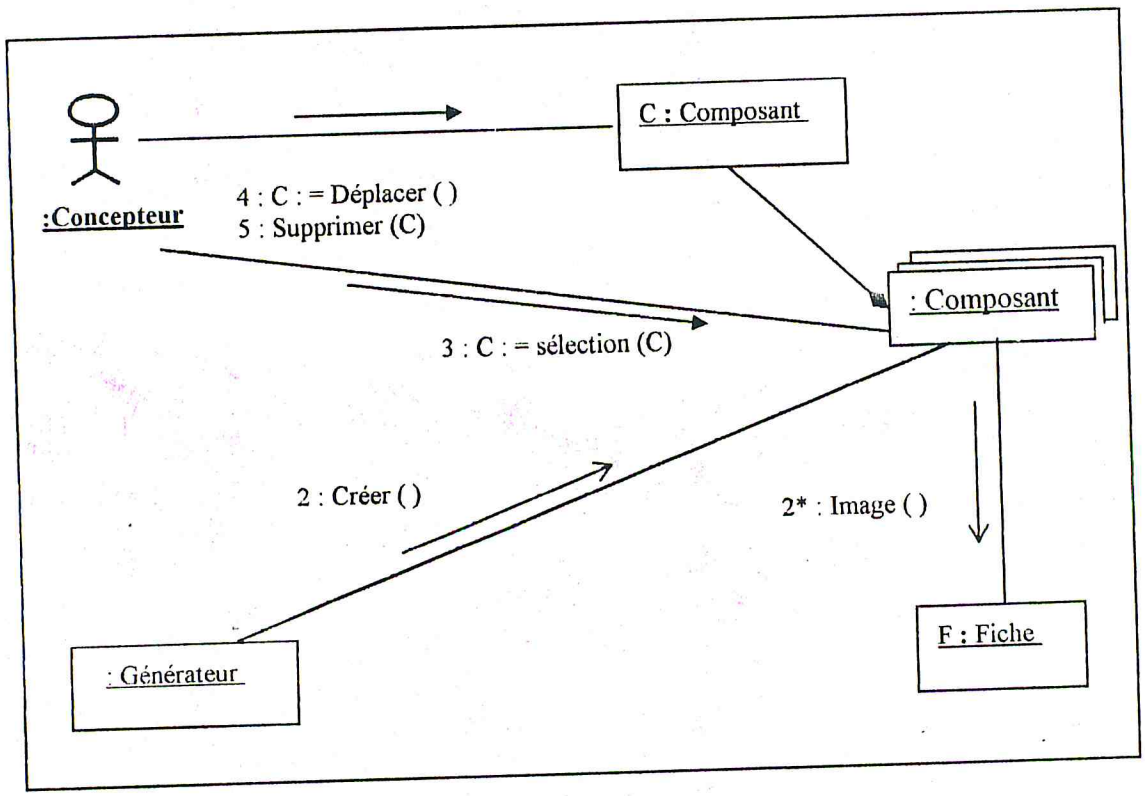


Figure III-26 : Diagramme de collaboration « Opérations sur un composant »

➢ **Fonctionnement de l'inspecteur d'objets**

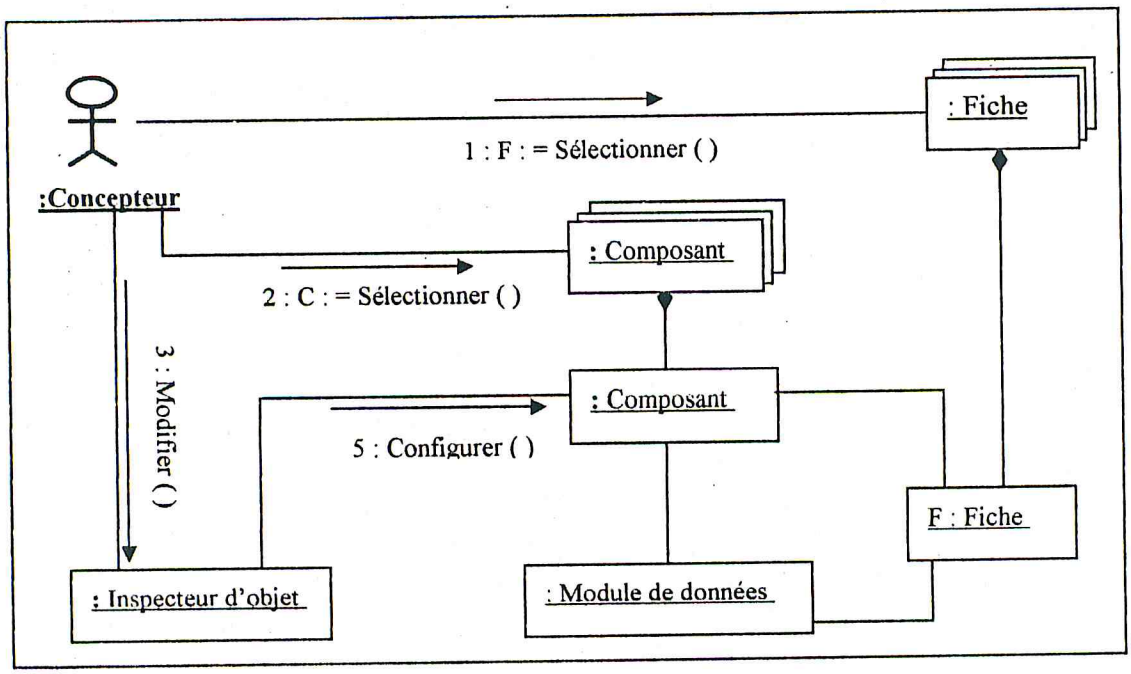


Figure III-27 : Diagramme de collaboration « Fonctionnement de l'inspecteur d'objets »

➢ **L'état de sortie:**

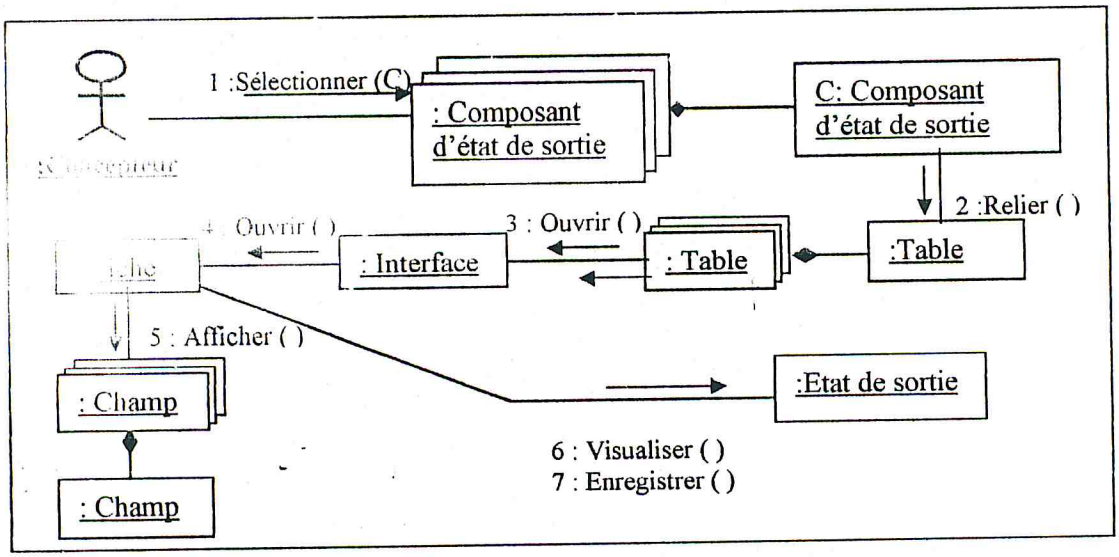


Figure III-49: Diagramme de collaboration de l'état de sortie

❖ **Diagramme de classes :**

Le diagramme de classe suivant représente les éléments de modélisation qui doivent être implémentés, d'une manière synthétique :

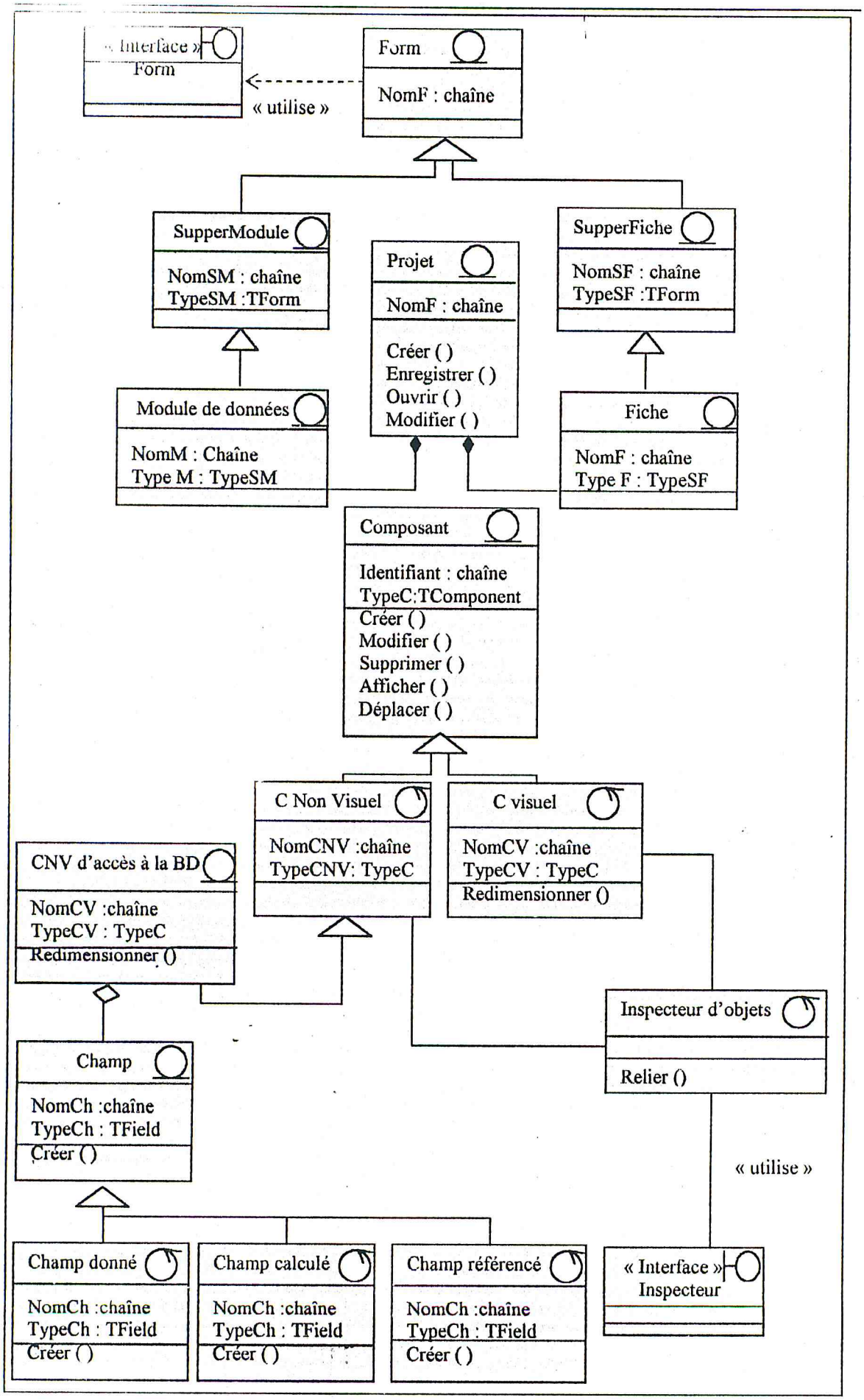


Figure III-28 : Diagramme de classes de générateur d'interfaces

*** Les composants**

Certains composants sont visuels, d'autres non visuels. Le concepteur peut avoir l'aspect définitif d'un composant visuel durant la phase de conception.

Parmi les composants visuels, on peut citer les contrôles de saisie (ou contrôles d'édition), les boutons, les boîtes liste, les labels, et cetera. La plus part de ces composants sont des composants visuels. Dans la mesure du possible, ces composants présentent en mode conception l'aspect qu'il auront durant l'exécution du logiciel.

Les composants non visuels travaillent en coulisses pour accomplir certaines tâches spécifiques. A titre d'exemple nous mentionnons les composants de base de données et main menu.

❖ **Diagrammes d'états-transitions et d'activités**

Les diagrammes d'états-transitions visualisent les automates d'états finis, du point de vue des états et des transitions.

Un diagramme d'activité est une variante des diagrammes d'état-transition. Dans un diagramme d'état-transition, les états et les transitions sont mis en avant alors que dans un diagramme d'activités, ce sont les activités et les transitions qui sont mises en avant.

Les diagrammes d'état-transition et d'activités du module générateur d'[illegible] sont représentés par les figures suivantes :

➢Diagramme d'états pour un composant crée

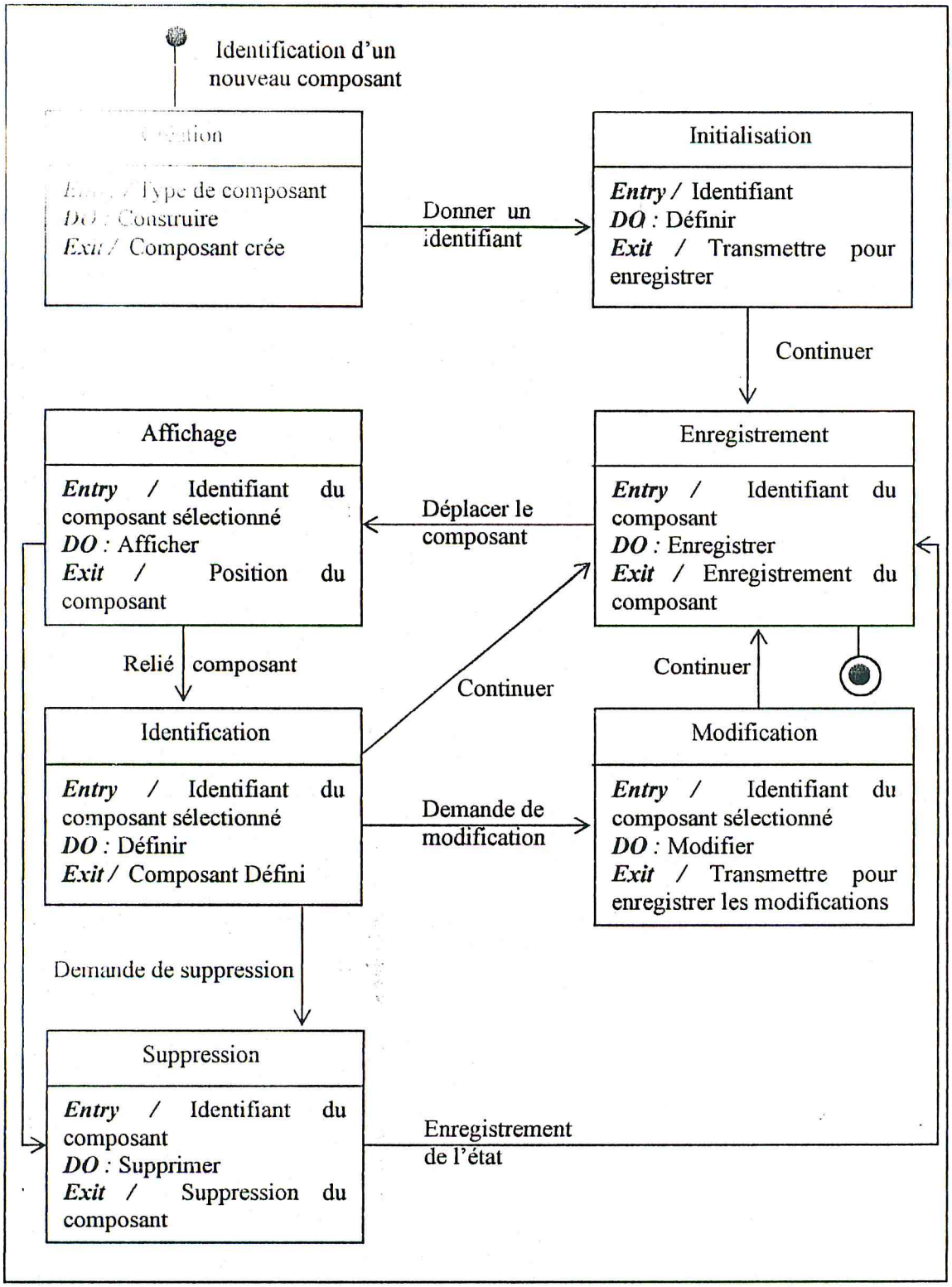


Figure III-29 : Diagramme d'états pour un composant crée

➢ **Diagramme d'états pour la création d'un champ:**

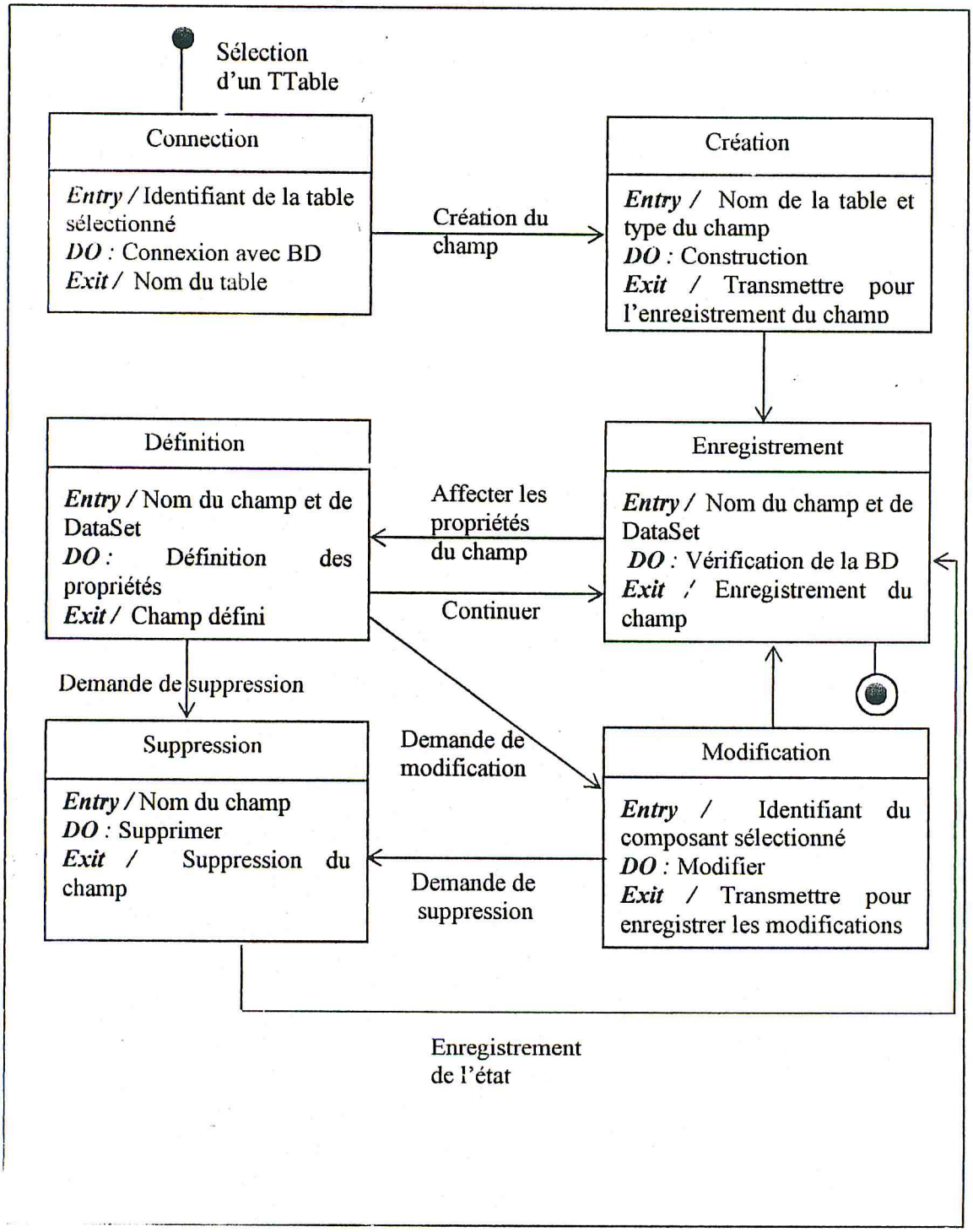


Figure III- 30 : Diagramme d'états pour la création d'un champ

➢Diagramme d'états pour la création d'un projet

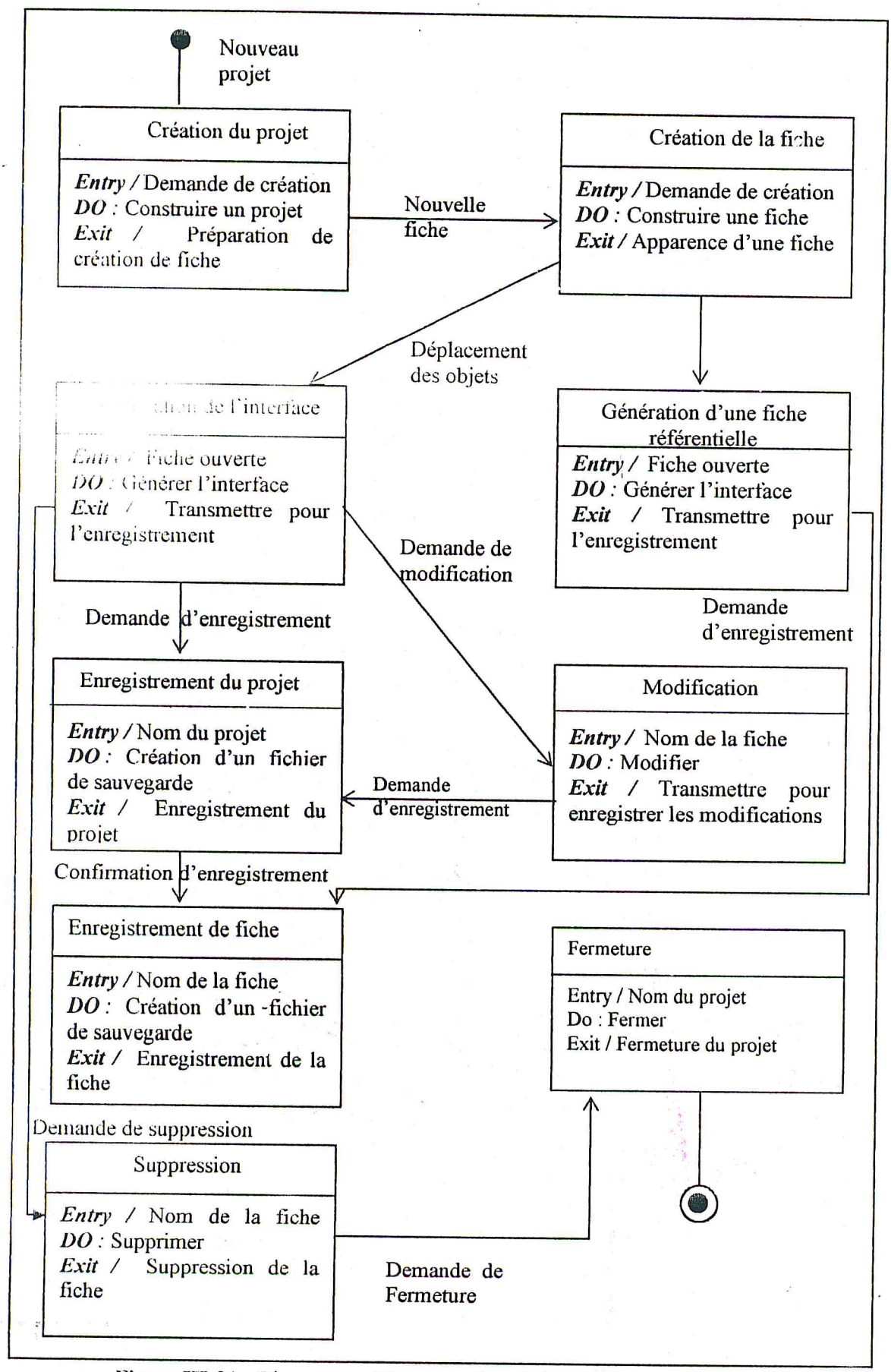


Figure III-31 : Diagramme d'états pour la création d'un projet

➢ Diagramme d'activités pour la création d'état de sortie

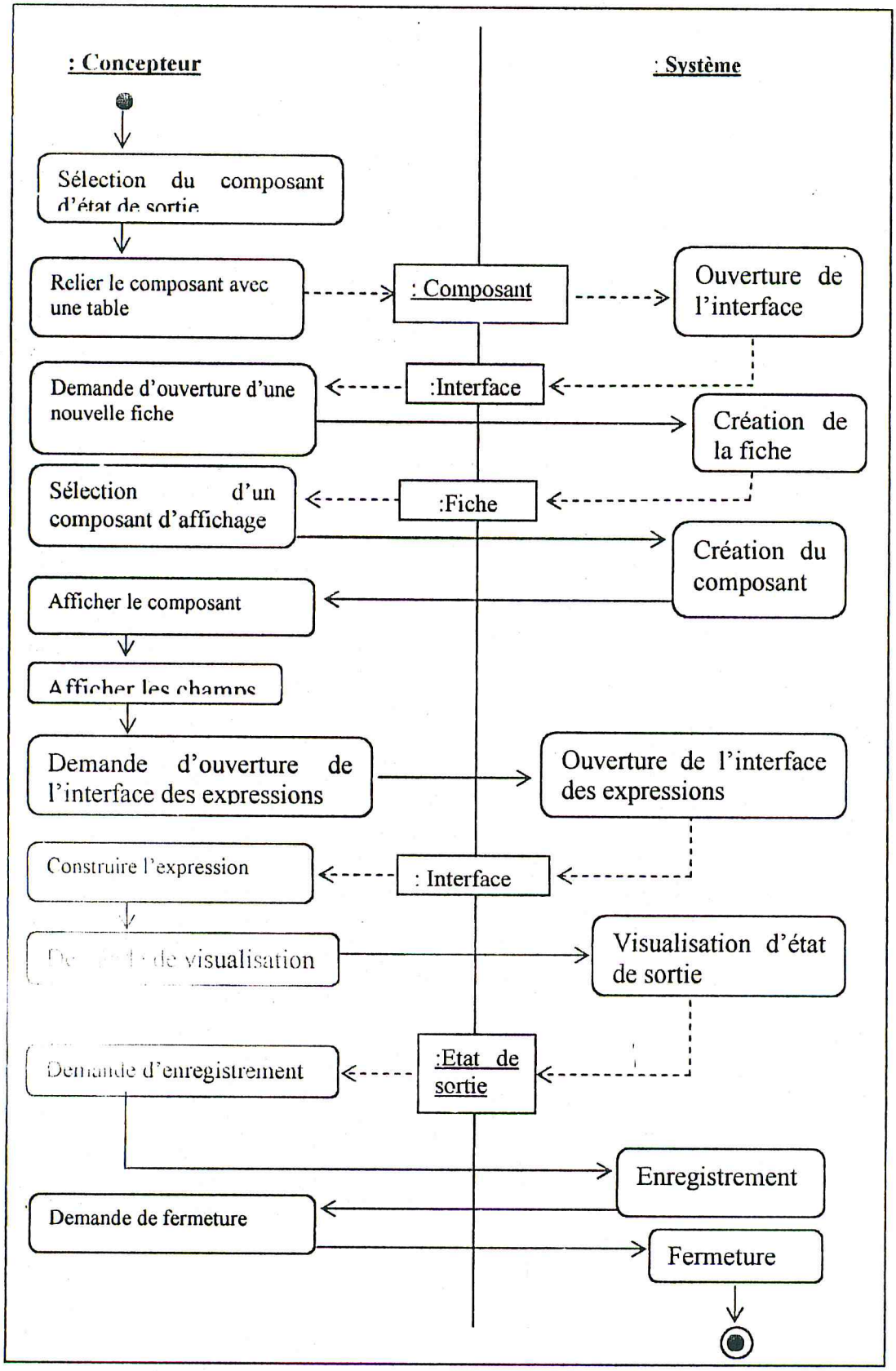


Figure III-32 : Diagramme d'activités pour la création d'état de sortie

➢ Diagramme d'activités pour la création de projet

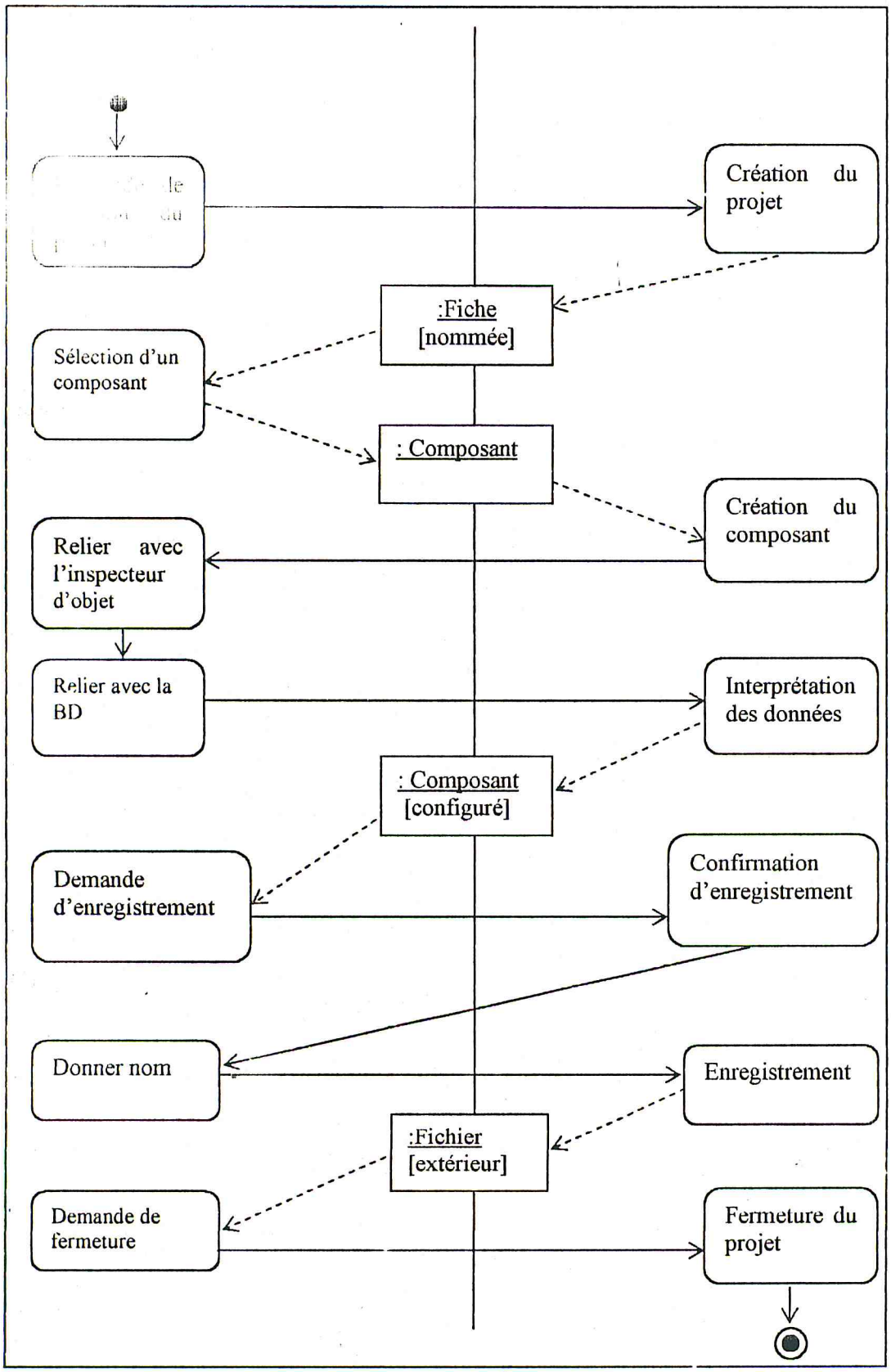


Figure III-33 : Diagramme d'activités pour la création de projet

V -2-.2.Module constructeur de requêtes SQL

Le SQL (Structured Query Language) permet d'interroger une base de données, d'en modifier des informations. C'est un langage universel d'interrogation des bases de données, qui permet à différents systèmes d'échanger des données entre eux.

Le concepteur performant – qui a bien assimilé ce qui se passe en amont et en aval du SI – doit pouvoir utiliser des requêtes SQL qui lui permettront d'aller plus vite et d'être plus efficace. Nous devons lui permettre de construire ces requêtes. La construction d'une requête se fera localement sont avoir recours à un autre module.

❖ **Architecture du module constructeur de requêtes SQL**

L'architecture du module constructeur de requêtes SQL est présentée par le schéma décrit par la « figure III-33 » suivante :

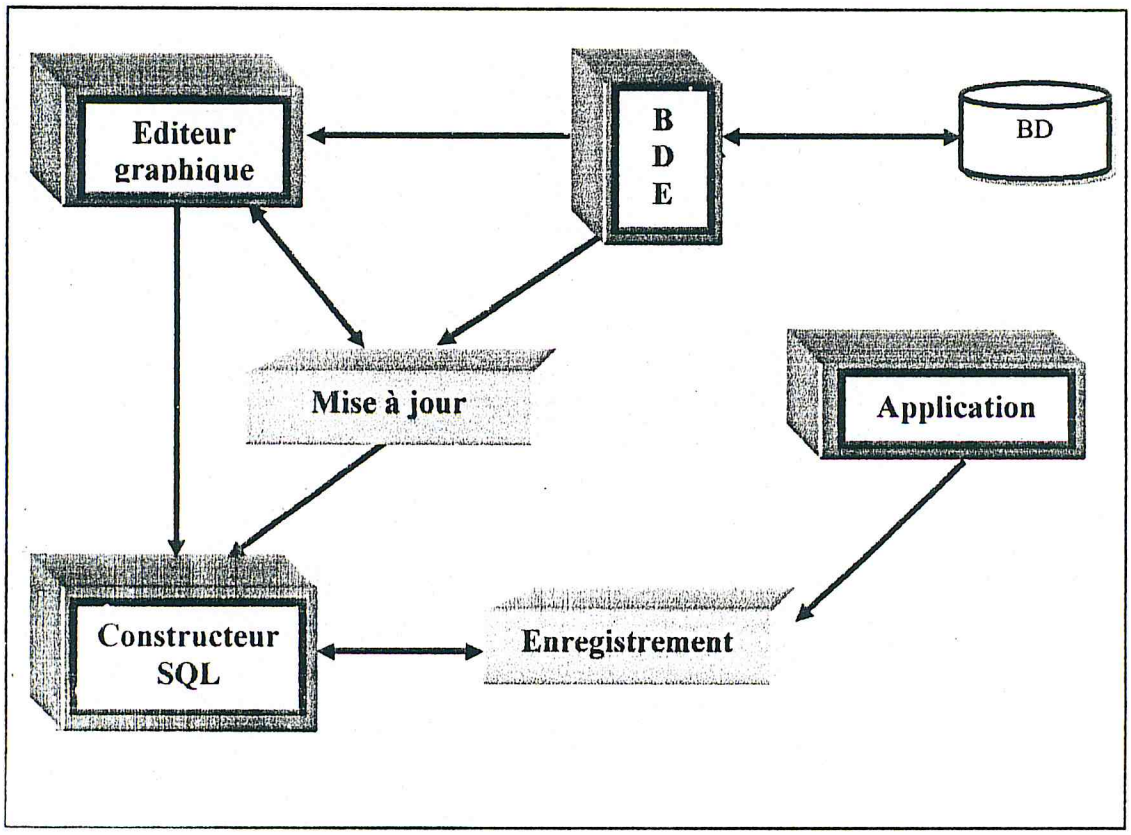


Figure III-34 : Architecture du module constructeur de requête

a) Editeur graphique

Cette classe permet la manipulation des données et des structures de données référencées par l'application L'éditeur graphique (EG) offre à l'utilisateur la possibilité de choisir une BD à partir de celles déjà existantes. Une interface est proposée au concepteur afin de choisir les tables de la requête et une autre pour les relier à l'aide d'un 'trait' qui sera interprété par l'interpréteur SQL.

La deuxième phase de la construction de la requête correspond à la saisie des différents critères de recherche souhaités par le concepteur. La formulation de la requête doit se faire en mode assisté et dans un langage quasi-naturel. Le constructeur aura à construire la requête SQL correspondant aux choix du concepteur.

Le constructeur de requêtes offre au concepteur la possibilité de .

- Créer une nouvelle requête en utilisant un ensemble d'outils facilitant la saisie des différents critères au concepteur : une liste de connecteurs logiques (et, ou,...), une liste d'opérateurs relationnels (<, >, =, ...) et une liste d'attributs liés a la bases de données de l'application choisie.

- Utiliser une requête déjà cataloguée sachant que les requêtes couramment utilisées peuvent être sauvegardées et présentées ultérieurement au concepteur sous forme d'un fichier.

- Faire appel, selon les exigences du concepteur, à une pondération des critères en cas d'utilisation des opérateurs logiques « ou » et « et ». Pour ce faire, la requête est subdivisée selon les connecteurs, c'est-à-dire qu'une pondération ne peut jouer qu'entre les termes d'une alternative. Dans ce cas, chacun des deux critères séparés par un connecteur logique est évalué à part et la pondération devra accompagner les t-uplets résultats de la requête, qui seront affichés. L'opération se fera, évidement, d'une façon récursive.

- Effectuer les différentes opérations de mises à jour d'une requête en vue de modifier, supprimer ou ajouter un ou plusieurs critères.

- Sauvegarder la requête formulée dans la base de requêtes et ce en donnant la main à l'utilisateur d'enregistrer une requête soit sous un nouveau nom soit sous un ancien pour écraser l'ancienne valeur de la requête que pourrait pointer ce nom.
- Exécuter une requête par le noyau SQL puis afficher l'ensemble des résultats répondant aux critères de sélection.
- Sauvegarder le résultat d'une requête.

b) Constructeur SQL

Permet au concepteur d'assister durant la saisie de sa requête à l'aide d'une vérification en ligne de la syntaxe de la requête. Une fois l'édition de la requête achevée, le concepteur voit dans une fenêtre la requête affichée dans le langage SQL

c) Le moteur de base de données de Borland (BDE)

Pour permettre l'accès à des BDs locales et à des BDs client-serveur, nous utilisons le BDE (Borland Data base Engine, moteur de BD Borland). Le BDE est une collection de DLL et d'utilitaires qui permettent d'accéder à diverses BD. Ainsi, notre application peut se connecter à une BD sans avoir connaissance du fonctionnement spécifique de la BD en question.

A) Paquetage du module constructeur de requêtes SQL

Les éléments du paquetage du module constructeur de requêtes SQL sont représentés par la figure suivante :

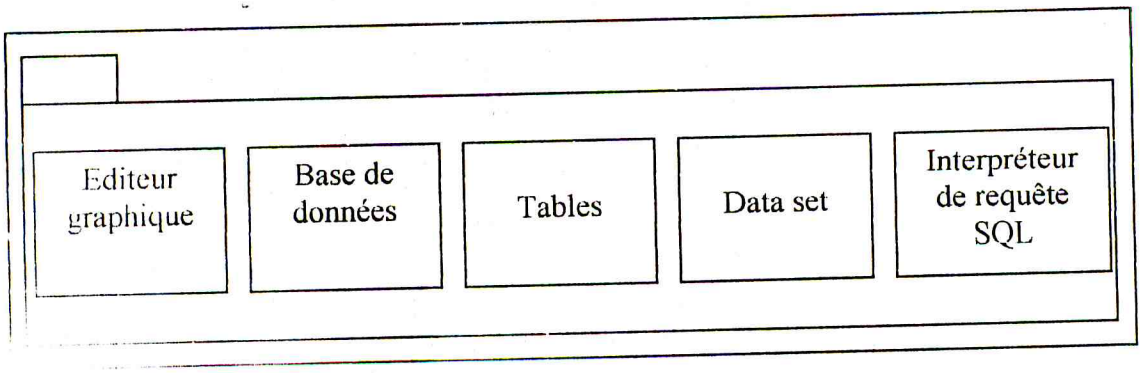


Figure III-35 : Paquetage du constructeur de requêtes SQL.

❖ Diagrammes de collaboration :

Les diagrammes de collaboration du module générateur d'interface sont représentés par les figures 36, 37, 38.

➢ Choix des tables de la requête :

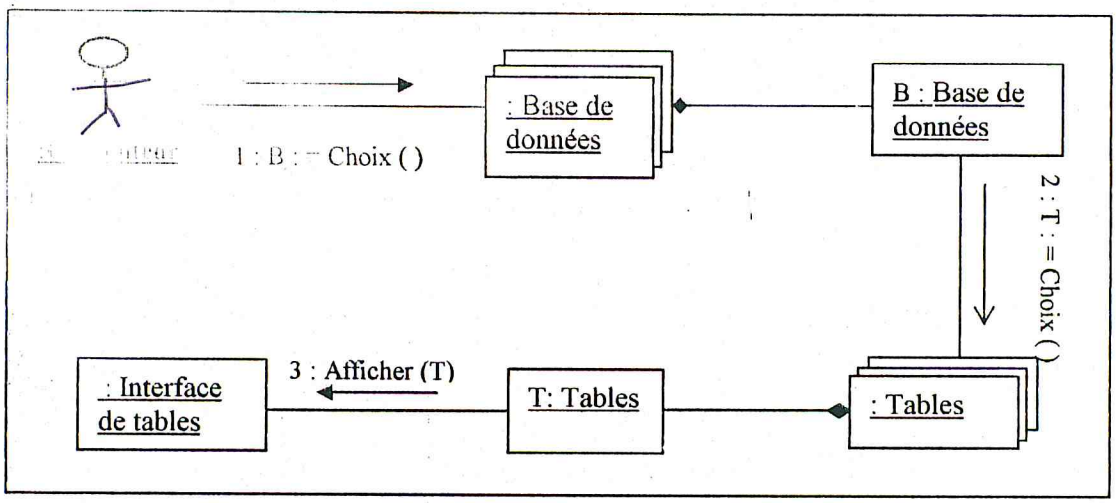


Figure III-36 : Diagramme de collaboration « Choix des tables de la requête »

➢ Choix des champs de la requête :

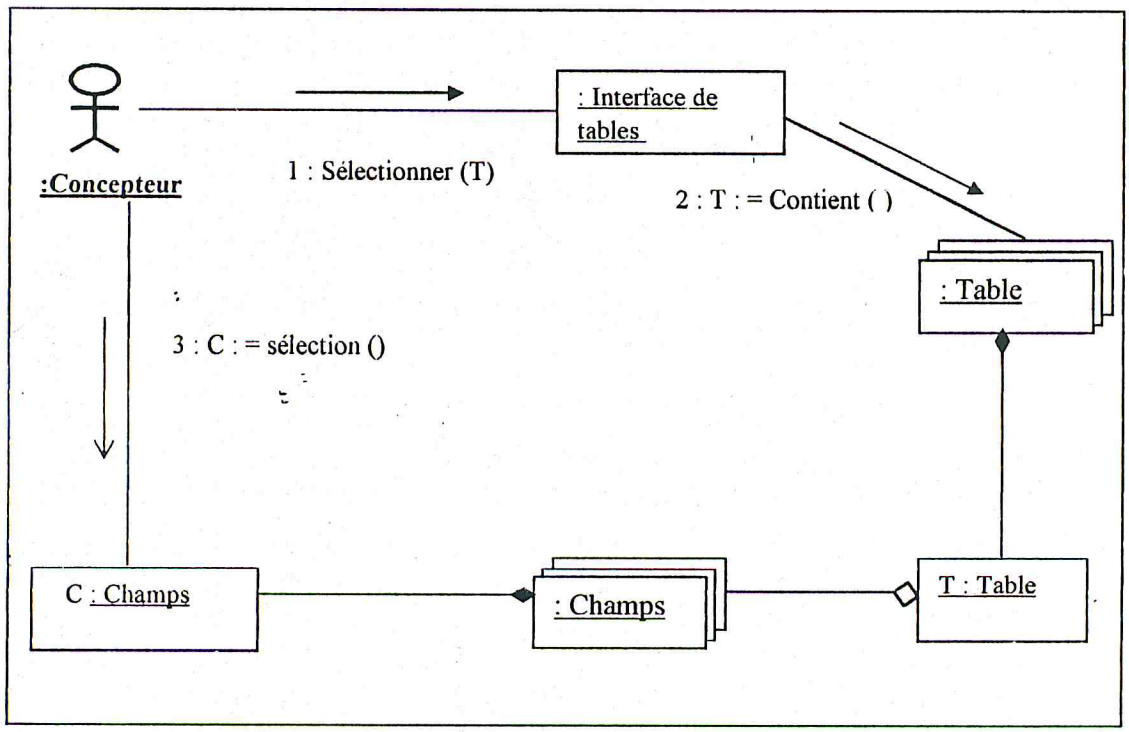


Figure III-37 : Diagramme de collaboration « Choix des champs de la requête »

➢ **Utilisation des critères :**

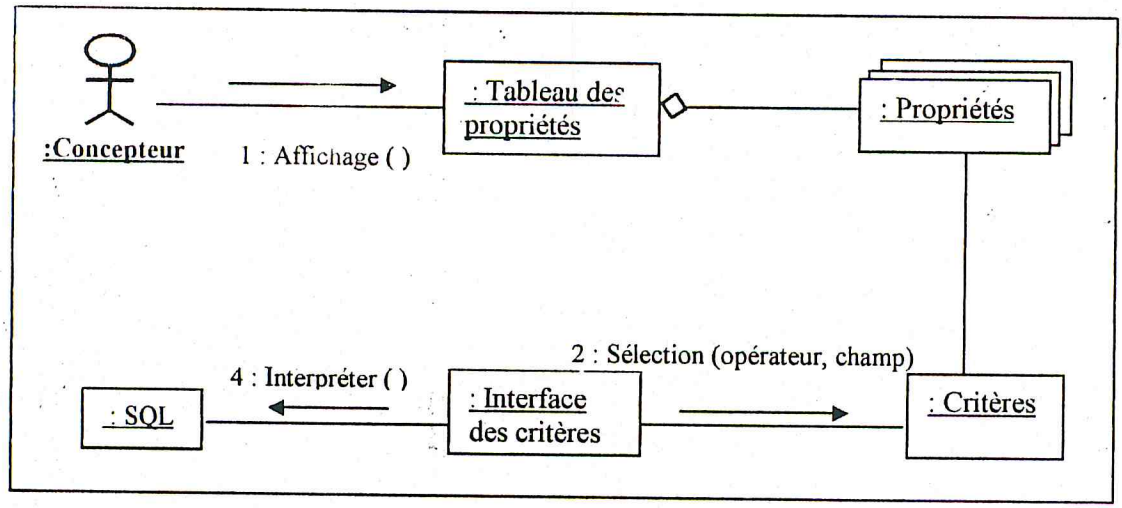


Figure III-38 : Diagramme de collaboration « Utilisation des critères »

D) Diagramme de classes :

Le diagramme de classe représenté par la figure III-38 représente les éléments de modélisation qui doivent être implémentés, d'une manière synthétique :

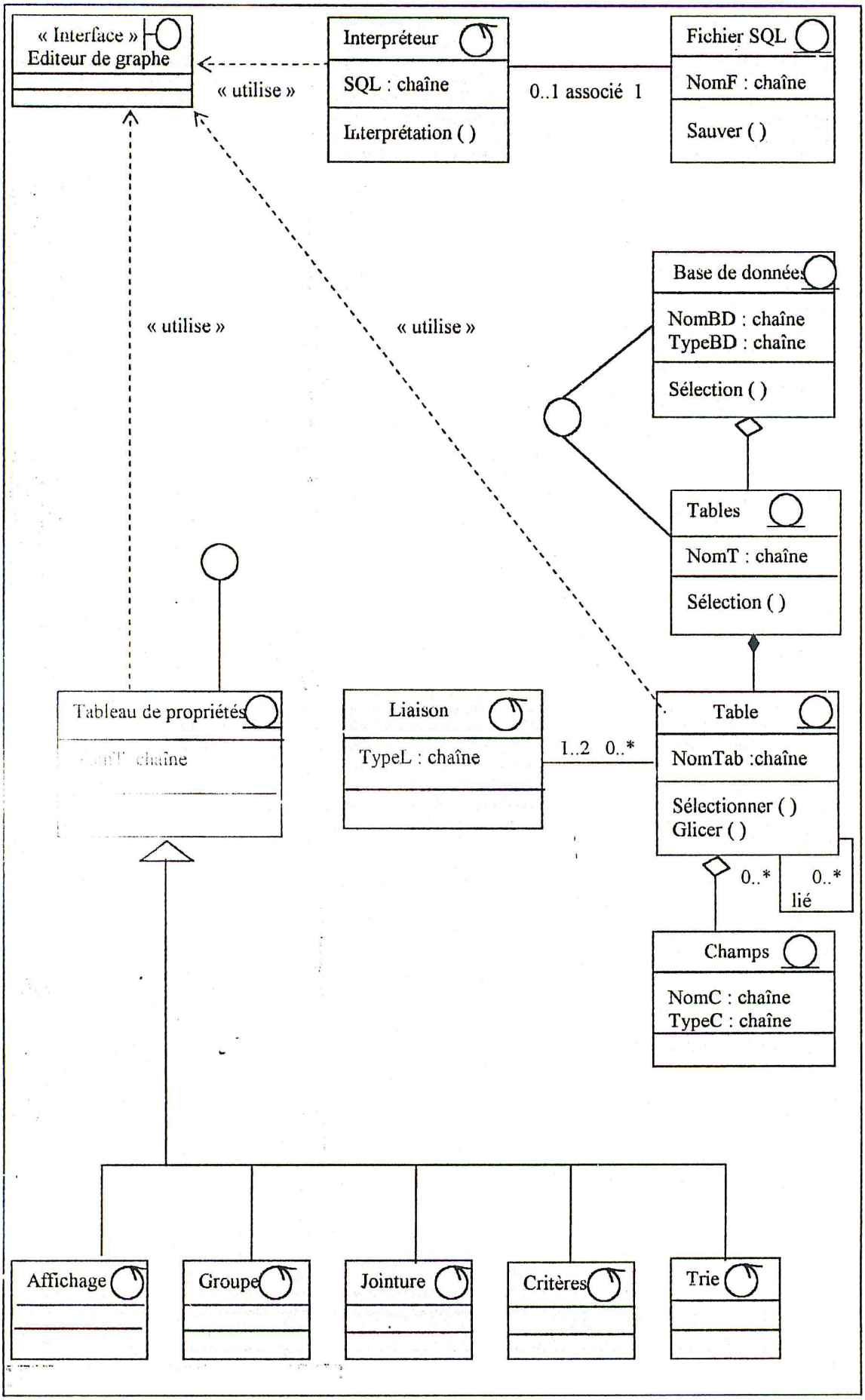


Figure III-39 : Diagramme de classes du constructeur de requêtes SQL

❖ Diagrammes d'états-transitions et d'activités :

➢ Création d'une requête SQL :

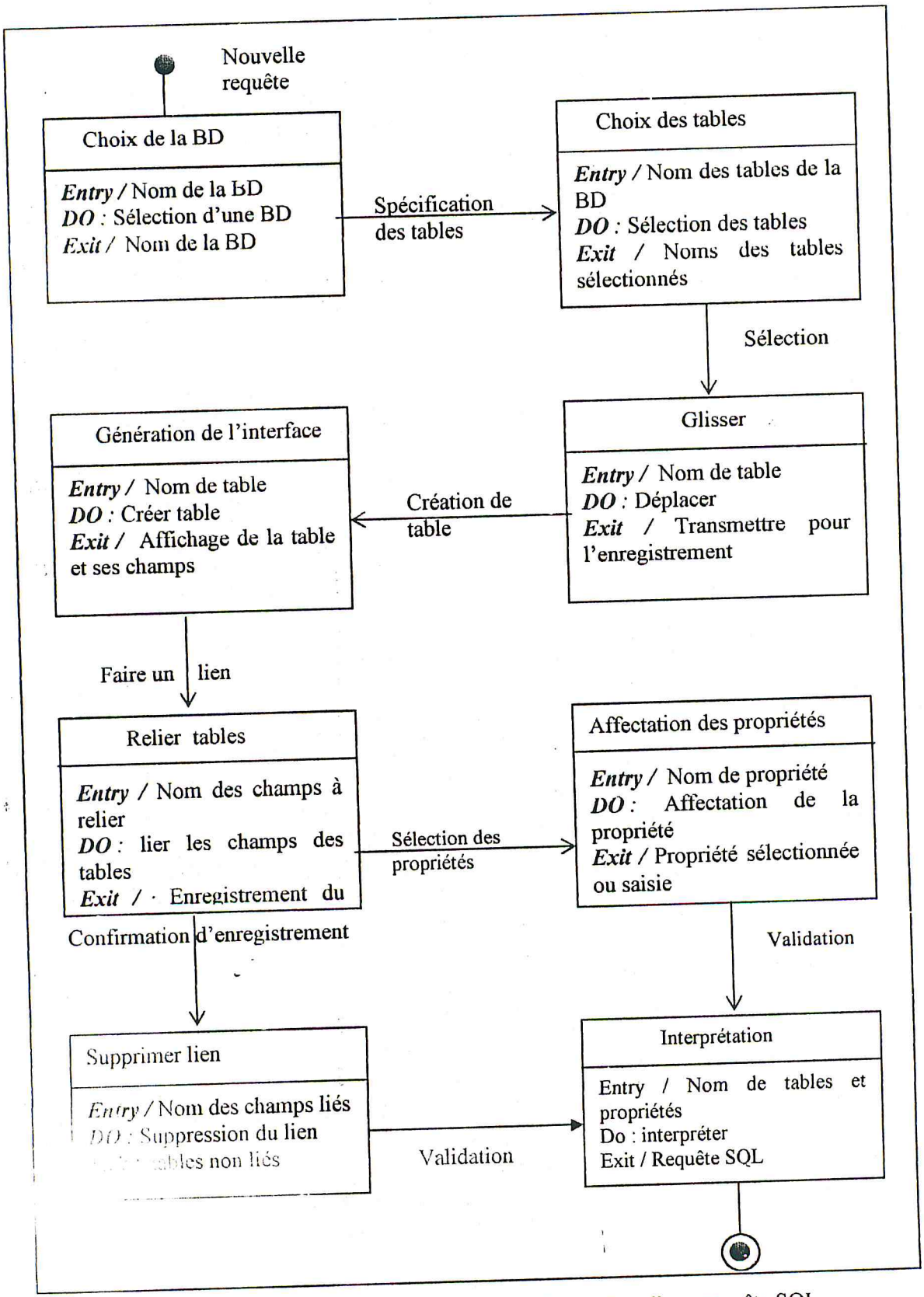


Figure III-40 : Diagramme d'états pour la création d'une requête SQL

➢ Création d'un fichier de requête SQL

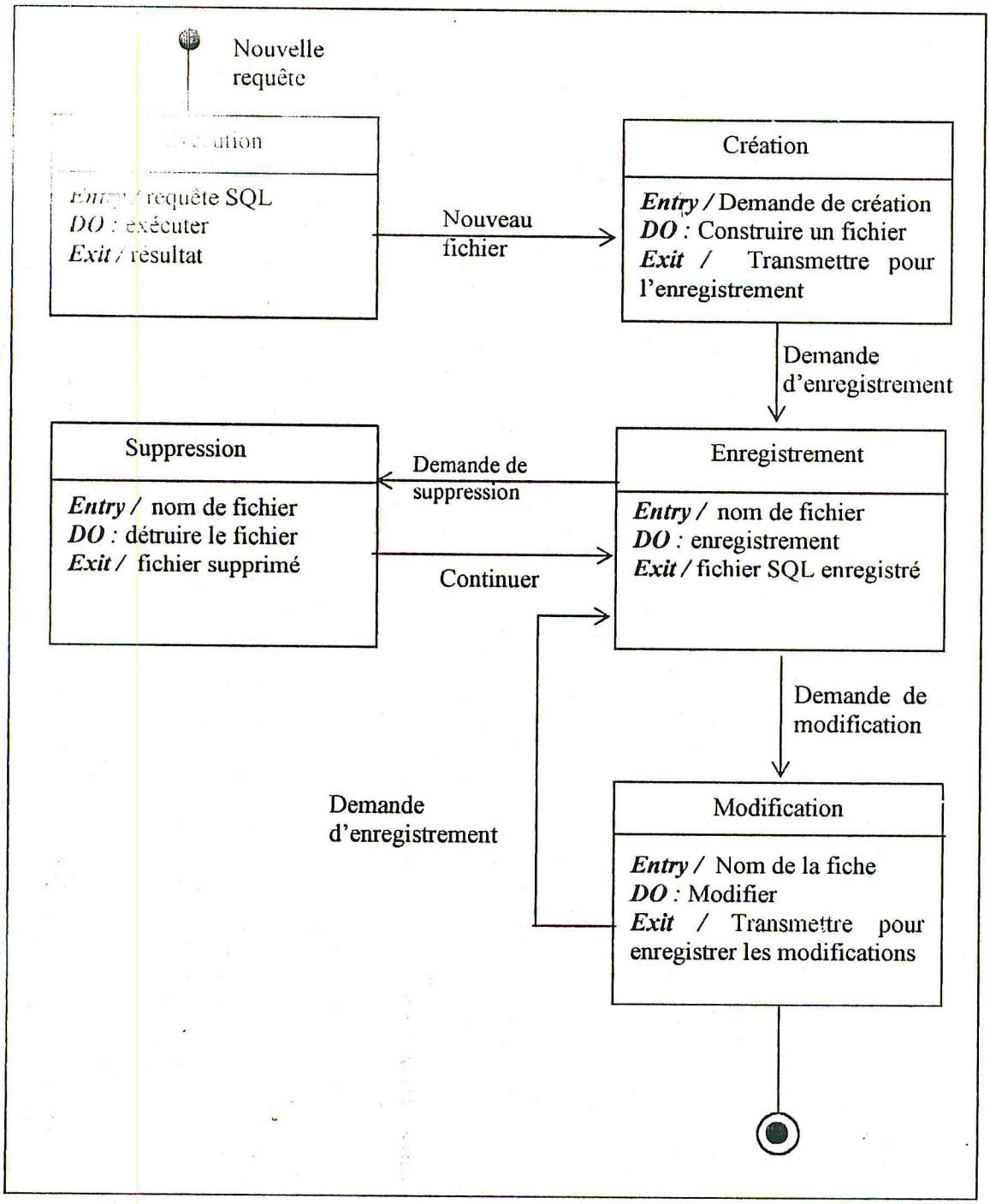


Figure III-41 : Diagramme d'états pour la création d'un fichier de requête SQL

➢ **Diagramme d'activités pour la création du fichier SQL:**

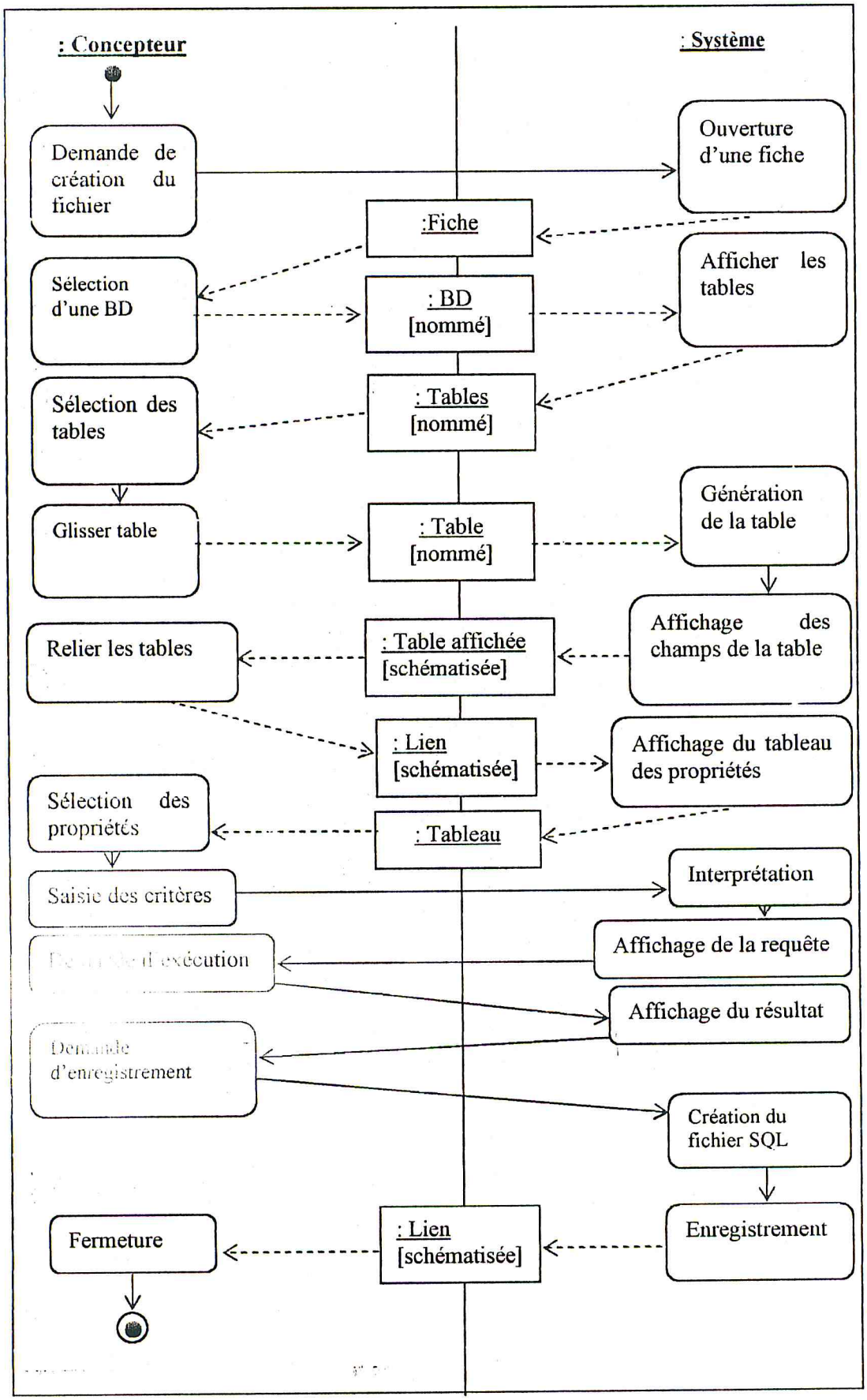


Figure III-42 : Diagramme d'activités pour la création du fichier SQL

V -2-.3. Module système expert

Les systèmes expert cherchent à produire l'expertise d'humaine au niveau de la machine. Par expertise, il faut comprendre à la fois la connaissance passive dans un domaine spécifique et d'autre part la capacité de déduire ou d'induire une nouvelle connaissance à partir de ces faits de base. C'est à dire être à même de raisonner. La technique des SE s'est très rapidement étendue et a gagné les entreprises traditionnellement soucieuses de la rentabilité [Mes, 88].

Un SE permet une standardisation. En effet, un expert humain ne demeure pas insensible aux conditions de l'environnement (fatigue, nervosité, humeur,...). Celles-ci affecteront par conséquent son travail, de sorte que l'expert paraîtra quelque fois incohérent. Le SE, au contraire, lorsqu'il est mis en présence des mêmes conditions statuera toujours d'une façon identique. La qualité du service se verra par conséquent améliorée par des décisions rapides toujours cohérentes ainsi que par la réduction des erreurs engendrées par exemple par la fatigue d'un expert humain.

❖ Architecture du module système expert

Les SEs représentent un nouveau concept : il s'agit d'une nouvelle façon de penser informatique où le concept primordial est la séparation des connaissances (les données concernant le sujet) et du traitement proprement dit. En effet, en informatique classique, les données et les traitements sont intimement mêlés [Mes, 88].

L'architecture du module système expert est présentée par le schéma décrit par la « figure III-43 » suivante :

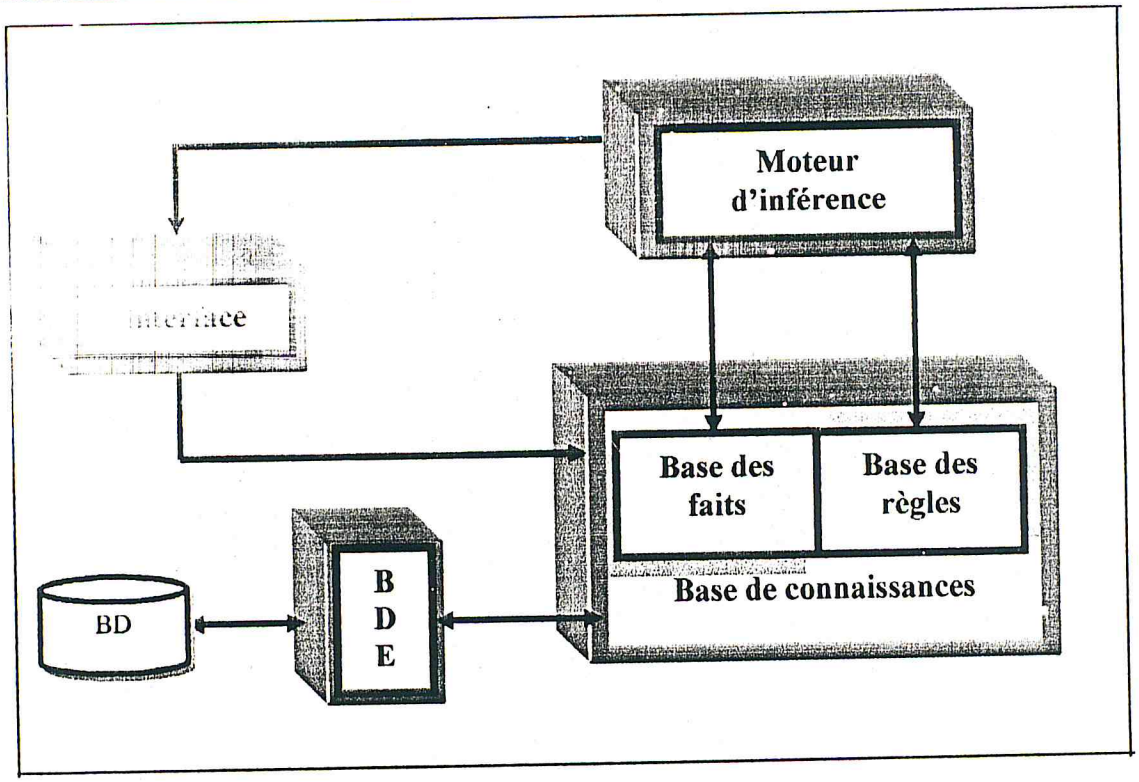


Figure III-43 : Architecture du module système expert

➢ **L'interface**

Le traitement de l'information requiert différents types d'interfaces : interface utilisateur, base de données, programmes externes... Un bon SE exploitera judicieusement ces différentes possibilités afin de tirer parti de la puissance de chacun de ces outils. La conception d'interfaces est maintenant reconnue avoir une influence significative sur le temps d'apprentissage, la vitesse d'exécution, le taux d'erreur, la satisfaction de l'utilisateur etc.

L'interface doit se révéler le plus convivial possible et aisée à l'emploi. La convivialité sera assurée par des menus déroulants et un environnement de fenêtrage avec boite de dialogue. L'accès à ces fenêtre est possible en utilisant un inspecteur d'objet, les principales propriétés de l'inspecteur d'objets qui ont un rapport avec le SE sont :

Propriétés	Composants	Rôle
Expression	TField	calculer une expression
Expression de validation	TField	Se produit juste avant l' enregistrement des données.
Après suppression	Bouton, DBGrid, DBNavigateur	Se produit après la suppression d'un enregistrement
Après enregistrement	Bouton, DBGrid, DBNavigateur	Se produit après l'insertion d'un nouvel enregistrement
Contraintes	TField	Spécifie les contraintes de niveau enregistrement qui doivent êtres respectées quand les données sont modifiées.
Message d'erreur	TField	Contient un message d'erreur personnalisé qui est affiché quand l'utilisateur tente de saisir une valeur ne respectant pas les contraintes de données du champ.
Format d'affichage	TField	Détermine comment la valeur d'un champ numérique est formatée pour l'affichage dans un contrôle orienté données.
Evènement	Bouton	Utilisé pour parcourir la BD, ouvrir une fiche, imprimer un état, ouvrir un dialogue et la recherche.

Tableau III-3 : Les principales propriétés qui ont un rapport avec le SE.

Remarque : Les propriétés avant suppression et avant enregistrement ne figurent pas dans l'inspecteur d'objet mais s'exécute automatiquement par le SE lors de la suppression ou de l'enregistrement des données.

* La base de connaissances

La base de connaissances contient toute l'information dont le concepteur du logiciel aurait besoin pour s'acquitter de son travail.

Représenter les connaissances dans un ordinateur consiste à trouver une correspondance entre le monde extérieur et un système symbolique permettant de raisonner. Le modèle de représentation de la connaissance utilisé est la représentation procédurale qui comprend les procédures. Les procédures sont des programmes classiques-désignés explicitement pour pouvoir les invoquer sans possibilité d'alternative [Mes, 88].

*** La base des faits**

La base des faits est la mémoire de travail du SE. Les faits, que les règles régissent, doivent souvent être mis à jour si nous voulons l'expertise, fournie par le SE, reste d'actualité. A partir du moment où nous savons que la tâche de tenir à jour une foule d'informations est effectuée avec efficacité par le SGBD, nous pouvons aisément libérer le SE de cette besogne en établissant un lien de communication entre lui et la base de données qui contient tous les fait, constamment mis à jour, dont il a besoin pour sa propre fin. Les fait de notre système sont les contraintes correspondant aux champs des tables de la base de données de travail.

*** La base des règles**

La base des règles contient les règles qui indique quelles conséquences tirer lorsque telle situation est établie. Les règles déduites sont un cas spécial des règles, nous disons que deux règles sont déduites si la première règle est déduite de la deuxième règle.

Les règles de conception sont représentées sous forme de règles procédurales et de règles de productions, cette représentation procédurale permet la spécification d'interactions directes entre les faits. Les règles de production ont la forme générale suivante : Si (liste de conditions) alors (liste d'action), les conditions sont en générale les contraintes associés aux champs des tables de la BD. Les actions peuvent être de plusieurs types, des actions de mis à jour de données, des actions d'appel de procédures et des actions ou d'appel à des méthodes associées à certains composant.

➢ **Moteur d'inférence**

Le moteur d'inférence permet de manipuler les données stockées dans la base de connaissances de façon à pouvoir résoudre les problèmes posés au système.

Le moteur d'inférence doit être alimenté avec des données pour pouvoir fonctionner. Le faits sont chargés dans le système en connectant le système à une BD et en positionnant le pointeur de la BD sur un enregistrement d'une table spécifiée. Chaque fait doit être libellé entre parenthèse.

Exemple : (Nom not nul) , (Age > 20).

L'ordre de chargement des faits est sans importance : le moteur d'inférence les examinera un à un, systématiquement. En revanche, la base des faits est ré-initialisée (c'est-à-dire complètement effacée) à chaque nouvelle exécution.

Pour que le moteur d'inférence puisse produire des nouveaux faits, il parcourt la base des fait, examine le fait condition, cherche dans la partie gauche de la règle s'il y a correspondance, si non passe à la règle suivante.

Le SE communique directement avec la BD, en prenant en charge le réflexe qu'il vient de lancer. Ce réflexe est considéré comme une requête. Il va apporter avec lui un ensemble de contraintes existants dans la BD. Ces contraintes vont constituer des contraintes de recherche. Le SE va considérer toutes ces contraintes pour déduire si l'exécution de l'évènement choisi par l'utilisateur est possible ou non.

❖ **Paquetage du module système expert**

Les éléments du paquetage du module système expert sont représentés par la figure suivante :

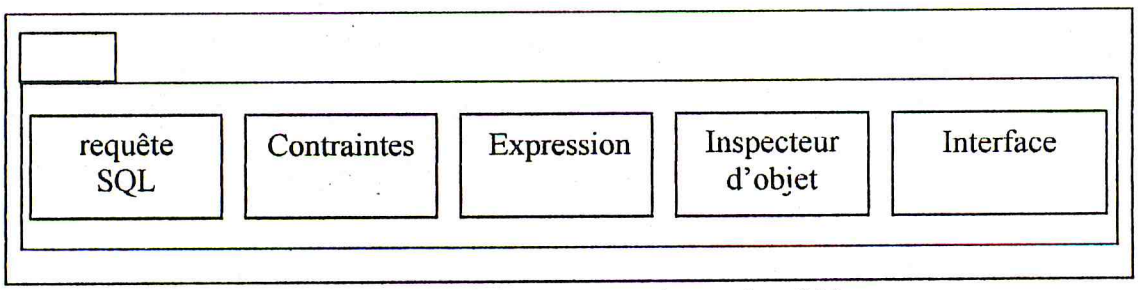


Figure III-44 : Paquetage du constructeur de requêtes SQL.

❖ Diagrammes de collaboration

Les diagrammes de collaboration du module générateur d'interface sont représentés par les figures suivantes :

➢ comportement de la classe avant suppression de données

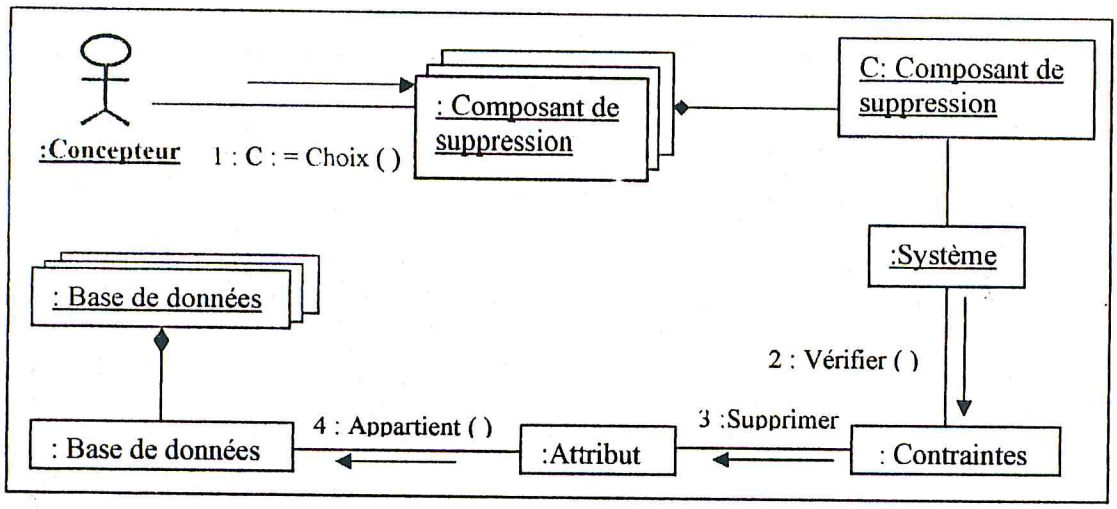


Figure III-45: Diagramme de collaboration du comportement de la classe avant suppression.

➢ comportement de la classe avant l'ajout de données

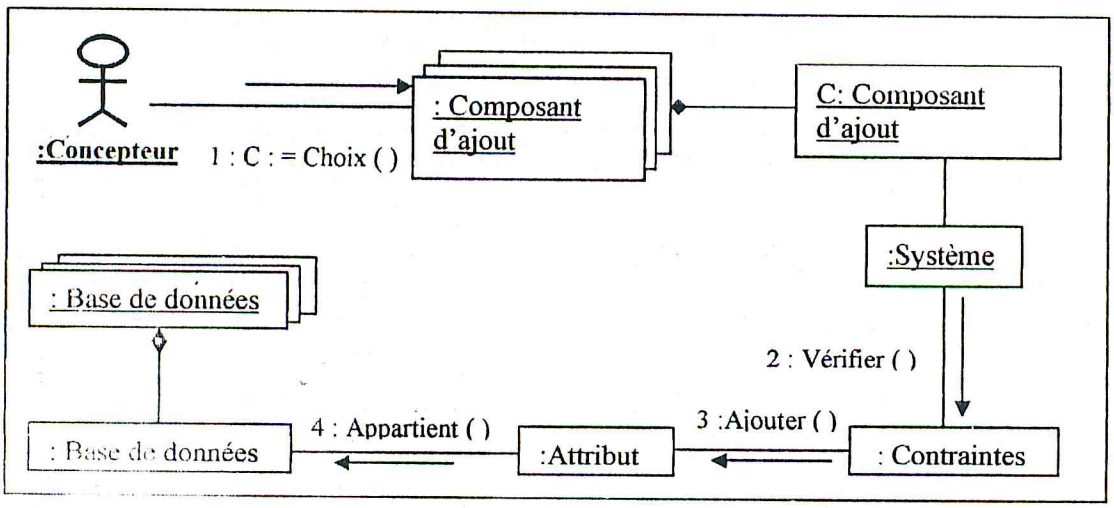


Figure III-46: Diagramme de collaboration du comportement de la classe avant l'ajout de données.

➢ **comportement de la classe avant modification de données**

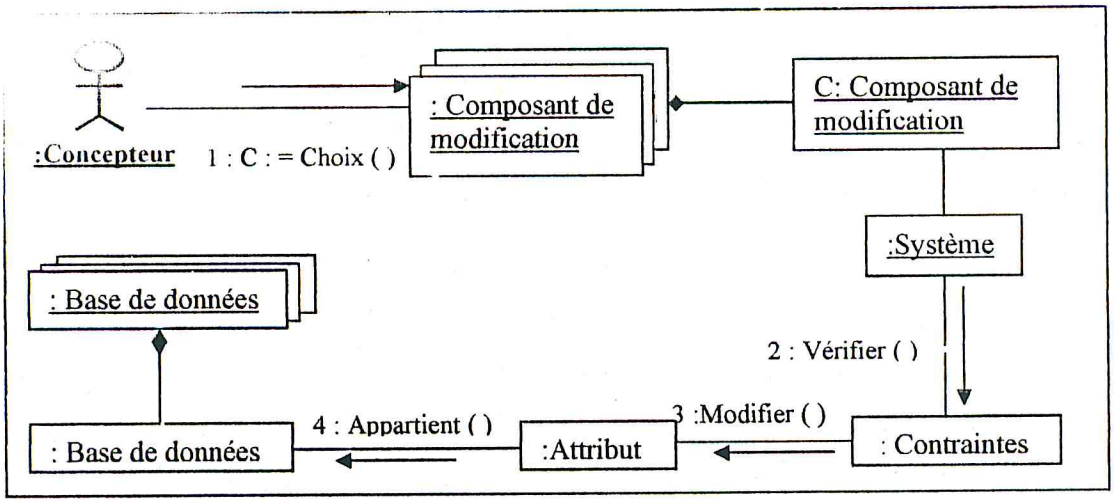


Figure III-47: Diagramme de collaboration du comportement de la classe avant modification de données.

➢ **comportement de la classe après suppression de données**

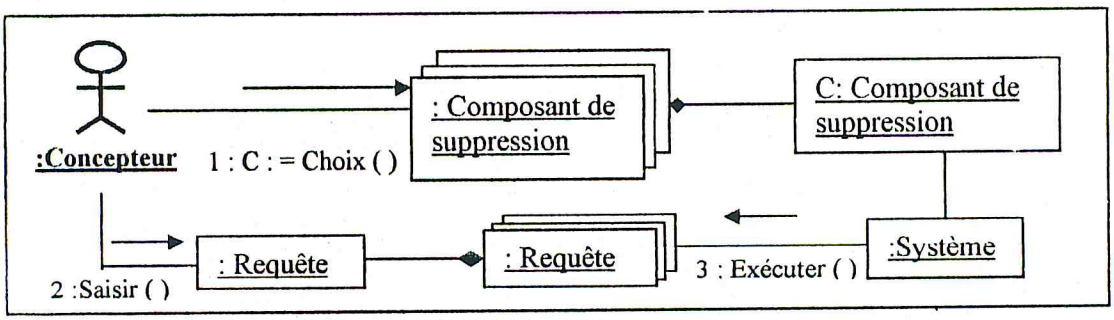


Figure III-48: Diagramme de collaboration du comportement de la classe après suppression de données.

➢ **comportement de la classe après l'ajout de données :**

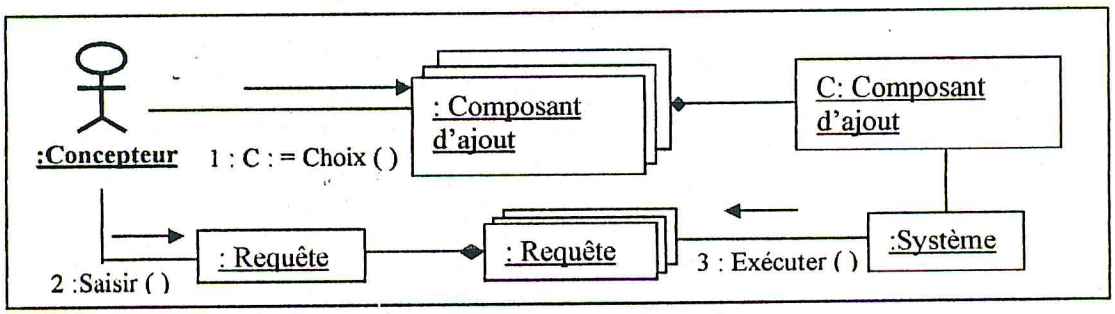


Figure III-49: Diagramme de collaboration du comportement de la classe après l'ajout de données.

➢ **comportement de la classe après modification de données**

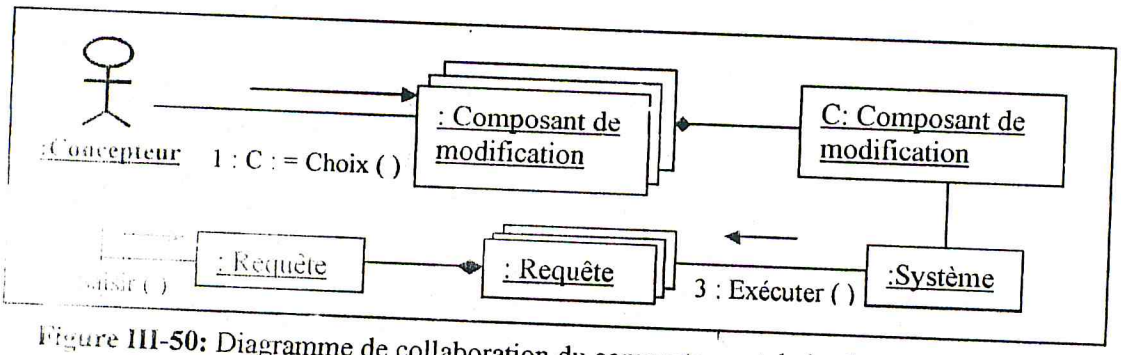


Figure III-50: Diagramme de collaboration du comportement de la classe après modification de données.

➢ **comportement de la classe format d'affichage**

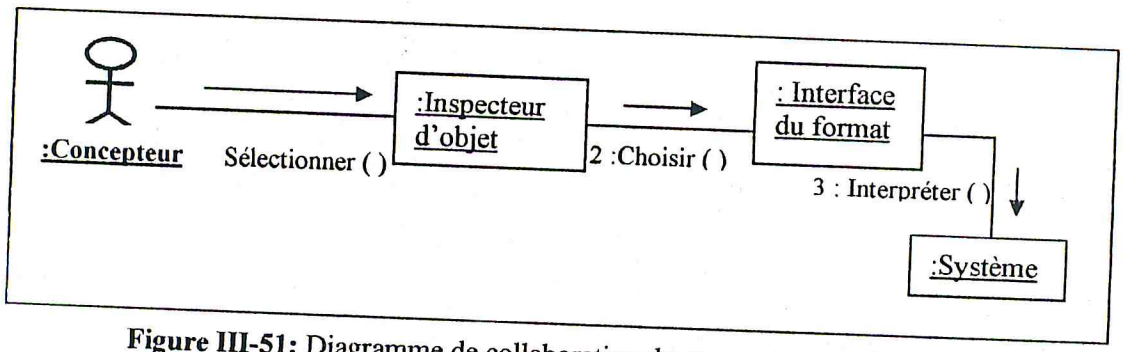


Figure III-51: Diagramme de collaboration du comportement de la classe après modification de données.

➢ **comportement de la classe expression**

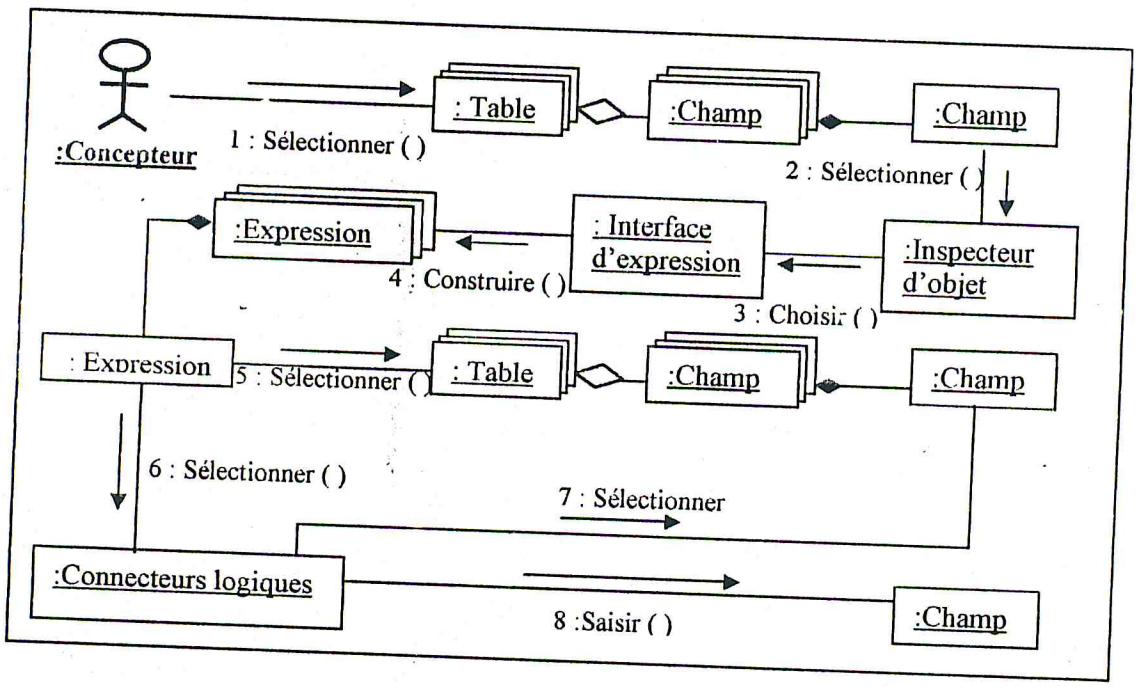


Figure III-52: Diagramme de collaboration du comportement de la classe expression.

➢ **comportement de la classe expression de validation**

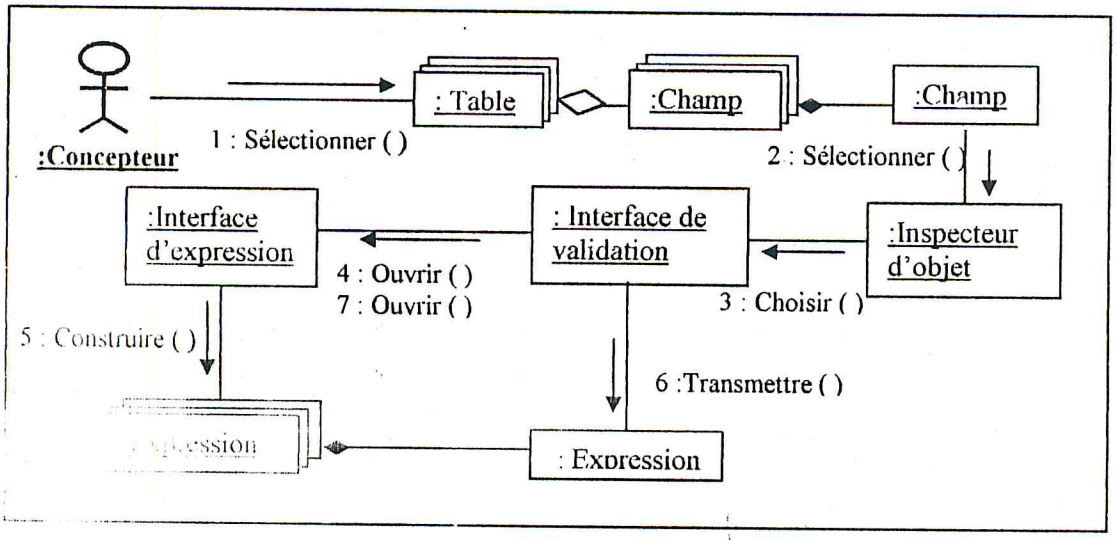


Figure III-53: Diagramme de collaboration du comportement de la classe expression de validation.

➢ **comportement de la classe contraintes**

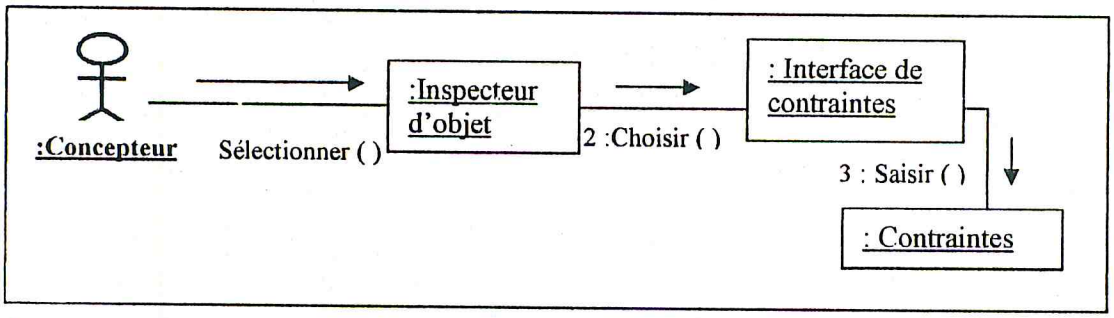


Figure III-54: Diagramme de collaboration du comportement de la classe contraintes

❖ Diagramme de classes du module système expert

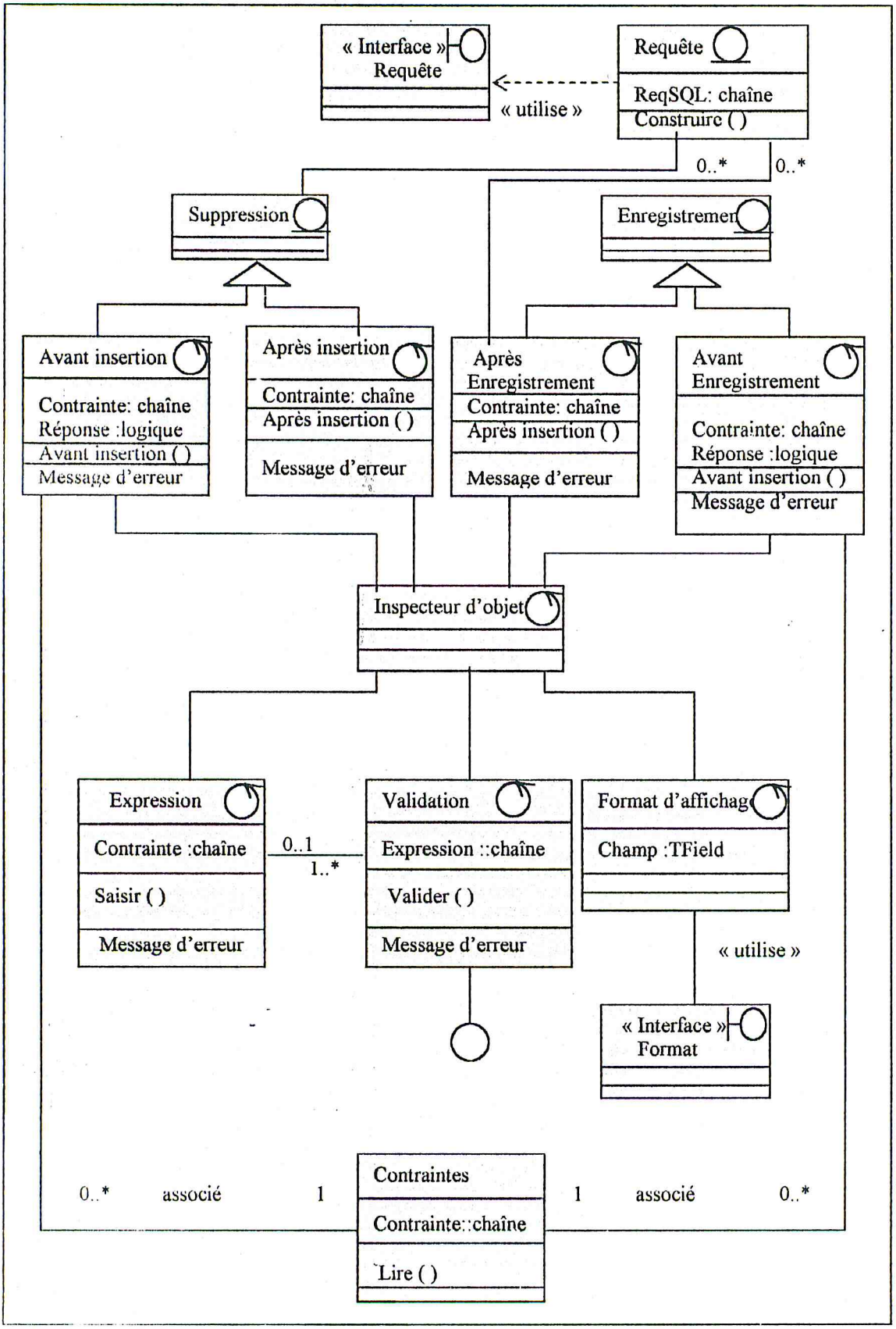


Figure III-55 : Diagramme de classes du système expert

A) Diagrammes d'activités :

➢Diagramme d'activités pour la suppression des données:

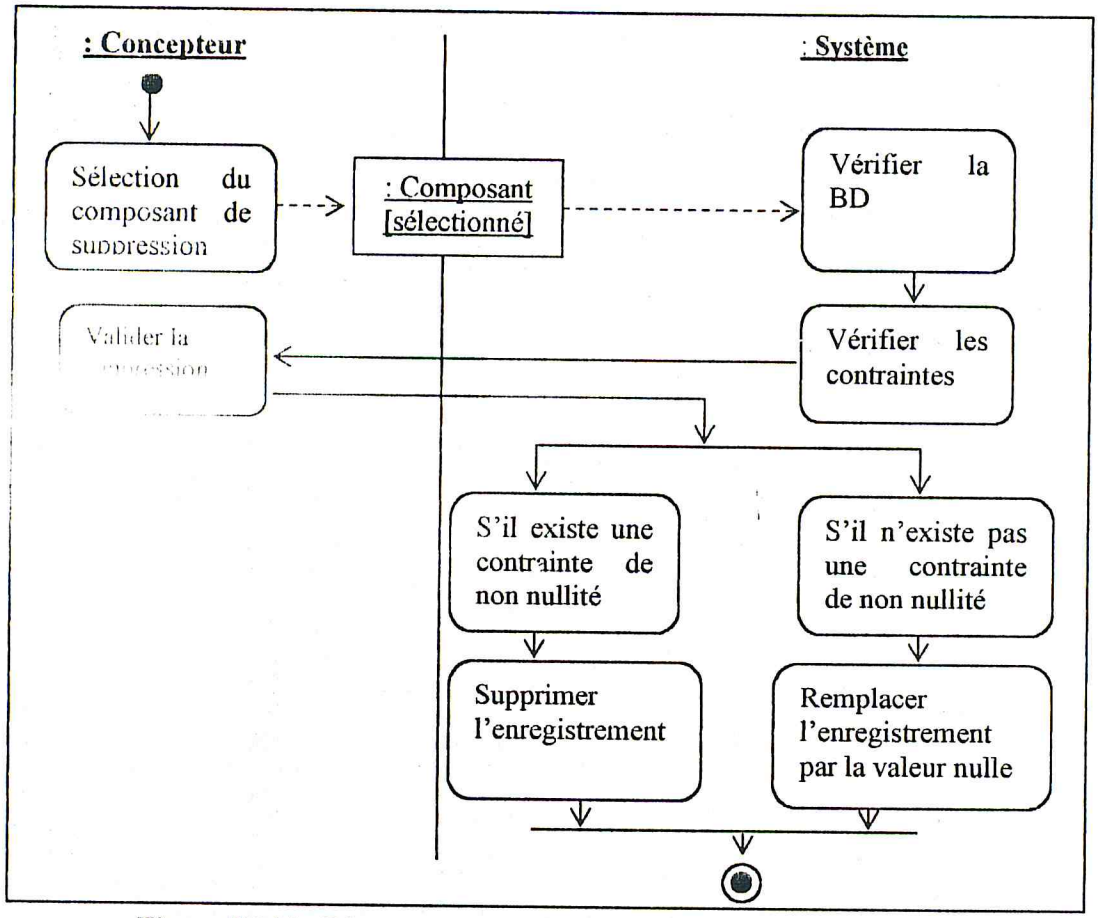


Figure III-56 : Diagramme d'activités pour la suppression des données

➢Diagramme d'activités pour la suppression des données:

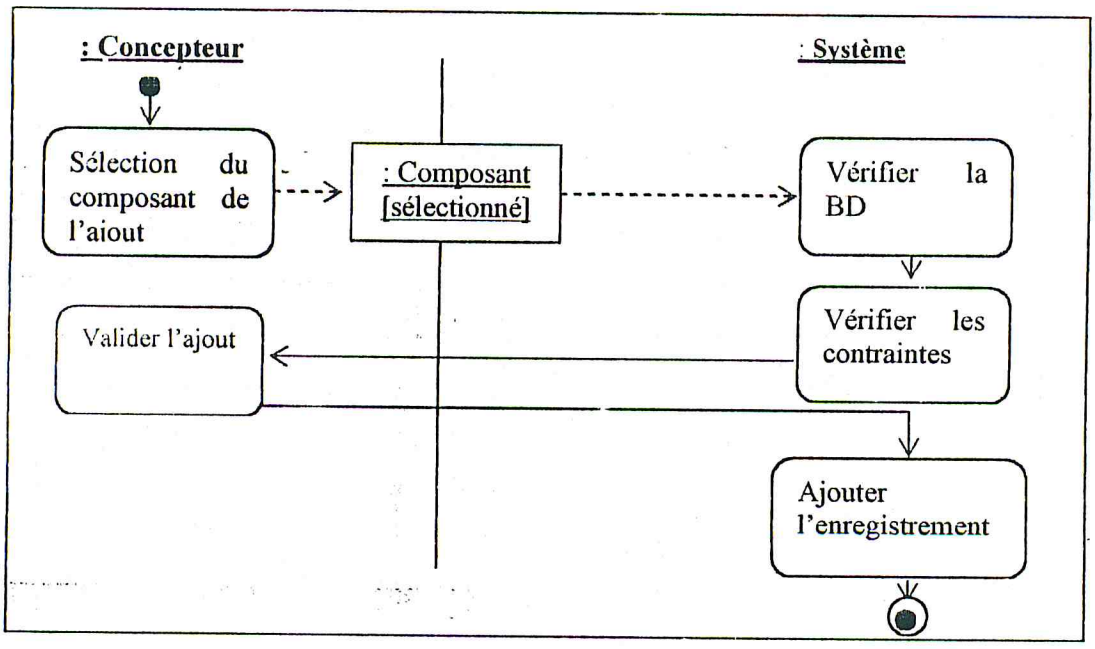


Figure III-57 : Diagramme d'activités pour l'ajout des données

➢ Diagramme d'activités pour la modification des données

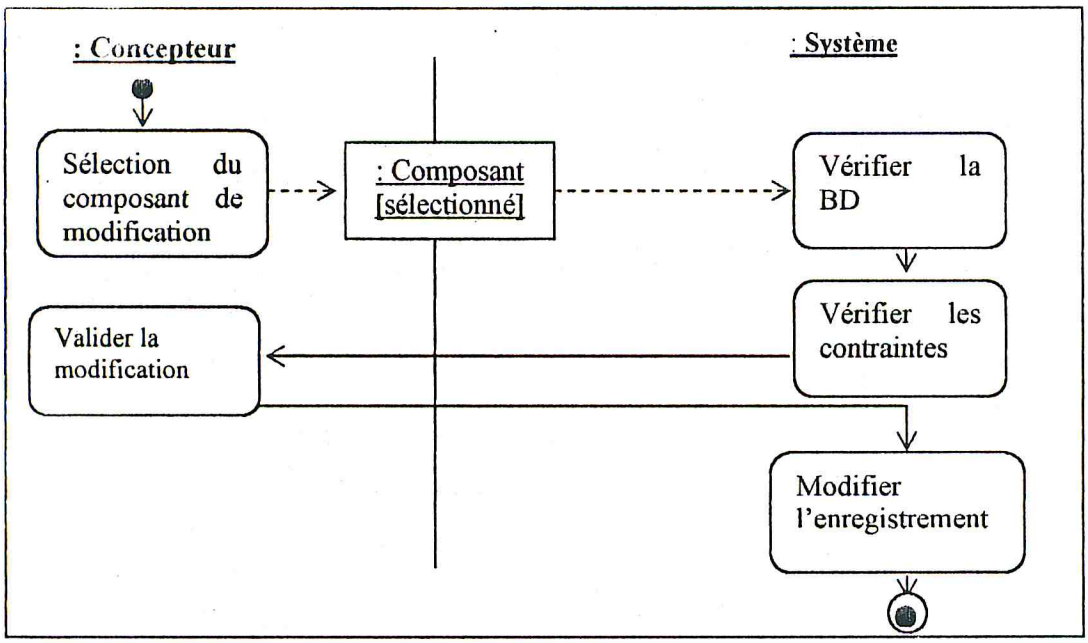


Figure III-58 : Diagramme d'activités pour la modification des données

➢ **Après suppression et après enregistrement des données**

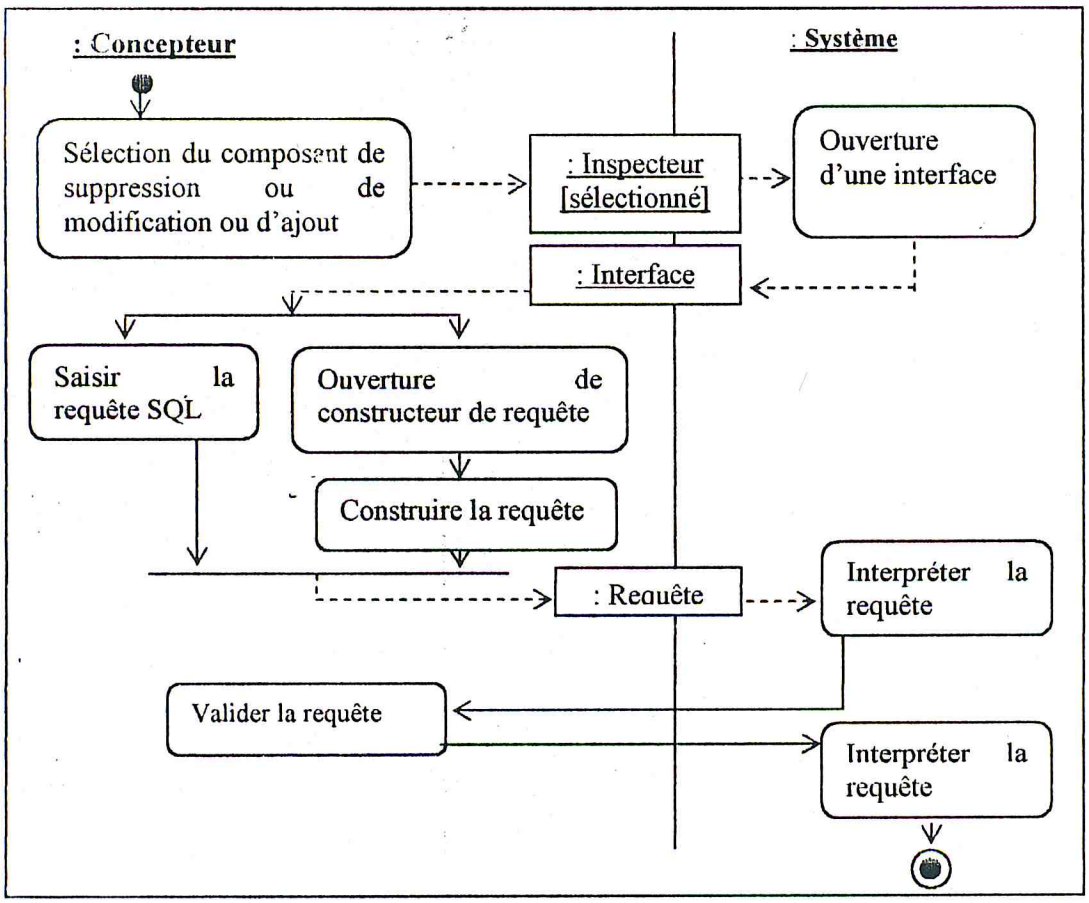


Figure III-59 : Diagramme d'activités pour l'après suppression et l'après modification

➢ **Expression de validation**

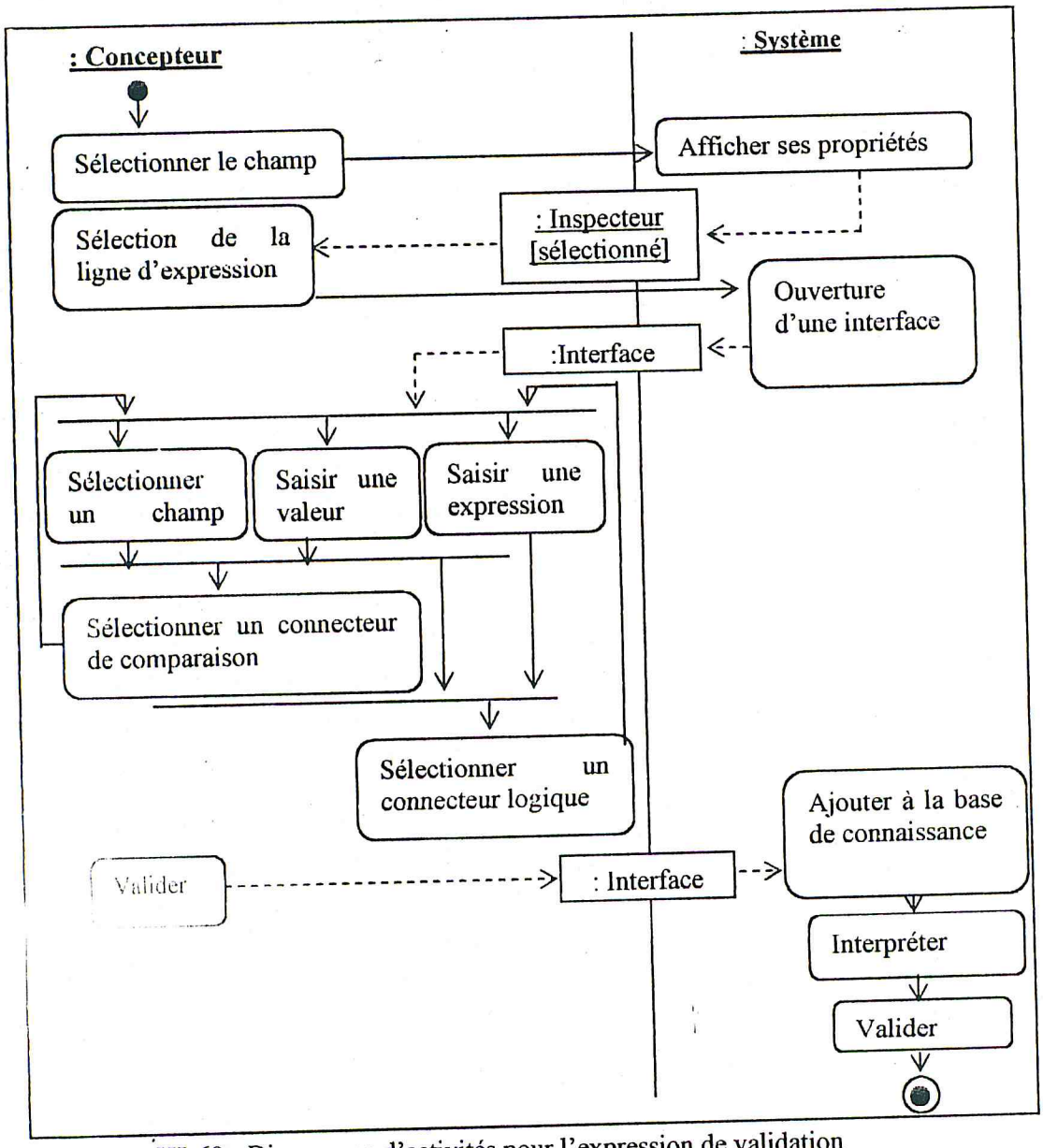


Figure III-60 : Diagramme d'activités pour l'expression de validation

➢Format d'affichage :

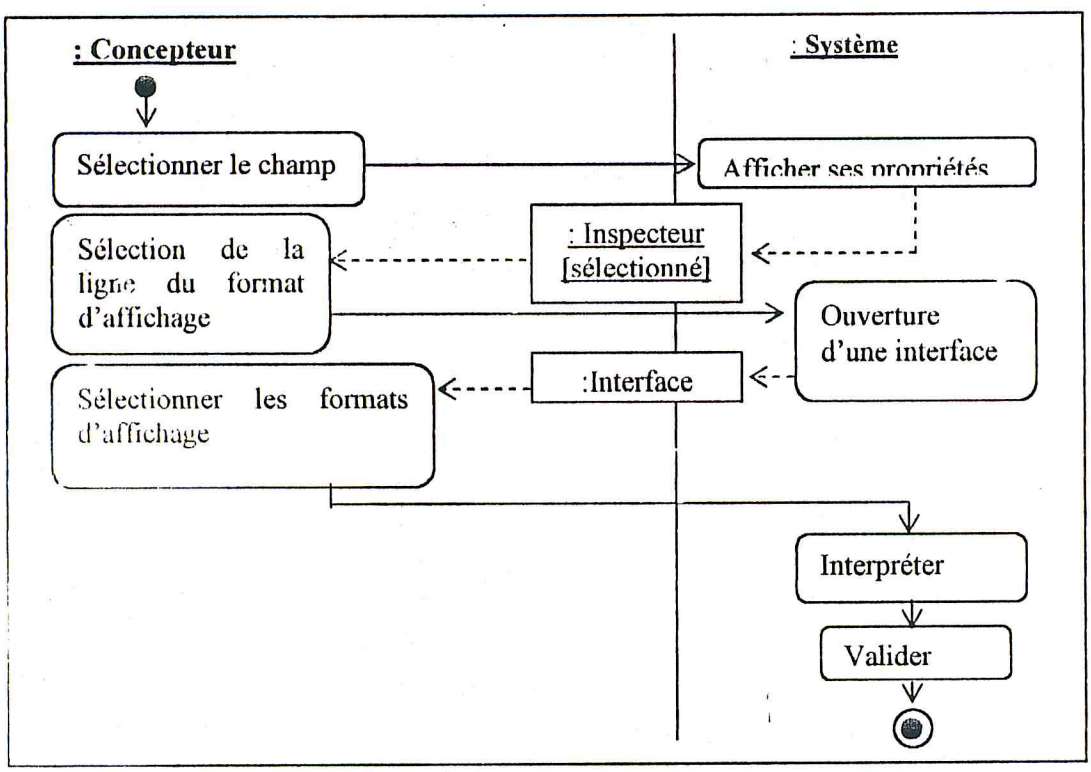


Figure III-61: Diagramme d'activités pour le format d'affichage

V -2-.4. Module interface utilisateur

Ce module est au service de l'utilisateur car nous pensons à lui de manière opérationnelle dés la première étape du cycle de vie. Une des préoccupations majores est que nous puissions lui fournir des logiciels qui soient réellement à son service et non l'inverse.

Ce module présente l'interface qui donne à l'utilisateur la possibilité de choisir et d'accéder au logiciel du travail et au concepteur la possibilité d'accéder à la partie du système consacrée à la conception des logiciels.

❖ **Architecture du module constructeur de requêtes SQL**

L'architecture du module constructeur de requêtes SQL est présentée par le schéma décrit par la « figure III-62 » suivante :

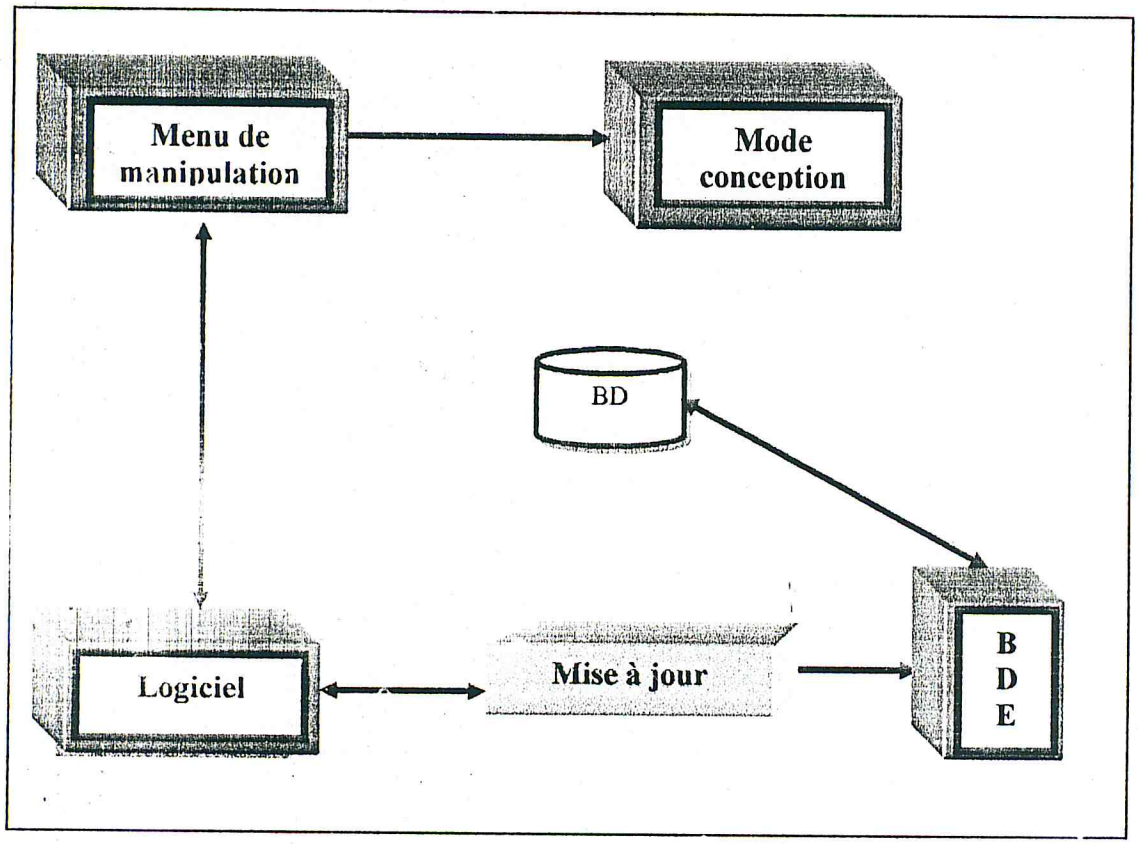


Figure III-62 : Architecture du module interface utilisateur

❖ Paquetage du module interface utilisateur

Les éléments du paquetage du module interface utilisateur sont représentés par la figure suivante :

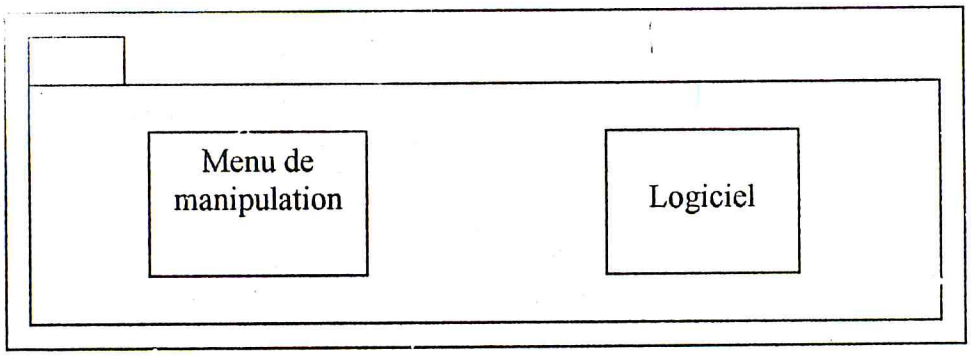


Figure III-63 : Paquetage de l'interface utilisateur.

❖ Diagrammes de collaboration

Les diagrammes de collaboration du module générateur d'interface sont représentés par le figure 64.

➢ Identification du concepteur

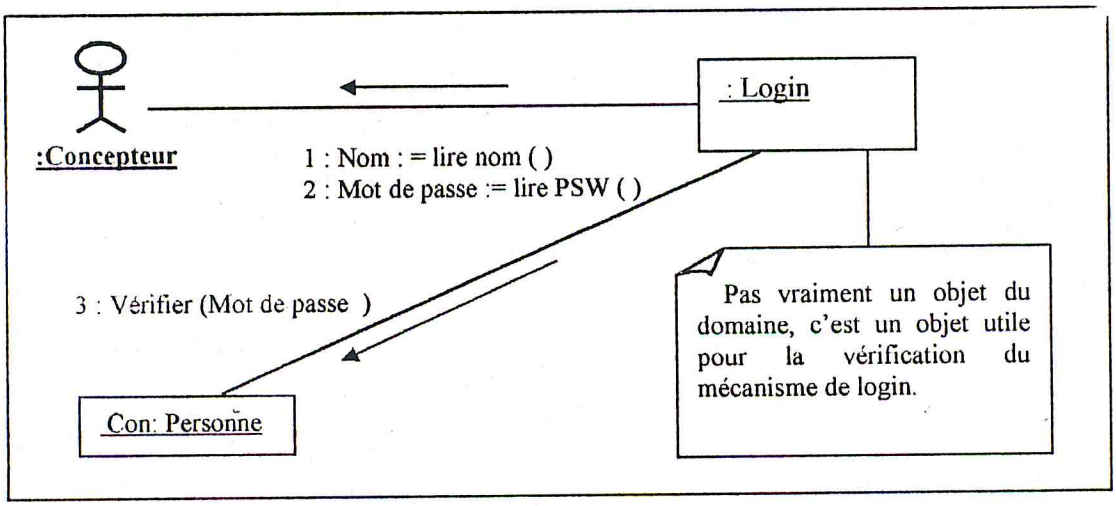


Figure III-64 : Diagramme de collaboration « Identification du concepteur »

❖ Diagramme de classes

Le diagramme de classe donné par la figure III-65 représente les éléments de modélisation qui doivent être implémentés, d'une manière synthétique :

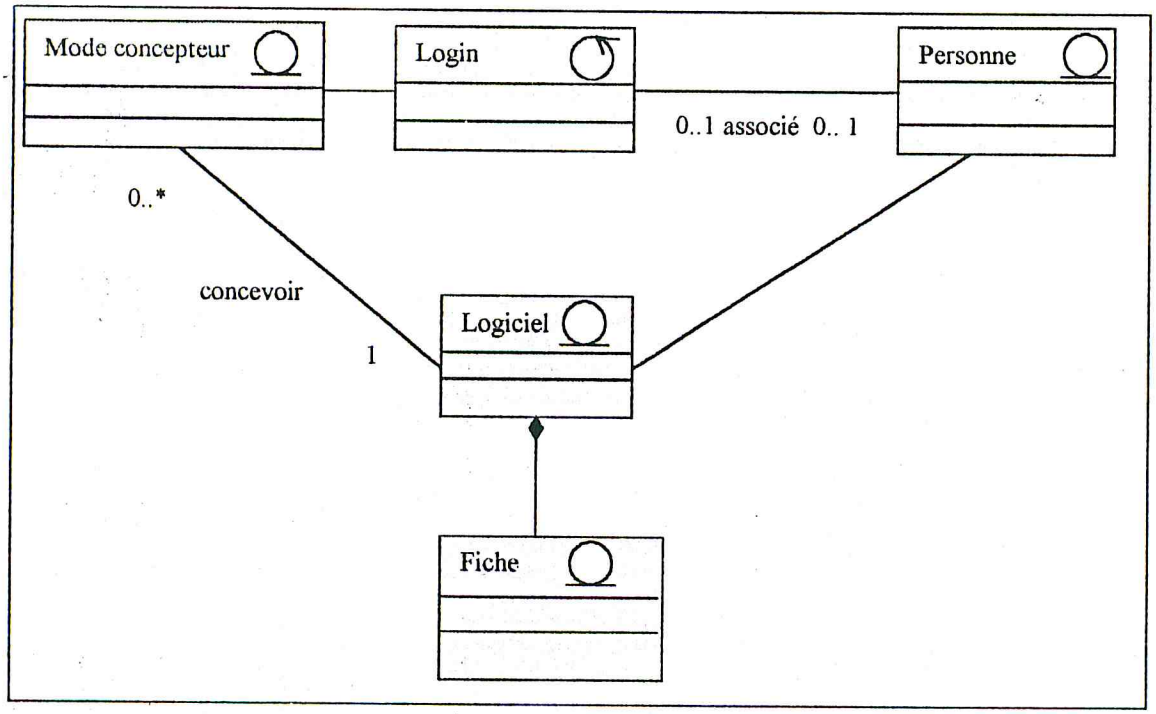


Figure III-65 : Diagramme de classes de l'interface utilisateur

❖ Diagrammes d'états-transitions et d'activités

➢ Comportement des objets de la classe login

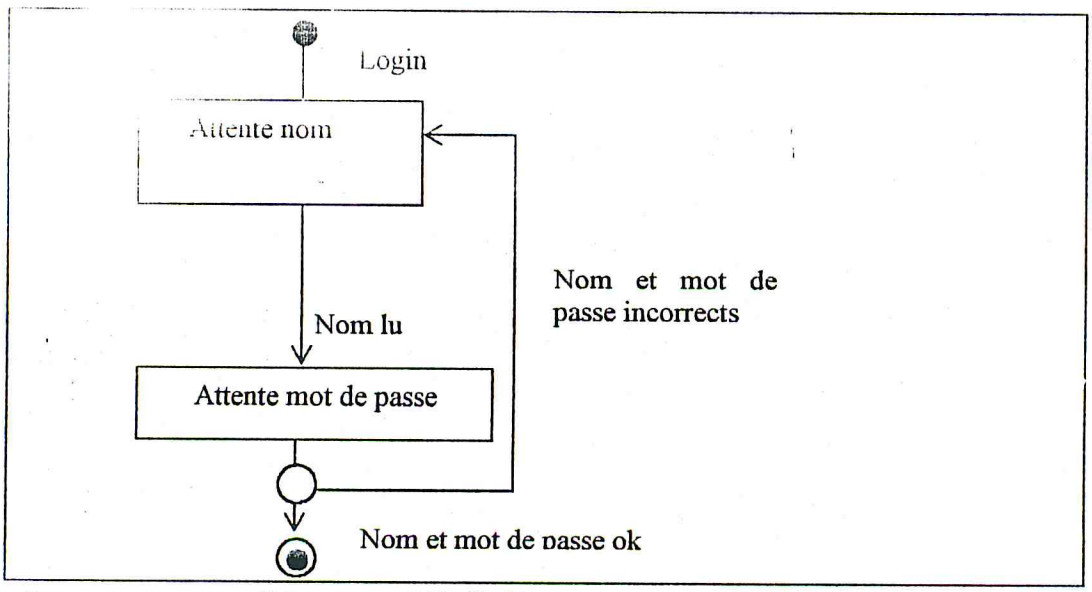


Figure III-66 : Diagramme d'états pour le comportement des objets de la classe login

➢ **Diagramme d'activités pour l'accès au mode conception**

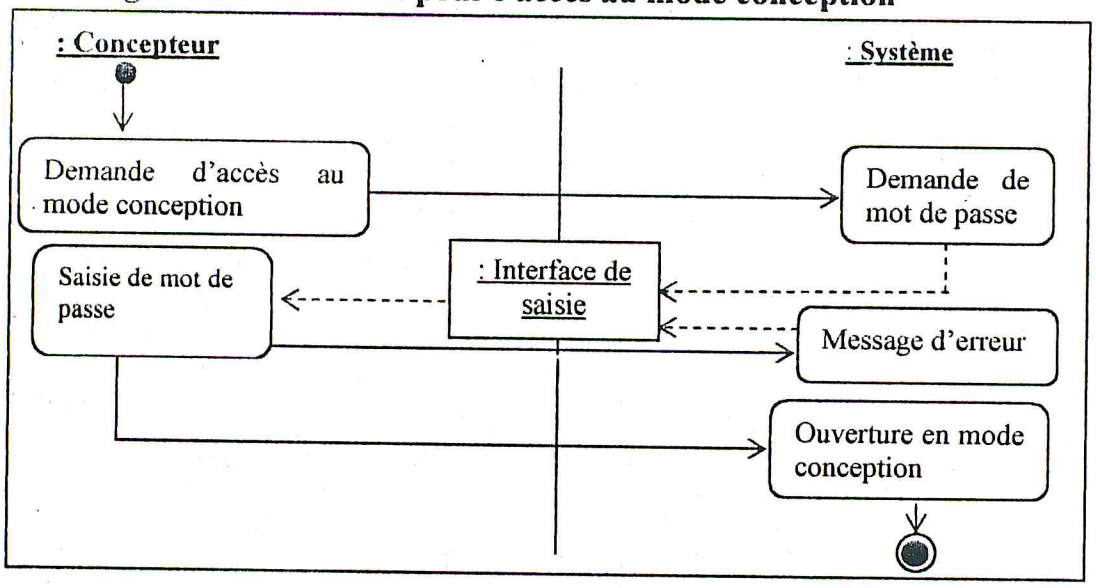


Figure III-67 : Diagramme d'activités pour l'accès au mode conception

➢ **Diagramme d'activités pour l'utilisation d'un logiciel :**

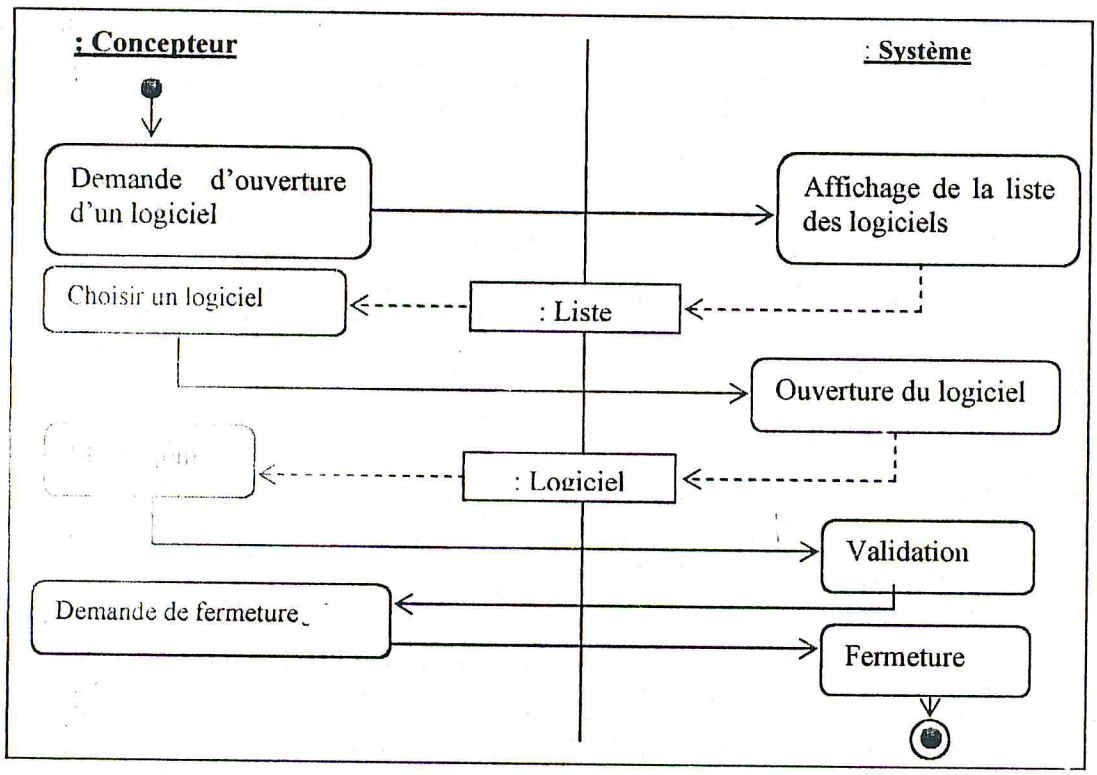


Figure III-68 : Diagramme d'activités pour l'utilisation d'un logiciel

VI- Conclusion :

Dans ce chapitre nous avons présenté un standard de modélisation et de spécification de données : UML. Puis nous avons d'aborder la conception de notre système, en commençant par la conception globale puis la conception détaillée de chaque module du système.

Chapitre IV

I- Introduction

Dans ce chapitre nous exposons les détails de mise en œuvre du schéma conceptuel décrit dans le chapitre précédent. Nous commençons par décrire l'implémentation de la solution choisie, puis, la tester et en fin, la valider.

II- Implémentation

A ce niveau, il s'agit de l'implémentation de la solution retenue au niveau de la phase de conception et de faire les programmes nécessaires pour cette implémentation.

II-1- Outils de développement

Une des décisions la plus importante dans l'implémentation d'un logiciel est le choix du langage et outils de programmation. Le langage choisi devrait permettre d'utiliser des noms significatifs. Il devrait y avoir des procédures, des fonctions et des contrôles. D'autres critères doivent être pris en compte.

- **Langage de programmation**

Traditionnellement, Delphi et les bases de données font bon 'ménage'. Delphi est connu pour être l'outil idéal pour développer des frontaux de bases de données.

Delphi 5 entreprise, est un logiciel de développement rapide (RDA :Rapide Application Développement)conçu par Borland pour développer plus rapidement et plus facilement des applications sous Windows.

- **Système de gestion de base de données**

Vu l'importance du nombre des contrats et sociétaires qui sont inscrit dans la plus part des entreprises, nous avons choisi un SGBD qui supporte le plus grand nombre d'enregistrements et qui offre la possibilité d'augmenter la taille de la base de données au besoin, chose qui est assurée avec INTERBASE.

INTERBASE est un langage de définition et d'administration de données en même temps interactif et bien adapté à la programmation avec des données. C'est un SGBD relationnelles complet.

II-2- Programmation

Cette activité consiste à passer du résultat de la conception détaillée à un ensemble de programmes ou de composants de programmes.

Pour extraire les information des bases de données et les utiliser dans la programmation, nous avons accédé aux tables systèmes de ces bases de données et nous avons étudié la relation entre les différents champs de ces tables (contraintes, clés primaires, clés étrangères, champs...).

Et pour créer les composants utilisés par le générateur d'interface, nous avons utilisé l'orienté objet et nous avons considéré l'héritage des propriétés des composants de Delphi .

III- Test du logiciel

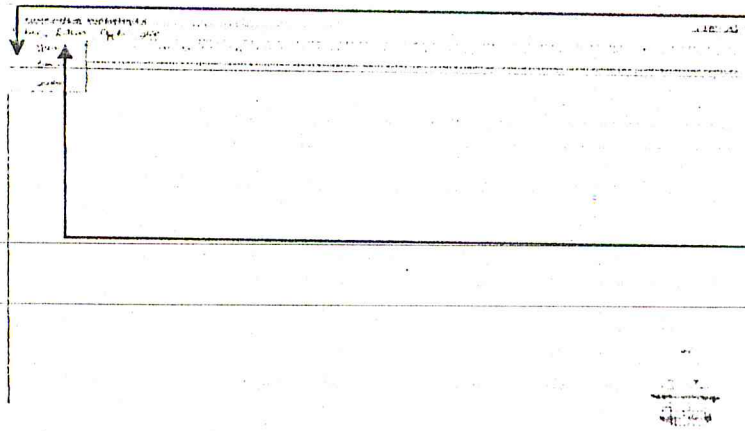
L'objectif du test d'un logicielest de détecter les erreurs. La mise au point a pour but de localiser ces erreurs et de les corriger.

Le test permet de réaliser des contrôles pour la qualité du système. Il s'agit de relever les éventuels défauts de conception et de programmation (revue de code, tests des composants,...) [Som, 88].

Pour tester notre logiciel, notre choix d'application est fait sur la gestion des projets dans la Caisse Nationale de la Mutualité Agricole (CNMA).

Dans cette section nous allons présenter des copies d'écran de notre logiciel, cette section peut être considérée comme un manuel d'utilisation de notre logiciel puisque nous allons passer par presque toutes les étapes d'utilisation :

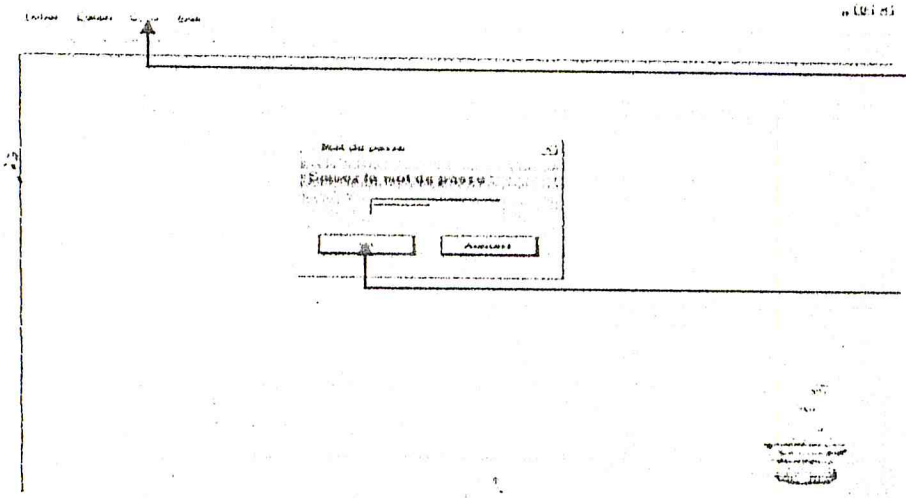
Lors de l'ouverture du logiciel, une interface contenant un menu va apparaître. L'utilisateur peut ouvrir ou fermer un logiciel conçu par le concepteur, ou accéder au mode conception après la saisie du mot de passe. Un agent intelligent est présent pour aider le concepteur dans la conception.



Le menu fichier permet l'ouverture ou la fermeture de l'application.

Le menu édition permet de copier, de couper ou de coller des textes.

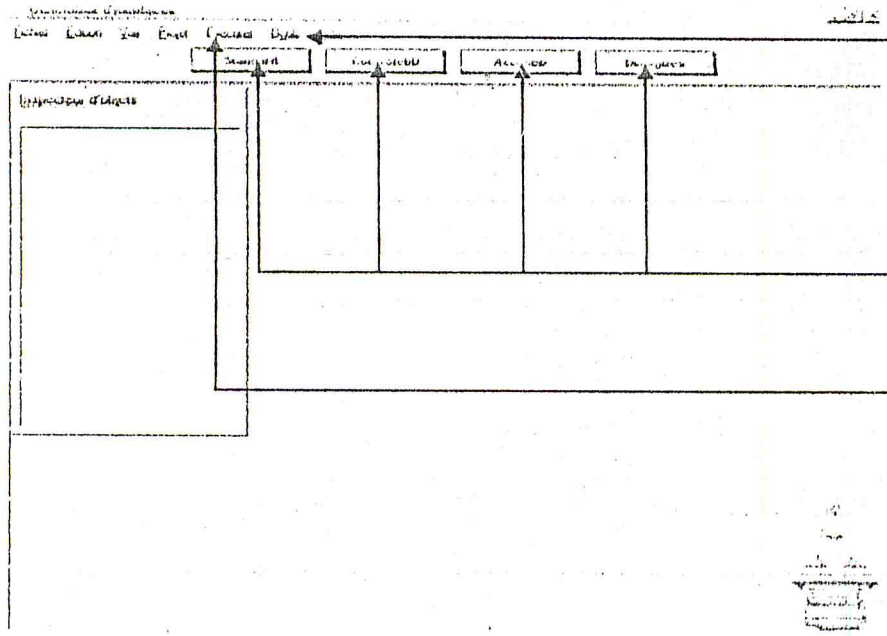
Figure IV-1 : Ouverture de l'application.



Le menu outil permet l'ouverture de la boite de mot de passe.

Validation du mot de passe.

Figure IV-2 : Accès au mode conception



Le menu outil permet l'accès au constructeur SQL ou à l'éditeur graphique.

Boutons de composants.

Menu exécuter permet l'exécution du projet et le retour au mode conception.

Figure IV-3 : Mode conception

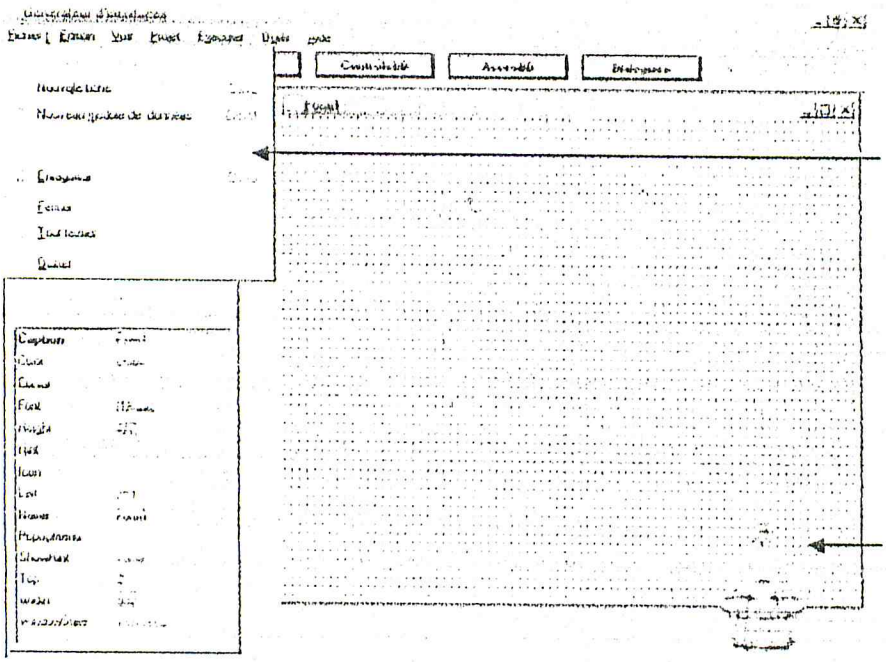


Figure IV-4 : Ouverture d'un nouveau projet.

Pour modifier les options du projet comme le nom du projet, le chemin d'exécution et la fiche principale, le concepteur doit utiliser le menu projet.

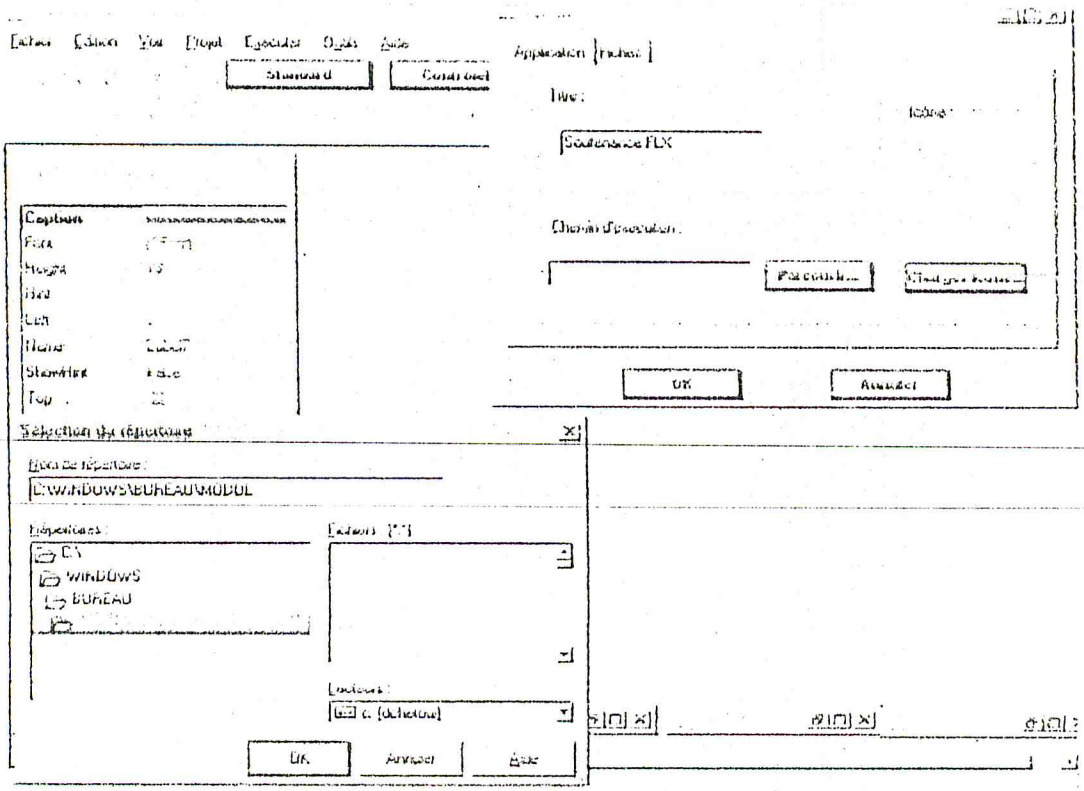


Figure IV-5 : Options du projet.

Pour ouvrir une autre fiche le concepteur doit sélectionner nouvelle fiche dans le menu fichier, puis, sélectionner une fiche référentielle.

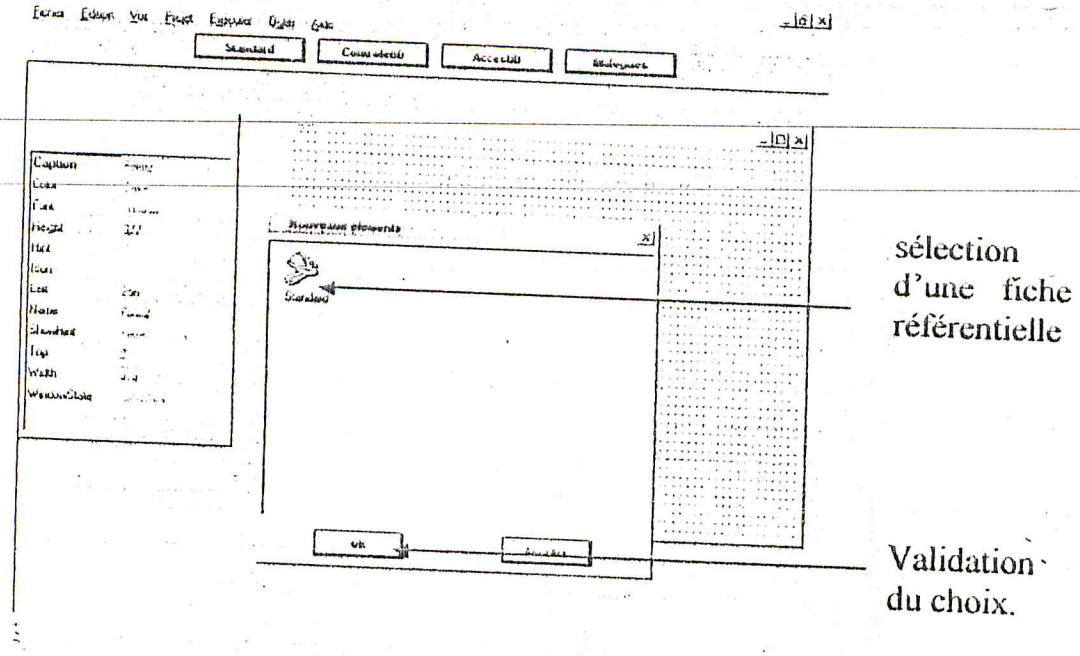


Figure IV-6 : Nouvelle fiche.

Pour créer une fiche référentiel, le concepteur doit cliquer par la droite sur la fiche et l'ajouter aux référentielles.

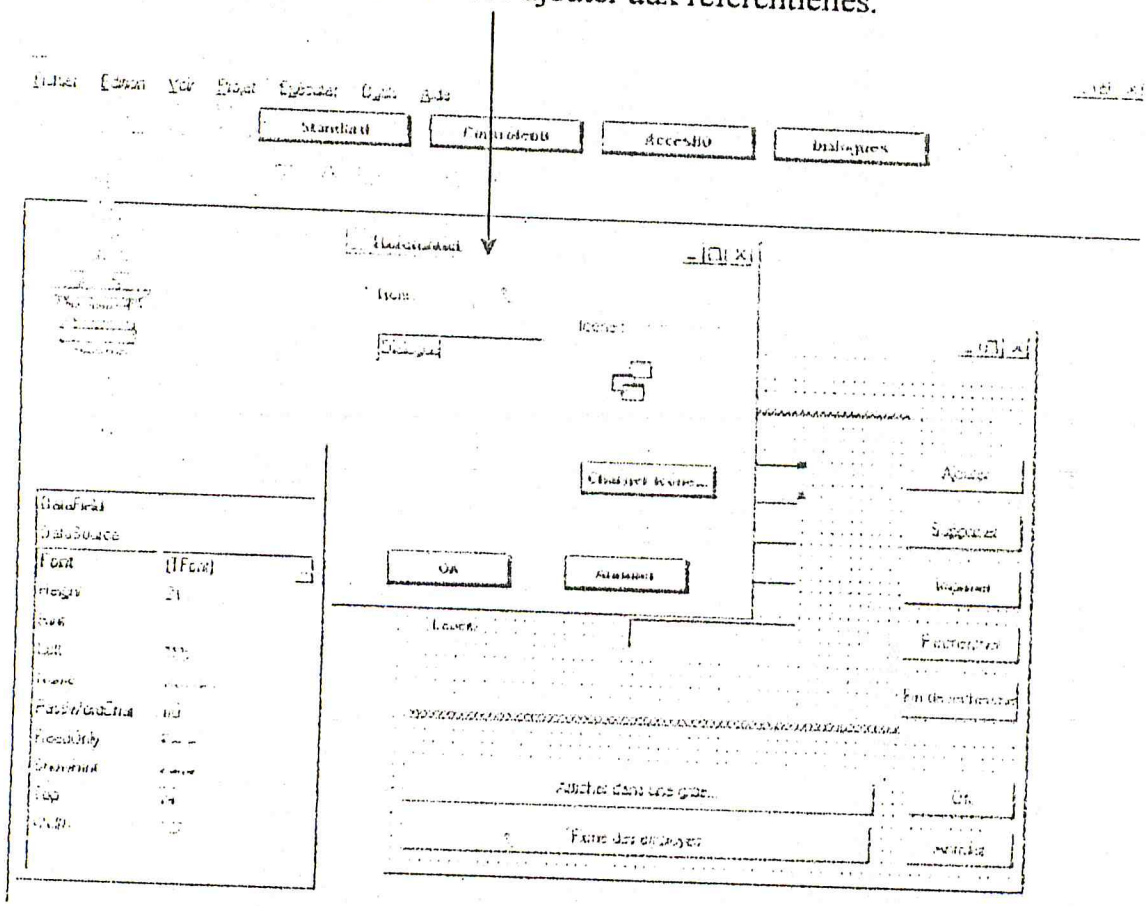


Figure IV-7 : Ajouter aux référentielles.

A l'aide des barres d'outils, le concepteur affiche les composants sur la fiche pour construire l'interface. Pour modifier les propriétés des composants, il utilise l'inspecteur d'objet.

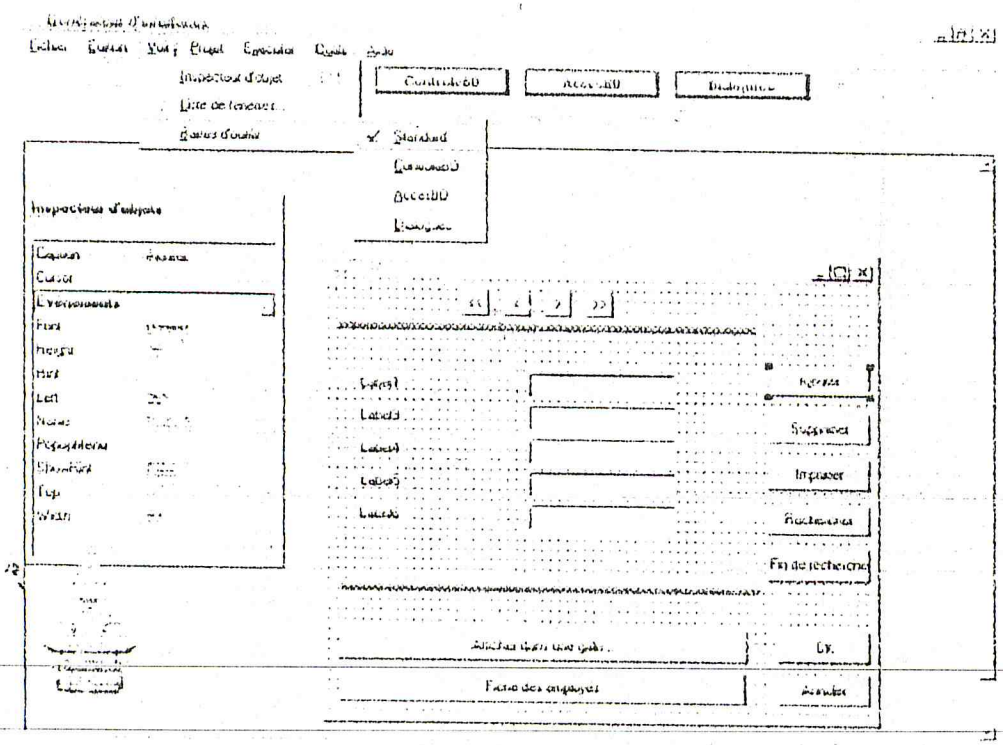
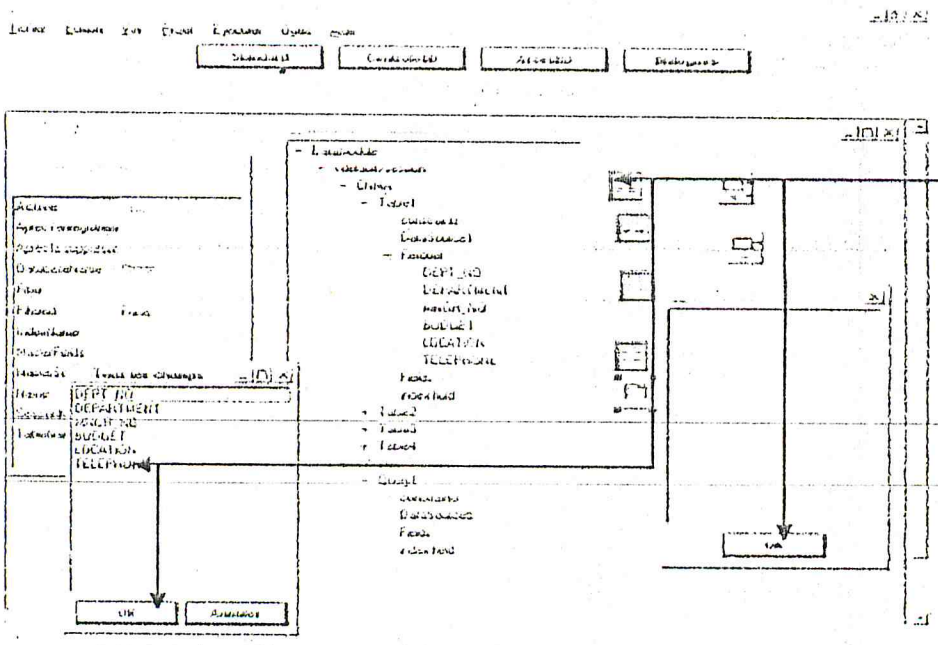


Figure IV-8 : Génération de l'interface

Pour relier les composants de la BD avec la BD, le concepteur doit ouvrir un module de données à l'aide du menu fichier, afficher les composants non visuels de BD et les relier avec la BD à l'aide de l'arbre et l'inspecteur d'objet



Pour créer les champs le concepteur doit cliquer par la droite sur le DataSet et sélectionner les champs.

Figure IV-9 : Relation avec la base de données.

Pour ajouter un champ calculé le concepteur doit cliquer par la droite sur la table est choisir Nouveau champ.

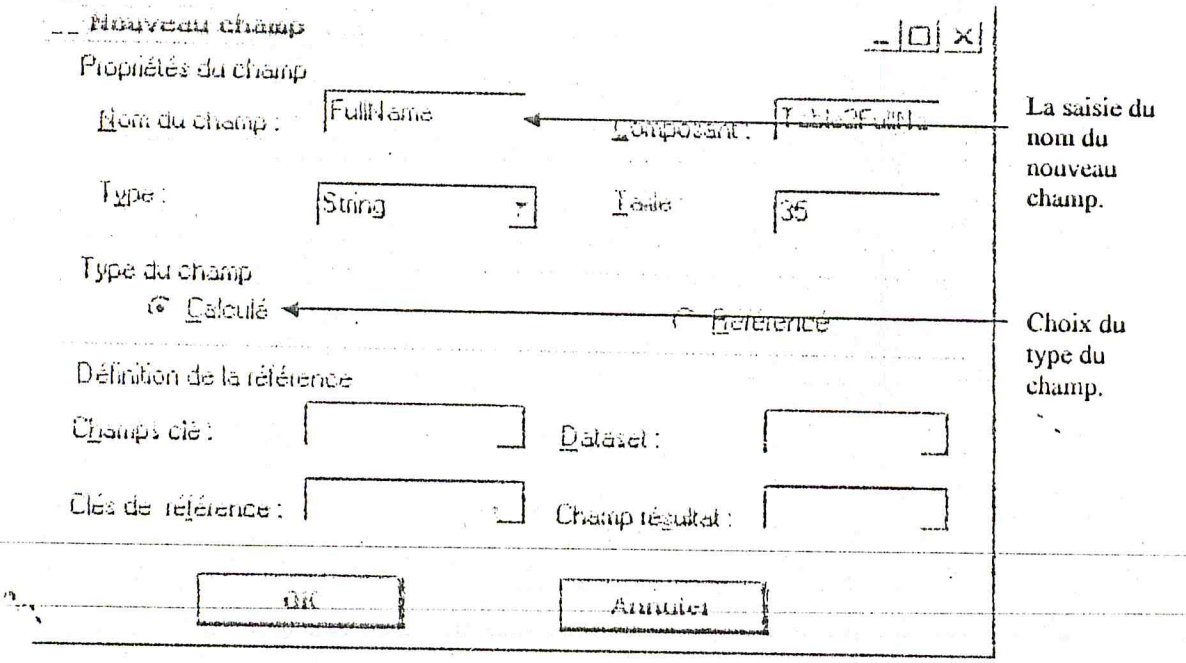


Figure IV-10 : Nouveau champ.

Après l'affichage du composant Main menu, il doit sélectionner la propriété Item dans l'inspecteur d'objet et spécifier les menus qui seront affichés.

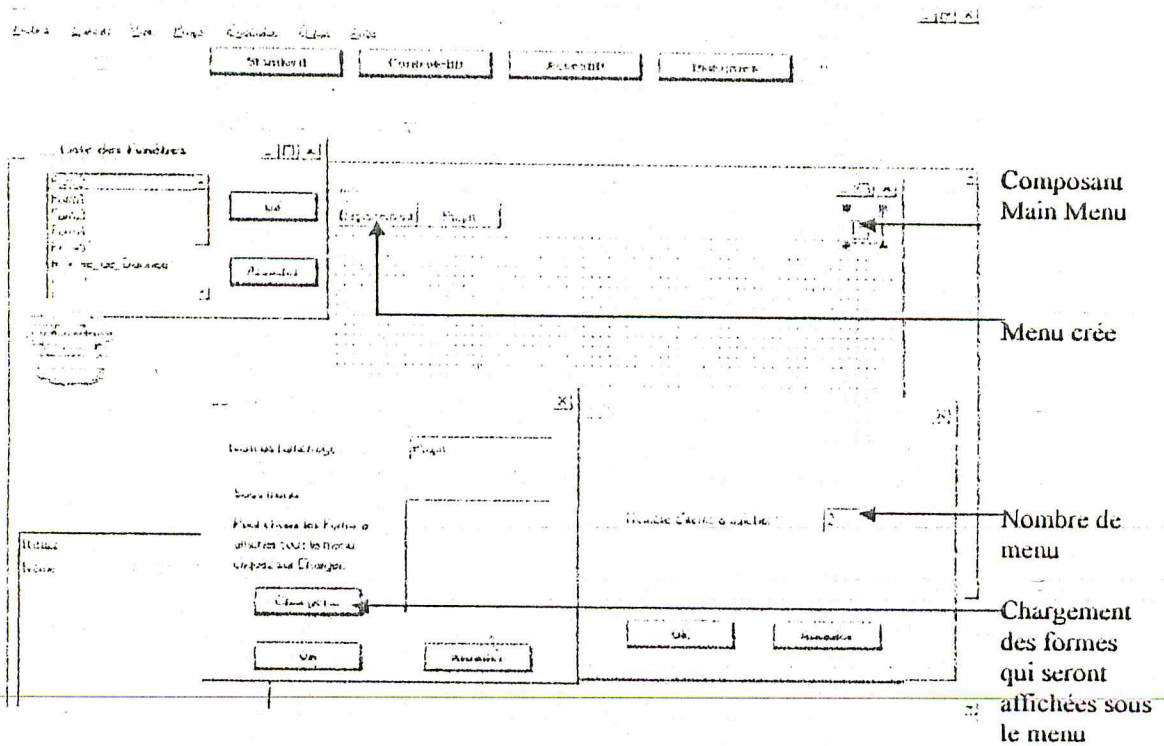


Figure IV-11 : Affichage de main menu.

Pour affecter les évènements aux boutons, il doit sélectionner la propriété évènement de l'inspecteur d'objet, puis l'évènement et le DataSet ou fiche correspondants.

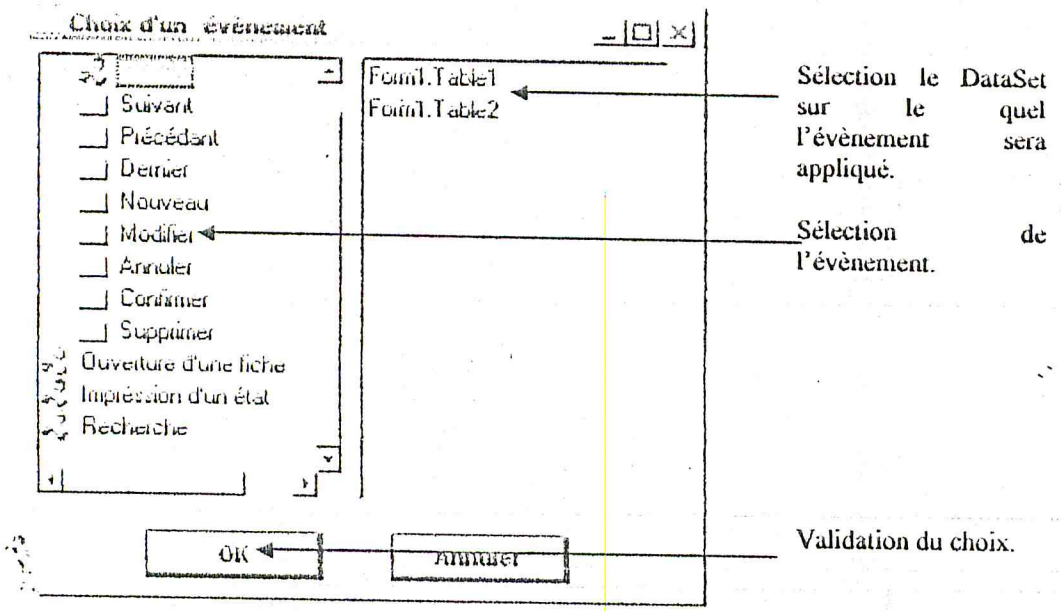


Figure IV-12 : Choix d'évènement.

Pour modifier le format d'affichage d'un champ, il doit sélectionner la propriété format d'affichage de l'inspecteur d'objets correspondant au champ.

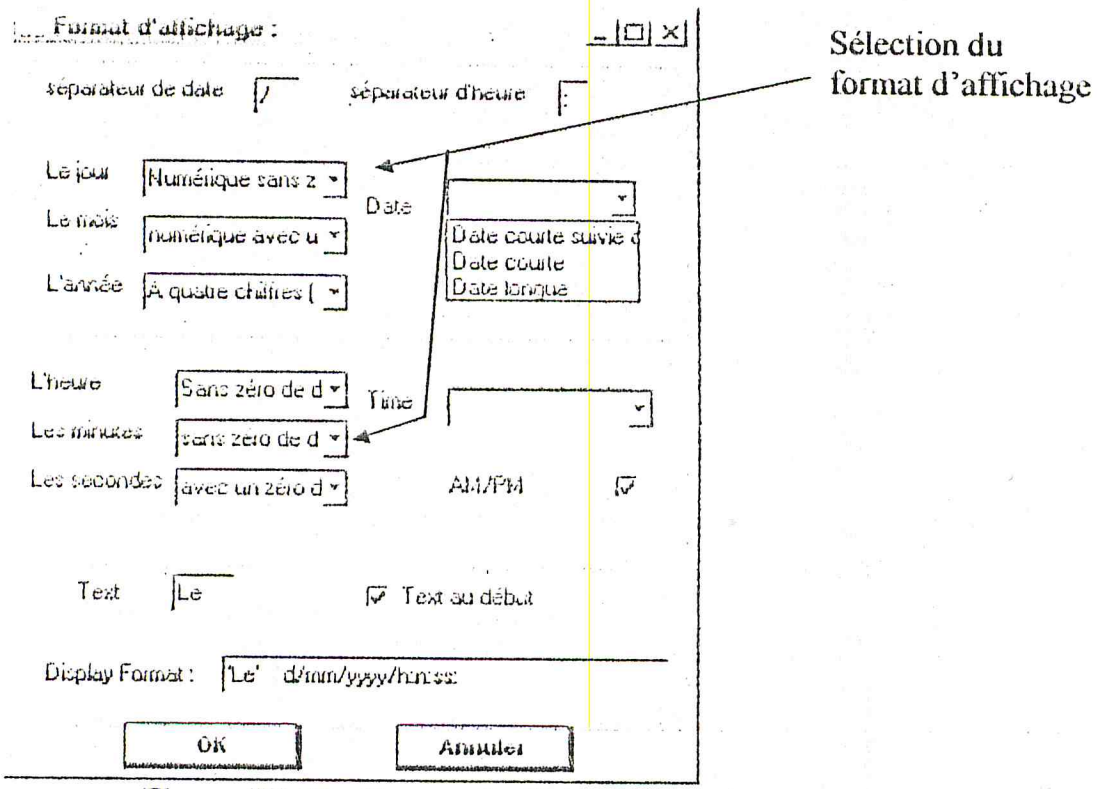


Figure IV-13 : Format d'affichage d'un champ.

Le concepteur peut affecter une expression à un champ, en sélectionnant le champ et la propriété expression dans l'inspecteur d'objet.

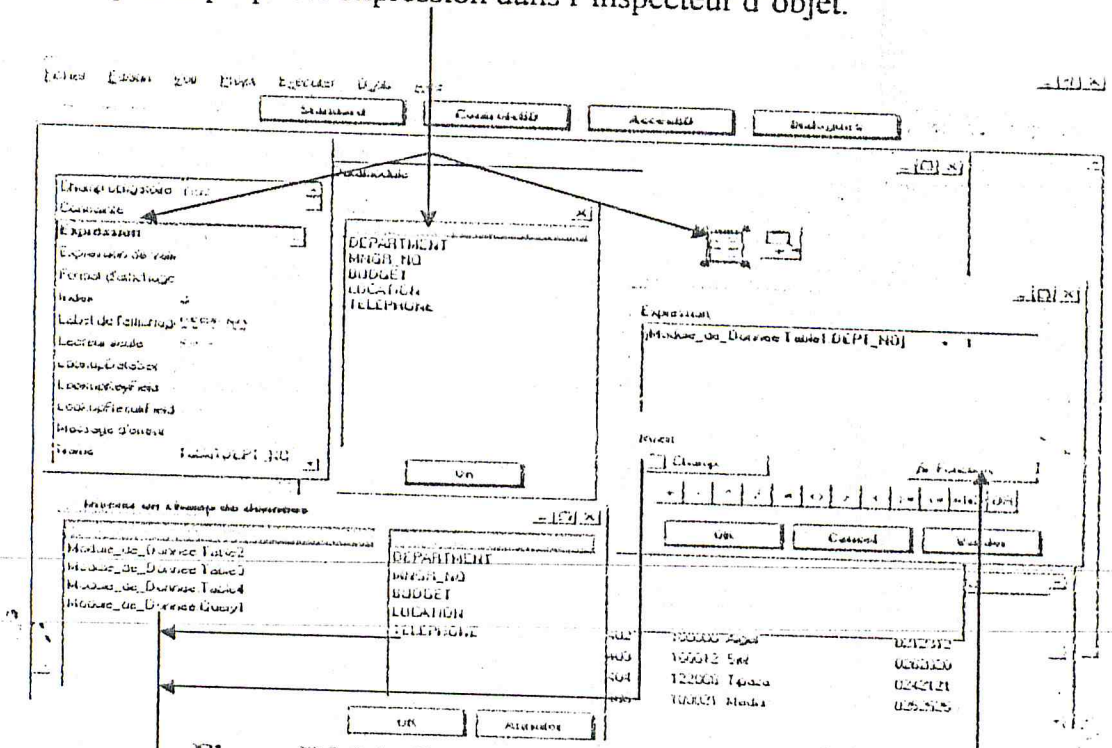


Figure IV-14 : Expression pour un champ

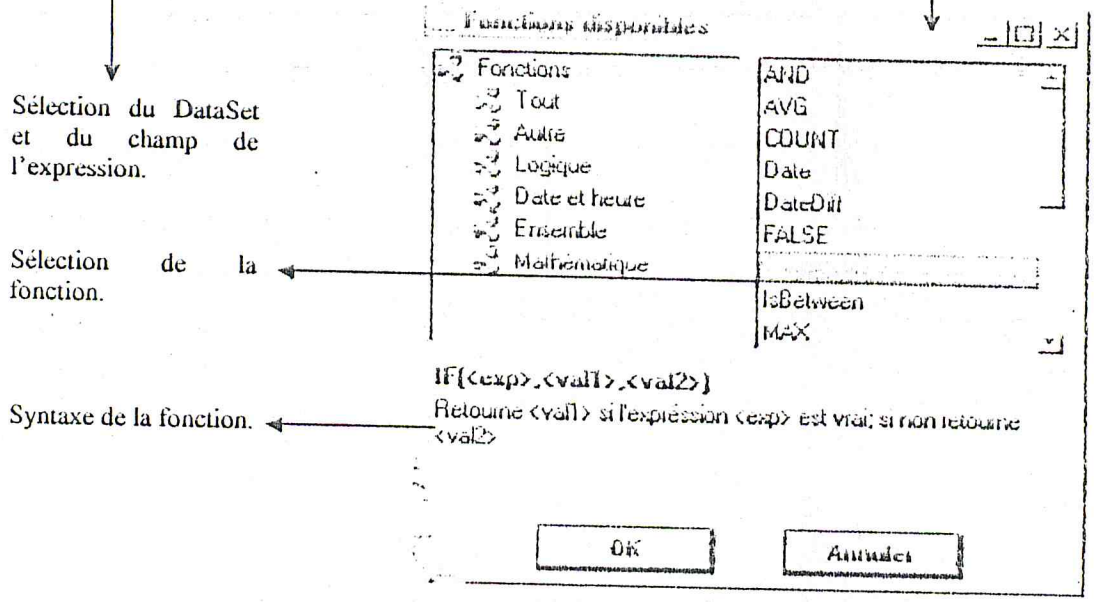


Figure IV-15 : Fonction disponibles.

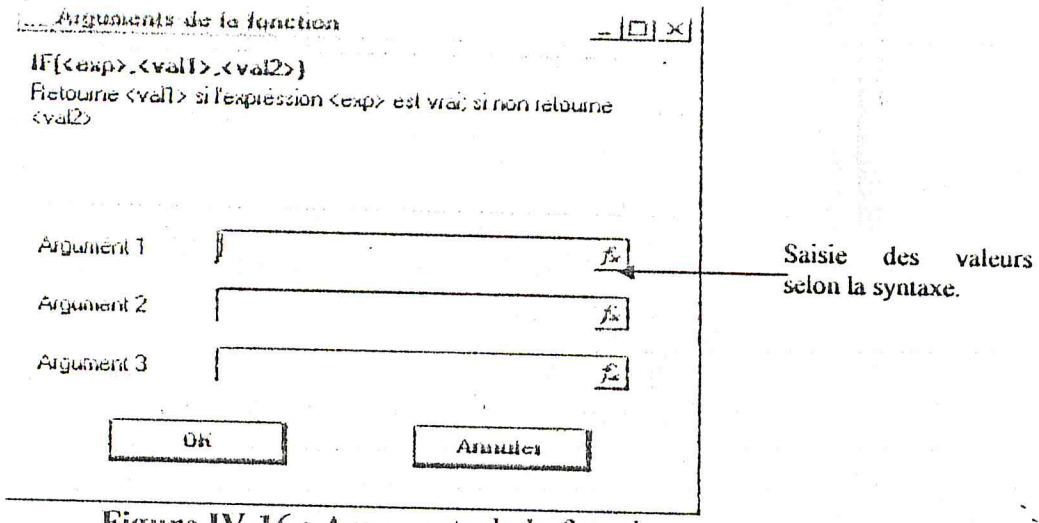


Figure IV-16 : Arguments de la fonction.

Pour l'impression, le concepteur doit afficher un composant d'état de sortie et un DataSet. Puis, sélectionner la propriété désignée pour générer l'interface de l'état. Pour notre exemple nous avons pris un TQuery et nous avons sélectionné un fichier SQL construit par le constructeur SQL.

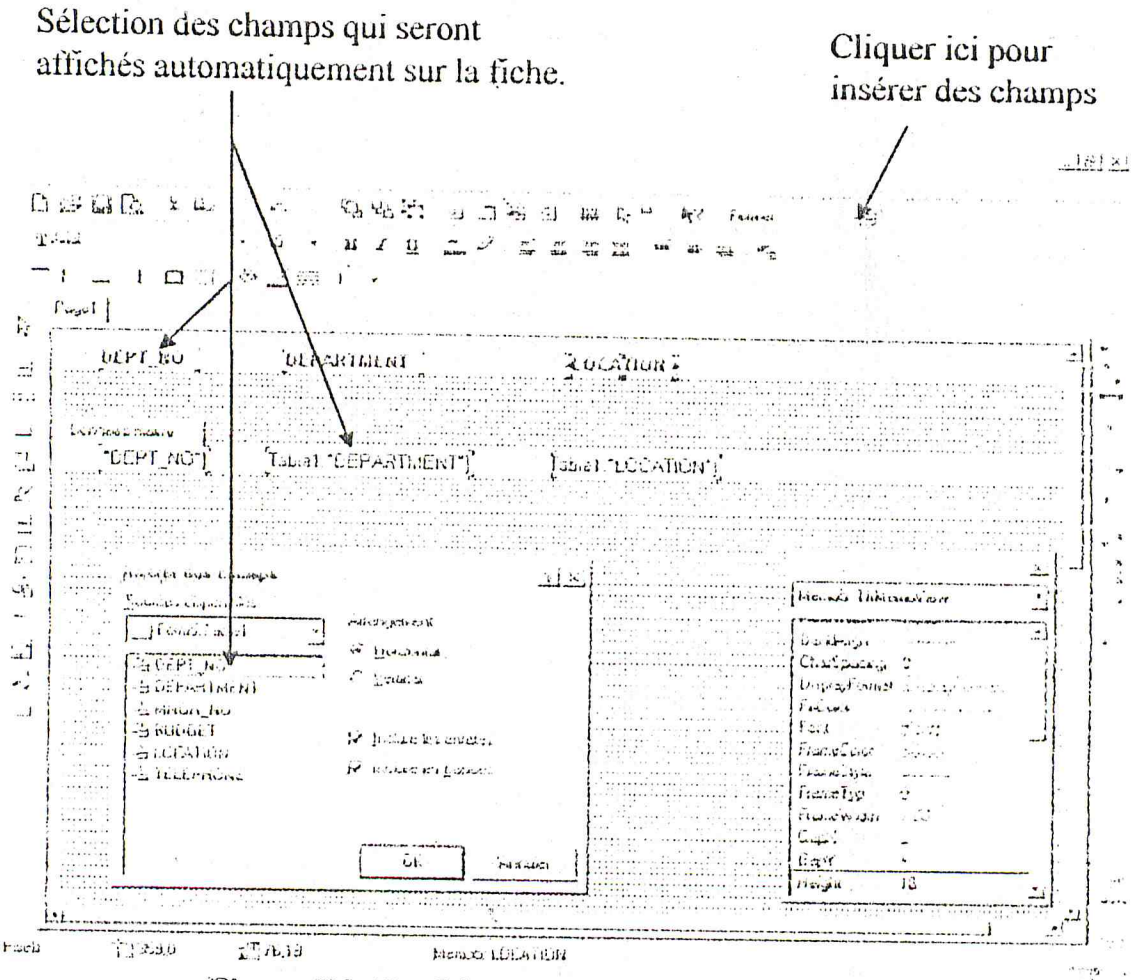


Figure IV-17 : Génération de l'état de sortie.

L'état de sortie généré et le suivant :

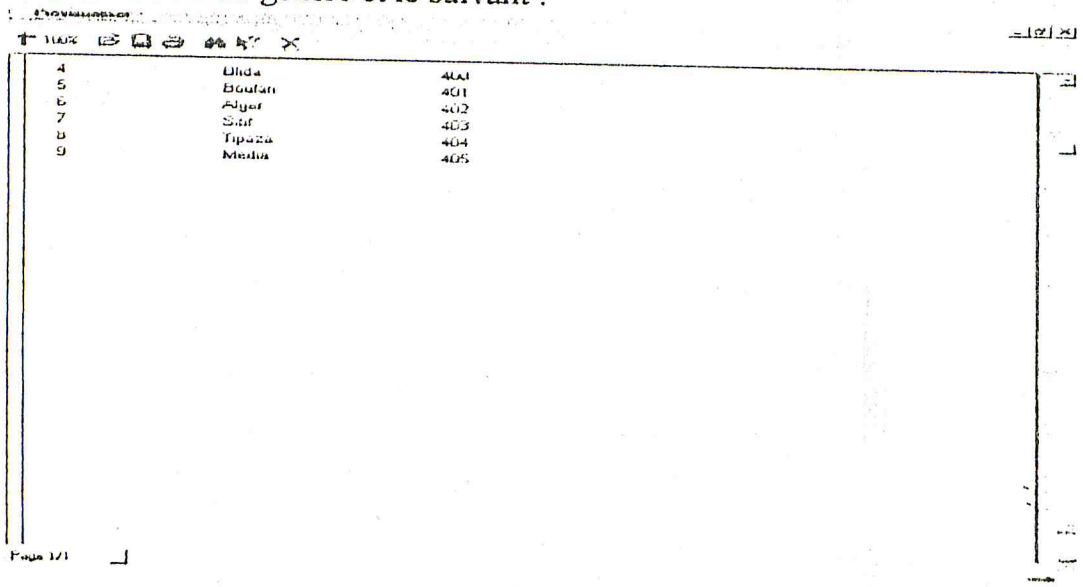
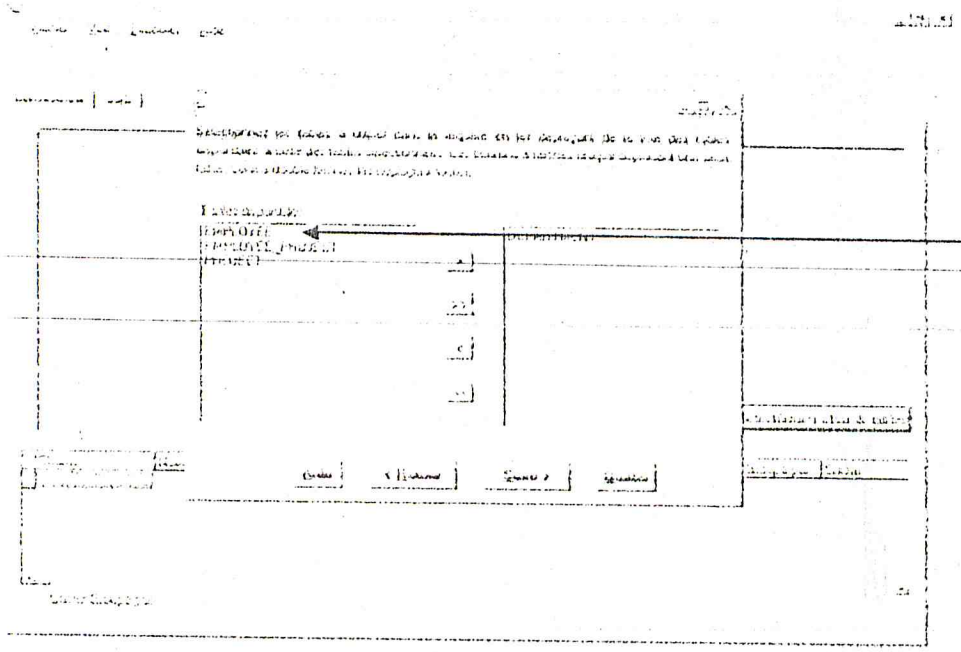


Figure IV-18 : Etat de sortie.

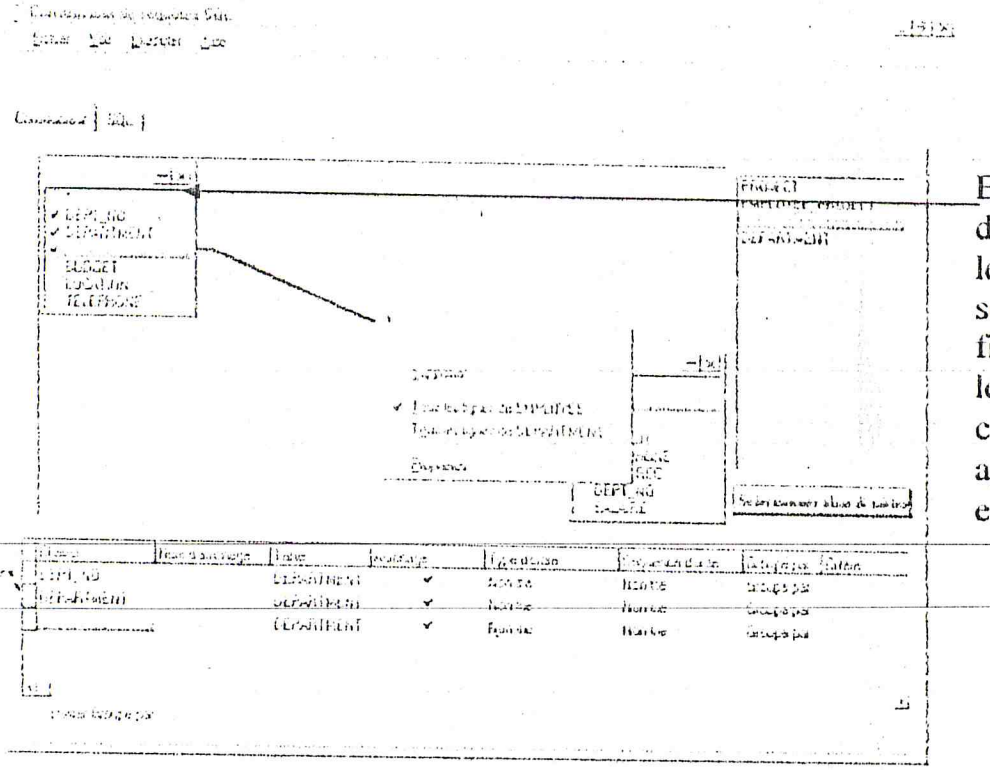
Pour construire la requête SQL, le concepteur sélectionne le constructeur SQL dans le menu outils dans. Puis l'alias de la BD de travail, pour notre exemple l'alias est appelé CNMA, et enfin, les tables.



Sélection des tables utilisés dans la requête.

Figure IV-19 : Constructeur de requête.

Pour construire la requête, le concepteur relie les champs d'égalité, sélectionne les champs qui doivent s'afficher et choisi les propriétés du lien.



En déplaçant les table sur la fiche, leurs champs apparaîtr-ent.

Figure IV-20 : Relation entre les tables de la requête

Le concepteur peut voir et saisir la requête :

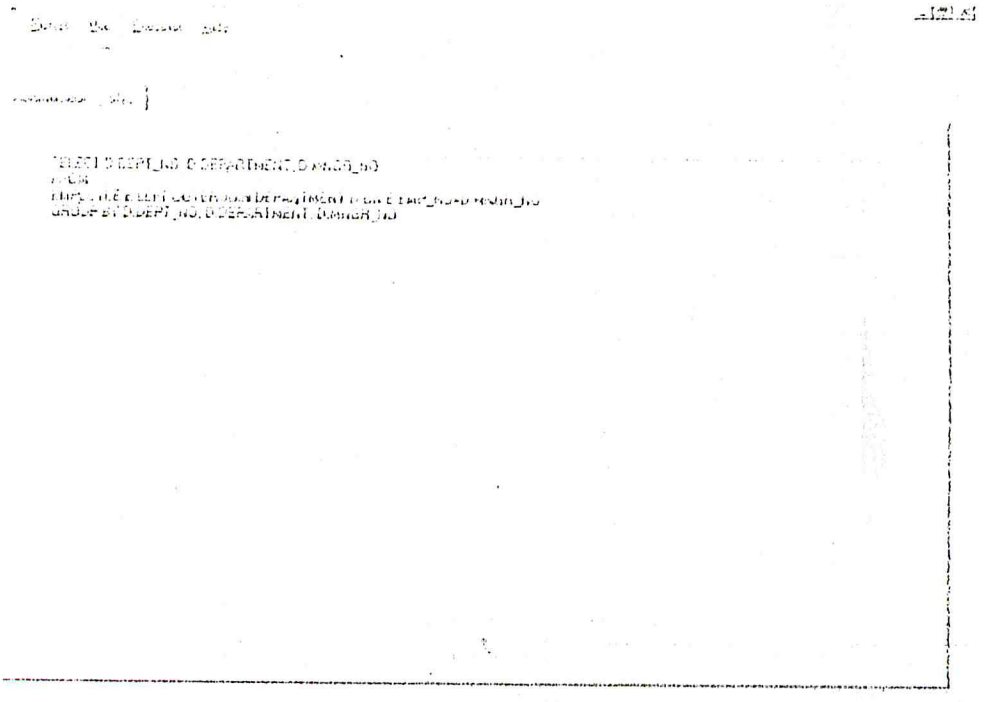


Figure IV-21: Requête SQL.

Pour les critères de la requête, le concepteur clique sur un critère du tableau, puis construit l'expression à l'aide du constructeur d'expression.

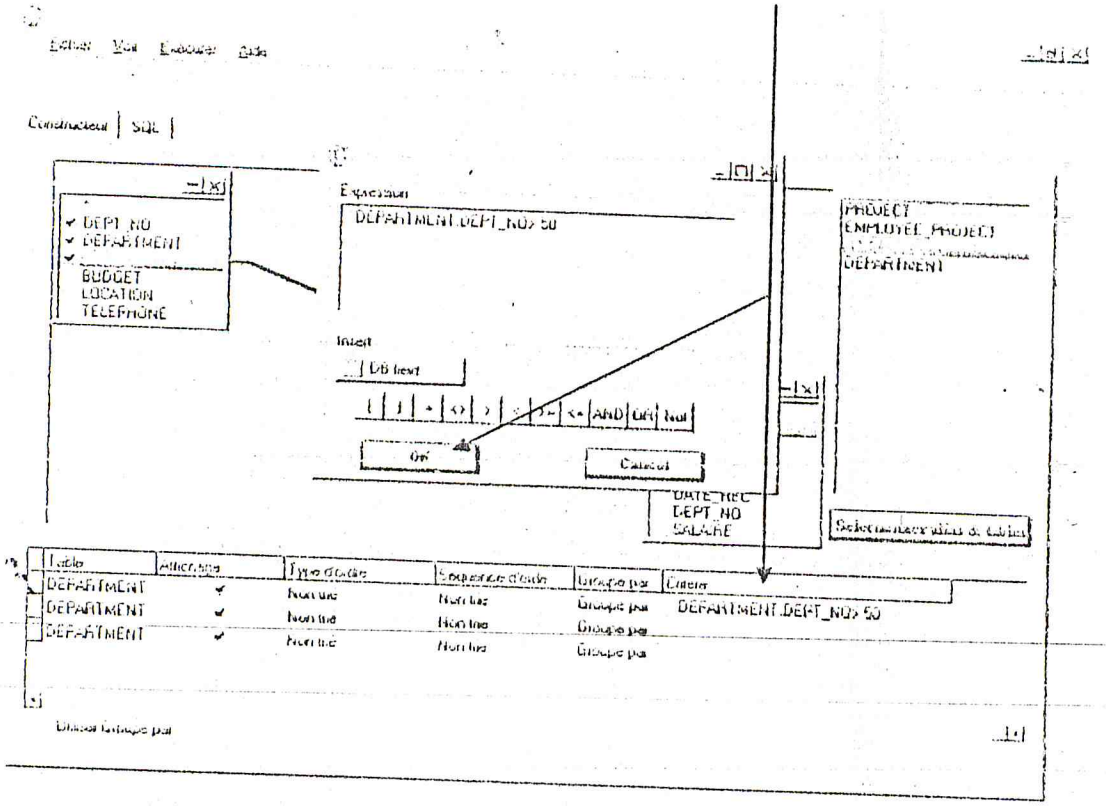


Figure IV-22 : Critères de la requête.

Après l'exécution, le résultat de la requête s'affiche dans une grille. Avant de quitter une demande de sauvegarde s'affichera pour sauvegarder la requête.

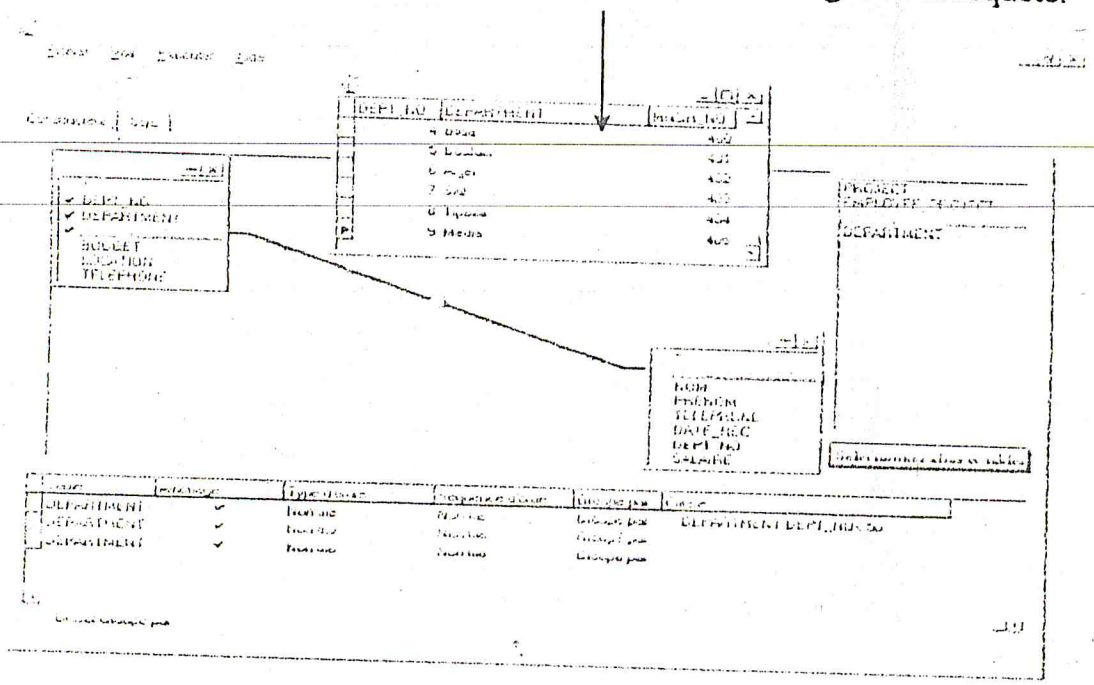
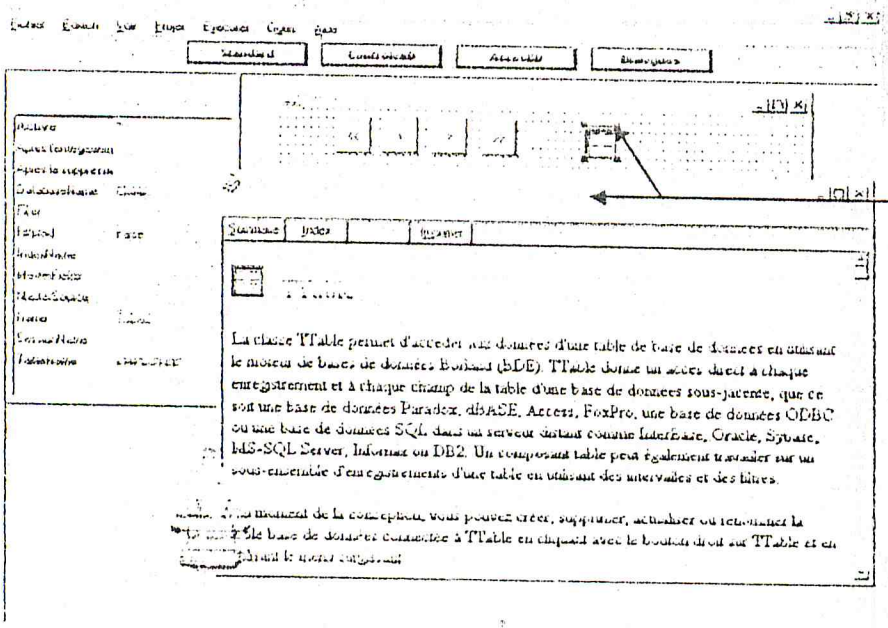


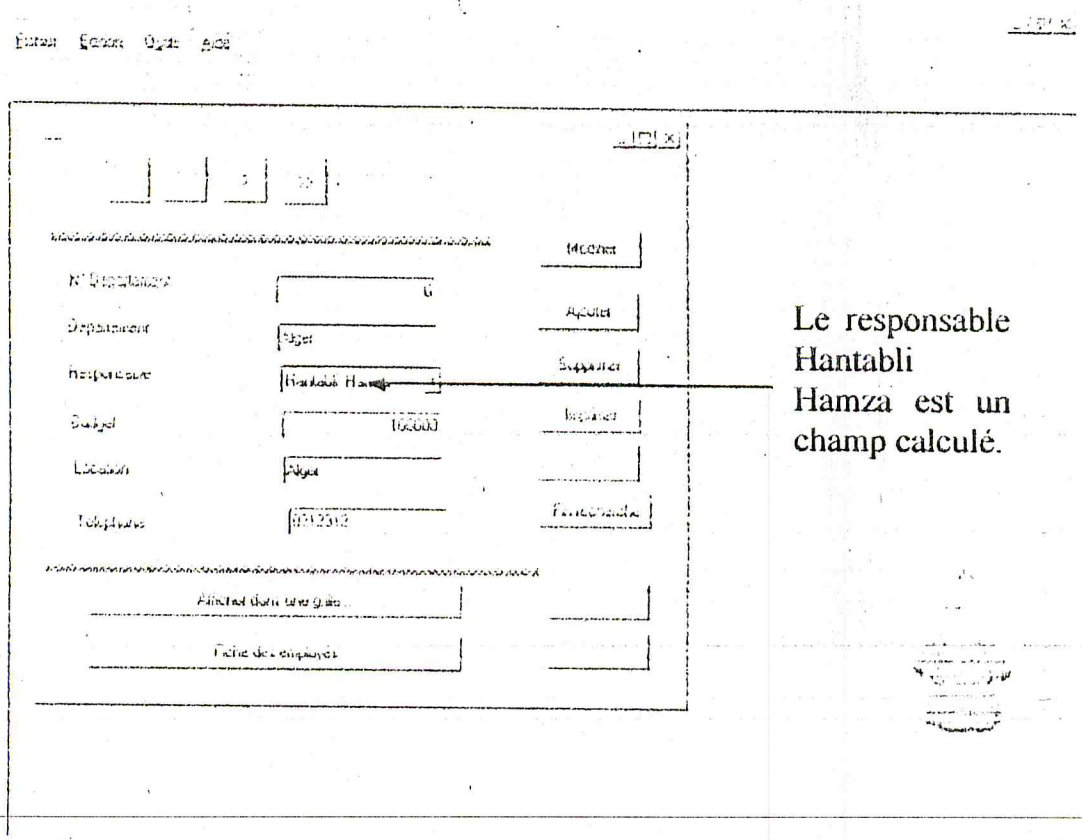
Figure IV-23: Résultat de la requête.



En sélectionnant un composant et cliquant sur F1 une fenêtre d'aide s'affiche.

Figure IV-24: L'aide de l'application.

Le concepteur exécute le projet, pour qu'un utilisateur puisse l'ouvrir pour le travail. L'utilisateur utilise les boutons de mise à jour, de parcours et d'ouverture de fiches simplement. Quand une contrainte est non prise en compte, un message s'affiche.



Le responsable Hantabli Hamza est un champ calculé.

Figure IV-25: Recherche des enregistrements

Pour la recherche des enregistrements, une interface d'expression s'affiche en cliquant sur le bouton de recherche.

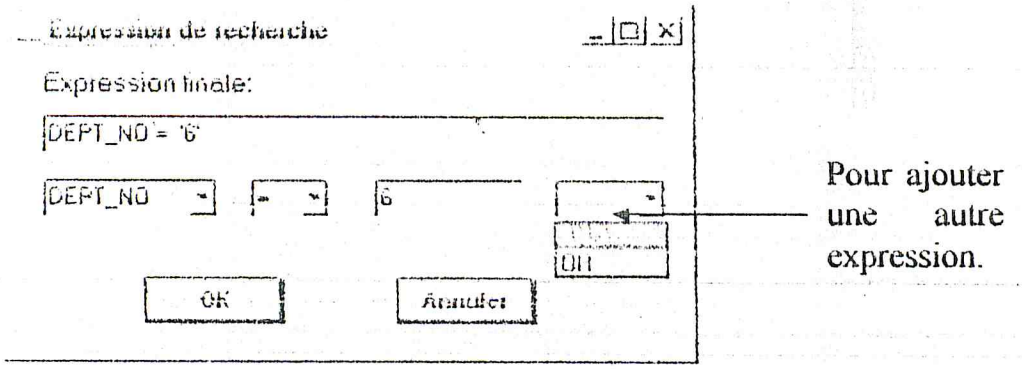


Figure IV-26: Expression de recherche.

Le résultat de la recherche s'affiche sur les masques de saisie de la forme pour terminer la recherche, et le concepteur clique sur le bouton Fin Recherche.

IV- Validation

La validation doit être envisagée lors de l'achèvement du travail de développement, une fois que la qualité technique du système est démontrée. Elle permettra de garantir la logique et la complétude du système [Brès, 93].

Nous avons choisi de faire la validation par test, le test précédant que nous avons fait montrer que les cas d'utilisation proposés dans le chapitre précédant sont validés.

V- Conclusion

L'application a été réalisée avec l'idée d'offrir à l'utilisateur tous les services de notre système, sous une forme agréable et convivial. Indiquant tout de même que sa réalisation a été simplifiée par les remarquable richesses et fonctionnalités des composants visuels proposés par l'environnement de développement utilisé.

Conclusion générale

Conclusion générale

L'objectif principal de ce mémoire consiste en la réalisation d'un système flexible qui pourra évoluer à fin de s'adapter aux changements de règles de gestion et/ou aux nouveaux besoins des utilisateurs sans intervention dans les fichiers sources de l'application.

La présence d'un processus de développement formalisé, bien défini et bien géré est un facteur de réussite d'un projet. Pour cette raison nous avons suivi le modèle en cascade, utilisant l'UML comme langage de modélisation.

Au cours de notre projet nous avons développé un système flexible de gestion et d'aide à l décision, entièrement dédié à la génération des interfaces utilisateur. Il permet grâce à son ergonomie, au concepteur de concevoir un logiciel qui répond aux besoins de l'utilisateur, en procédant par des outils simple, et tout cela sans avoir recours à la programmation et à la vérification des relations entre les tables de la base de données.

Ce système est simple d'utilisation, accessible et convivial, permettant à l'utilisateur final d'accéder au logiciel conçu par le concepteur et l'exploiter (consulter, insérer, modifier, mètre à jour, supprimer) en des temps record.

Le système ainsi développé, offre donc des nombreux avantages aux utilisateurs notamment le gain du temps, la minimisation des erreurs… , notons que le système peut faire l'objet d'intégration dans de grands progiciels (assurance, banque,...) pour permettre leurs extensibilité et de les dotés de sa flexibilité. D'où l'indépendance des clients et une durée de vie plus longue pour les progiciel.

Bibliographie

Bibliographie

[Ber, 02] F. Bernardi, « Méthode d'analyse orienté objet UML », Dunod, 2002.

[Bow, 02] S.Bowman, Emerson, Darnovsky, « L'intro SQL », CampusPress, 2002.

[Bre, 93] J.Bres, « ateliers de génie logiciel », Masson, 1993.

[Bri, 88] Briand.R, « Méthode de développement des systèmes experts », Eyrolles, 11988.

[Bru, 01] Bruno Bouzy, Cycle de vie de logiciel, Eyrolles, 2001 .

[Chr, 90] Claud Chrisment, Mise en oeuvre des bases de données, Eyrolles, 1990.

[D[illegible] Claude Delobel , Christophe Lécluse, Philippe Richard, « Base de données :des systèmes relationnels aux systèmes à objets », interEditions, 1991.

[Ern, 88] Christian Ernst, « Introduction aux systèmes experts de gestion », Eyrolles, 1988.

[Fan &al 00] Rémy Fannader, Hervé Lerroux, « UML principes de modélisation », Dunod, 2000.

[Gab, 98] Joseph Gabay, « Merise vers OMT et UML », Masson, 1998.

[Gar, 02] Georges Gardarin, « Base de données », Eyrolles, paris 2002.

[Gar, 89] Georges Gardarin, « Base de données relationnelle », Eyrolles, paris 1989.

[Kar, 93] Karkan.J, « Système expert :un nouvel outil d'aide à la décision », Masson, 1993.

[Ken, 01] kellani N, Mignit D.Paré P.,Rosenthal-Sabroux C, « De Merise à UMl », Eyrolles, Paris 2001.

[Kol, 97] Christophe Kolski, « Interface hommes-machines :application aux systèmes industriels complexes » Hermes, 1997.

[Lev, 93] Levin.P, « système interactif d'aide à la décision et système expert», Hermes, 1993.

[Mor, 92] José Morejon, « Principes et conception d'une base de données relationnelle», Les éditions d'organisation, 1992.

[Mes, 88] Messaoui.R, « Optimisation des performances d'un système de BD relationnelle :une approche par système expert », Thèse de doctorat, Université de Montréal, 1988.

[Mul, 97] Pierre-Alain Muller, « Modélisation objet avec UML », Eyrolles, 1997.

[1] www.developpez.com/sgbd/Plusfacile.com

Annexe A : Initiation au SQL

I-Introduction

Le SQL (Structured Query Language) permet d'interroger une base de données, d'en modifier des informations. C'est un langage universel d'interrogation des bases de données, qui permet à différents systèmes d'échanger des données entre eux.

II-Les requêtes simple

Soit 3 tables : Eleves(#NomElv, AdrElv, VilleElv), Matieres(#NomMat, Co[illegible]#NomElv, #NomMat, #Date, Note).

➢ **Interrogation simple :** Liste des élèves.

SELECT NomElv *(Qu'est ce que je dois afficher ?)* FROM Eleves; *(Où cela se trouve t-il ?)*

➢ **La close WHERE :**

Elle permet de spécifier la ou les conditions que doivent remplir les lignes choisies.

Liste des élèves habitant Toulon.

```
SELECT NomElv
FROM Eleves
WHERE VilleElv = 'Toulon';
```

Remarque : Dans la close WHERE, on peut utiliser que des propriétés qui sont dans la table sélectionnée

➢ **La close GROUP BY :**

Il est possible de subdiviser la table en groupes, chaque groupe étant l'ensemble de lignes ayant une valeur commune.

Liste des élèves par ville.

```
SELECT NomElv, VilleElv
FROM Eleves
GROUP BY VilleElv;
```

➢ **La close HAVING :**

Elle ne s'utilise qu'avec le GROUP BY et permet de donner la ou les conditions que doivent remplir ces groupes.

Liste des élèves regroupés par ville où habitent plus de 10 élèves.

```
SELECT NomElv, VilleElv
FROM Eleves
GROUP BY VilleElv
HAVING Count(NomElv) > 10;
```

➢ **La close ORDER BY:**

Elle permet de spécifier l'ordre dans lequel vont être affichées les lignes.

Liste des matières par ordre décroissant de coef., puis par ordre alphabétique de nom.

```
SELECT NomMat
FROM Matires
ORDER BY Coef Desc, NomMat Asc;
```

➢ **Récapitulatif** :

```
SELECT noms des colonnes à afficher
FROM nom de la table où se trouvent les colonnes susmentionnées
WHERE condition(s) à remplir par les lignes
GROUP BY condition(s) de regroupement des lignes
HAVING condition(s) à remplir par le groupe
ORDER BY ordre (Asc, Desc) d'affichage
```

III-Les requêtes multi-tables

Soit 4 tables : Eleves(#RefElv, NomElv, PreElv, VilleElv, ClasseElv), Classes(#NomCla, Niveau), Cours(#RefElv, #NomMat, NbHeure), Matieres(#NomMat).

➢ **Requêtes où les données sélectionnées sont dans plusieurs tables :**

Liste des élèves et nom des cours qu'ils suivent pendant plus de 3 heures.

```
SELECT NomElv, NomMat
FROM Eleves, Cours
WHERE (Eleves.RefElv = Cours.RefElv) AND (Cours.NbHeure > 3);
```

(d'abord il faut faire les jointures puis après les sélections)

➢ **Requêtes où les données proviennent d'une table mais où la condition de sélection est faite sur une table différente :**

Liste des élèves qui font de la Peinture pendant plus de 2 heures.

```
SELECT NomElv, PreElv
FROM Eleves
WHERE RefElv IN (SELECT RefElv FROM Cours WHERE (NomMat =
'Peinture') AND (NbHeure > 2));
```

IV-Les Jointures

➢ **La jointure LEFT OUTER JOIN :**

On utilise LEFT OUTER JOIN pour créer une jointure externe gauche.

➢ **La jointure RIGHT OUTER JOIN :**

On utilise RIGHT OUTER JOIN pour créer une jointure externe droite.

➢ **La jointure INNER JOIN :**

On utilise INNER JOIN pour fusionner les enregistrements de deux tables lorsque le champ commun contient des valeurs identiques.

V- Les fonctions de groupes

Les fonctions de groupes sont :

- L' AVG : La commande AVG permet de calculer la moyenne d'un champ
- Le COUNT : La commande COUNT permet de compter les lignes.
- Le SUM : La commande SUM fait la somme d'un champ.
- Min : La commande COUNT permet de trouver le minimum.
- Max : La commande COUNT permet de trouver le maximum.

Annexes

Annexe B : Le génie logiciel

I- Introduction

Le Logiciel est un élément logique du système. Par conséquent le Logiciel a des caractéristiques qui sont généralement différentes de celles du Hardware.

II- Caractéristiques du Logicie

(1) Le Logiciel est un élément logique du système par contraste à un élément physique. Il est donc différent du Hardware. Le hardware est un processus de créativité qui suit une suite d'étapes (analyse, conception, construction, test...). Ce processus sera traduit en une forme physique.

(2) Le Logiciel ne vieillit pas, mais il se détériore due au changement. A chaque changement, il est très probable d'introduire de nouvelles erreurs.

(3) La plupart des Logiciels sont construits à la demande des clients au lieu d'être assemblés à partir de composants existants.

III-Les Composantes du Logiciel

Un Logiciel est un élément du système. C'est une information qui est constituée:

(1) d'un ensemble d'instructions (programmes informatiques) qui accomplissent la fonction et les performances requises au cours de leur exécution.

(2) des structures de données qui permettent aux programmes de correctement manipuler les informations.

(3) les documents décrivant les opérations et l'utilisation des programmes.

IV-Les Applications du Logiciel

- ❖ Logiciels Système
- ❖ Logiciels Temps Réels
- ❖ Logiciels de Gestion (Systèmes d'Information)
- ❖ Logiciels Scientifiques et Techniques
- ❖ Logiciels Incorporés
- ❖ Logiciels pour Micros
- ❖ Logiciels d'Intelligence Artificielle (Systèmes Experts, Reconnaissances des Formes, Démonstration de Théorèmes, Théorie des Jeux....)

V- Le Génie Logiciel

collection de méthodes et de procédures pour l'aide à la qualité de la programmation. Les objectifs essentiels du Génie Logiciel sont:

- Une méthodologie bien définie qui tient compte du cycle Plan, Développement et Maintenance

- Un ensemble d'éléments établis qui documentent chaque étape du cycle de vie et montrent les traces d'une étape à l'autre.

- Un ensemble de jalons qui peuvent être révisés à des intervalles réguliers du cycle de vie du logiciel.

Le cycle de vie d'un logiciel est une vue à long terme du logiciel qui contient les activités qui existent avant le développement du Logiciel et après que le Logiciel soit mis en utilisation. Le logiciel est toujours une partie d'un ensemble plus large d'un Système Informatique. Donc l'analyse du système et les définitions doivent être effectuées avant le plan du Logiciel.

VI- La Crise du Logiciel

- Le Planning et les coûts estimés sont souvent erronés
- La productivité et la qualité
- Le temps pour rassembler les données
- Le client non satisfaits du système complété
- La qualité du Logiciel est souvent suspecte
- les Logiciels existants difficiles à maintenir

VII- Critères de qualité du logiciel

Il existe des « critères » de qualité qui permettent de définir différents types de qualité :

- ✓ Exactitude : Aptitude d'un logiciel à fournir des résultats voulus dans les conditions normales d'utilisation.
- ✓ Robustesse : Aptitude à bien réagir lorsque l'on s'écarte des conditions normales d'utilisation.
- ✓ Extensibilité : Facilité avec laquelle un programme pourra être adapté pour faire face à une évolution des besoins de l'utilisateur.
- ✓ Réutilisabilité : Possibilité d'utiliser certaines parties du logiciel pour développer un autre logiciel répondants à d'autres besoins. Cette notion est souvent relié à l'orienté objet où une classe générale sera facilement réutilisable.
- ✓ Portabilité : Facilité avec laquelle on peut exploiter un logiciel dans différentes implémentations. Exemple : Windows 95 ou Linux.
- ✓ Efficience : Temps d'exécution, taille mémoire...

Ces critères de qualité sont des objectifs qu'un utilisateur va spécifier éventuellement dans l'expression de ses besoins.

Un méthode de développement permet de faciliter la satisfaction des critères de qualité.

VIII- Conclusion

La connaissance des réalités du logiciel est le premier pas pour une formation pratique des solutions pour le développement du logiciel.