ÉCOLE NATIONALE VETERINAIRE D'ALFORT

Année 2006

REALISATION D'UNE APPLICATION INFORMATIQUE DE SUIVI CLINIQUE EN CLIENTELE CANINE

THÈSE

Pour le

DOCTORAT VETERINAIRE

Présentée et soutenue publiquement devant

LA FACULTE DE MEDECINE DE CRETEIL

Le 14 décembre 2006

par

Laurent GHYSELINCK

Né le 3 avril 1978 à Denain (Nord)

JURY

Président : M.
Professeur à la Faculté de Médecine de CRETEIL

Membres
Directeur : M. Jean-Marie MAILHAC
Maître de Conférences de Pathologie Chirurgicale à l'ENVA
Assesseur : M. Christophe DEGUEURCE
Professeur d'Anatomie à l'ENVA

12 septembre 2006

LISTE DES MEMBRES DU CORPS ENSEIGNANT

Directeur : M. le Professeur COTARD Jean-Pierre

Directeurs honoraires : MM. les Professeurs MORAILLON Robert, PARODI André-Laurent, PILET Charles, TOMA Bernard

Professeurs honoraires: MM. BUSSIERAS Jean, CERF Olivier, LE BARS Henri, MILHAUD Guy, ROZIER Jacques

DEPARTEMENT DES SCIENCES BIOLOGIQUES ET PHARMACEUTIQUES (DSBP)

Chef du département : M. BOULOUIS Henri-Jean, Professeur - Adjoint : M. DEGUEURCE Christophe, Professeur

-UNITE D'ANATOMIE DES ANIMAUX DOMESTIQUES
Mme CREVIER-DENOIX Nathalie, Professeur
M. DEGUEURCE Christophe, Professeur*
Mlle ROBERT Céline, Maître de conférences
M. CHATEAU Henri, Maître de conférences

-UNITE DE PATHOLOGIE GENERALE , MICROBIOLOGIE, IMMUNOLOGIE
Mme QUINTIN-COLONNA Françoise, Professeur*
M. BOULOUIS Henri-Jean, Professeur

-UNITE DE PHYSIOLOGIE ET THERAPEUTIQUE
M. BRUGERE Henri, Professeur
Mme COMBRISSON Hélène, Professeur*
M. TIRET Laurent, Maître de conférences

-UNITE DE PHARMACIE ET TOXICOLOGIE
Mme ENRIQUEZ Brigitte, Professeur *
M. TISSIER Renaud, Maître de conférences
M. PERROT Sébastien, Maître de conférences

-UNITE DE BIOCHIMIE
M. MICHAUX Jean-Michel, Maître de conférences
M. BELLIER Sylvain , Maître de conférences

- UNITE D'HISTOLOGIE , ANATOMIE PATHOLOGIQUE
M. CRESPEAU François, Professeur
M. FONTAINE Jean-Jacques, Professeur *
Mme BERNEX Florence, Maître de conférences
Mme CORDONNIER-LEFORT Nathalie, Maître de conférences

- UNITE DE VIROLOGIE
M. ELOIT Marc, Professeur *
Mme LE PODER Sophie, Maître de conférences

-DISCIPLINE : PHYSIQUE ET CHIMIE BIOLOGIQUES ET MEDICALES
M. MOUTHON Gilbert, Professeur

-UNITE DE GENETIQUE MEDICALE ET CLINIQUE
M. PANTHIER Jean-Jacques, Professeur
Melle ABITBOL Marie, Maître de conférences

-DISCIPLINE : ETHOLOGIE
M. DEPUTTE Bertrand, Professeur

-DISCIPLINE : ANGLAIS
Mme CONAN Muriel, Ingénieur Professeur agrégé certifié

DEPARTEMENT D'ELEVAGE ET DE PATHOLOGIE DES EQUIDES ET DES CARNIVORES (DEPEC)

Chef du département : M. FAYOLLE Pascal, Professeur - Adjoint : M. POUCHELON Jean-Louis , Professeur

- UNITE DE MEDECINE
M. POUCHELON Jean-Louis, Professeur*
Mme CHETBOUL Valérie, Professeur
M. BLOT Stéphane, Maître de conférences
M. ROSENBERG Charles, Maître de conférences
Mme MAUREY Christelle, Maître de conférences contractuel

- UNITE DE CLINIQUE EQUINE
M. DENOIX Jean-Marie, Professeur
M. AUDIGIE Fabrice, Maître de conférences*
Mme GIRAUDET Aude, Professeur contractuel
Mme MESPOULHES-RIVIERE Céline, Maître de conférences contractuel
M. PICCOT-CREZOLLET Cyrille, Maître de conférences contractuel

-UNITE DE REPRODUCTION ANIMALE
Mme CHASTANT-MAILLARD Sylvie, Maître de conférences* (rattachée au DPASP)
M. NUDELMANN Nicolas, Maître de conférences
M. FONTBONNE Alain, Maître de conférences
M. REMY Dominique, Maître de conférences (rattaché au DPASP)
M. DESBOIS Christophe, Maître de conférences
Melle CONSTANT Fabienne, Maître de conférences (rattachée au DPASP)
Melle LEDOUX Dorothée, Maître de conférences Contractuel (rattachée au DPASP)

- UNITE DE PATHOLOGIE CHIRURGICALE
M. FAYOLLE Pascal, Professeur *
M. MAILHAC Jean-Marie, Maître de conférences
M. MOISSONNIER Pierre, Professeur
Mme VIATEAU-DUVAL Véronique, Maître de conférences
Mlle RAVARY Bérangère, Maître de conférences (rattachée au DPASP)
M. ZILBERSTEIN Luca, Maître de conférences contractuel
M. HIDALGO Antoine, Maître de conférences contractuel

- UNITE DE RADIOLOGIE
Mme BEGON Dominique, Professeur*
Mme STAMBOULI Fouzia, Maître de conférences contractuel

-UNITE D'OPHTALMOLOGIE
M. CLERC Bernard, Professeur*
Melle CHAHORY Sabine, Maître de conférences contractuel

- UNITE DE PARASITOLOGIE ET MALADIES PARASITAIRES
M. CHERMETTE René, Professeur
M. POLACK Bruno, Maître de conférences*
M. GUILLOT Jacques, Professeur
Mme MARIGNAC Geneviève, Maître de conférences contractuel

-UNITE DE NUTRITION-ALIMENTATION
M. PARAGON Bernard, Professeur *
M. GRANDJEAN Dominique, Professeur

DEPARTEMENT DES PRODUCTIONS ANIMALES ET DE LA SANTE PUBLIQUE (DPASP)

Chef du département : M.MAILLARD Renaud, Maître de conférences - Adjoint : Mme DUFOUR Barbara, Maître de conférences

-UNITE DES MALADIES CONTAGIEUSES
M. BENET Jean-Jacques, Professeur*
Mme HADDAD/ H0ANG-XUAN Nadia, Maître de conférences
Mme DUFOUR Barbara, Maître de conférences

-UNITE D'HYGIENE ET INDUSTRIE DES ALIMENTS D'ORIGINE ANIMALE
M. BOLNOT François, Maître de conférences *
M. CARLIER Vincent, Professeur
Mme COLMIN Catherine, Maître de conférences
M. AUGUSTIN Jean-Christophe, Maître de conférences

- DISCIPLINE : BIOSTATISTIQUES
M. SANAA Moez, Maître de conférences

- UNITE DE ZOOTECHNIE, ECONOMIE RURALE
M. COURREAU Jean-François, Professeur
M. BOSSE Philippe, Professeur
Mme GRIMARD-BALLIF Bénédicte, Professeur
Mme LEROY Isabelle, Maître de conférences
M. ARNE Pascal, Maître de conférences
M. PONTER Andrew, Maître de conférences*

- UNITE DE PATHOLOGIE MEDICALE DU BETAIL ET DES ANIMAUX DE BASSE-COUR
M. MILLEMANN Yves, Maître de conférences*
Mme BRUGERE-PICOUX Jeanne, Professeur
M. MAILLARD Renaud, Maître de conférences
M. ADJOU Karim, Maître de conférences

Mme CALAGUE, Professeur d'Education Physique * Responsable de l'Unité AERC : Assistant d'Enseignement et de Recherche Contractuel

A Monsieur le Professeur de la Faculté de Médecine de Créteil, qui m'a fait l'honneur d'accepter la présidence du jury de thèse, hommage respectueux.

A Monsieur le Docteur Jean-Marie Mailhac, de l'Ecole Nationale Vétérinaire d'Alfort, qui m'a fait l'honneur de présenter et juger ce travail. Qu'il veuille bien trouver ici l'expression de ma vive gratitude et de mon profond respect.

A Monsieur le Professeur Christophe Degueurce, de l'Ecole Nationale Vétérinaire d'Alfort, qui m'a fait l'honneur d'être mon assesseur. Qu'il trouve ici le témoignage de ma respectueuse considération.

A Prudence, mon petit rayon de soleil, mon bébé, ma fille.

A Jessie, mon âme sœur, pour tous les moments qu'elle a passé à me supporter, et pour tous les moments où elle me supportera encore, qu'elle trouve ici l'expression de tout mon amour.

A ma mère, pour tous les moments d'énervement, de tendresse, de tensions, d'écoute. Qu'elle sache que je l'aimerai toujours coûte que coûte, et que je lui dois la vie que j'ai aujourd'hui.

A mon père, ce roc, cette montagne inébranlable qui sera pour toujours mon unique modèle en toutes circonstances et qui a fait de moi ce que je suis. Merci de tout cœur.

A mes grands-parents, pour leur affection bienveillante, pour m'avoir toujours soutenu, et particulièrement à la mémoire de mes grands-parents maternels. *Vous au Ciel, moi ici-bas, je ne vous oublierai pas…*

A toute ma famille.

A tous mes amis de l'Ecole et d'ailleurs, qui se reconnaîtront, car ils sont très nombreux.

REALISATION D'UNE APPLICATION INFORMATIQUE DE SUIVI CLINIQUE EN CLIENTELE CANINE

NOM et Prénom : GHYSELINCK Laurent

RESUME :

Il existe aujourd'hui de nombreux logiciels de gestion vétérinaire, et l'ENVA dispose d'un tel système baptisé CLOVIS. Cependant, les applications professionnelles sont surtout orientées vers la facturation et la comptabilité de la clinique, et proposent souvent un module de suivi clinique en inadéquation avec les attentes du praticien : facilité d'utilisation, convivialité et rapidité de saisie. L'auteur a donc voulu développer un outil informatique facilement évolutif, dénommé SYSTOLE, répondant à ce besoin.

Dans un premier temps, il décrit succinctement la théorie des bases de données, présente le Système de Gestion de Base de Données Relationnelle Microsoft Access® qui est utilisé pour le développement du projet, et détaille l'étape de conceptualisation grâce à la méthode MERISE. La deuxième partie propose, à l'aide de deux exemples concrets issus de l'application, de comprendre la programmation et les choix techniques réalisés lors du développement du logiciel. Cette partie permet d'appréhender en détail l'environnement Microsoft Access®. Enfin, la dernière partie traite de la finalisation de l'application et de son utilisation en situation de terrain. L'auteur y dresse un bilan de l'outil, de ses limites et de ses améliorations possibles. Les perspectives d'évolution de SYSTOLE n'y sont pas oubliées.

Mots-clés : environnement logiciel - informatique - base de données - Access® - suivi clinique - clientèle vétérinaire canine – vétérinaire praticien

JURY :

Président : Pr.

Directeur : Dr. MAILHAC

Assesseur : Pr. DEGUEURCE

Adresse de l'auteur :

M. GHYSELINCK Laurent

10 Rue Jules Ferry

59292 SAINT-HILAIRE LEZ CAMBRAI

DEVELOPMENT OF A CLINICAL MANAGEMENT SOFTWARE FOR THE CANINE PRACTICIONER

SURNAME : GHYSELINCK
Given Name : Laurent

SUMMARY :

Many veterinary management softwares exist nowadays and the National Veterinary School of Alfort has its own, a computer system called 'CLOVIS'. However, such commercial veterinary softwares are mostly turning towards billing and accounting ; the clinical management module they propose doesn't usually come up with practitioner expectations which are easy-to-use, conviviality and fast-keyboarding. The author thus wanted to develop 'SYSTOLE', an adaptable data-processing tool answering this lack.

At the first step, Microsoft Access®, the database system chosen for the project's development and database theory are briefly described ; moreover the MERISE method used at the concept stage is detailed. The second part helps understanding the programming and the technical choices concerning the software by giving two practical examples extracted from the application. This part is useful to grasp the Microsoft Access® database system environment. The last part deals with the finalization of the application and its use in daily veterinary practice. The author also draws up an assessment of the tool, its limits and its possible enhancements. The future improvement of 'SYSTOLE' is not forgiven to be discussed.

Key Words : software system - computer - database - Access® - clinical management - canine practice - veterinary practicionner

JURY :

President : Pr.

Director : Dr. MAILHAC

Assessor : Pr. DEGUEURCE

Author's address :

Mr GHYSELINCK Laurent

10 Rue Jules Ferry

59292 SAINT-HILAIRE LEZ CAMBRAI

FRANCE

TABLE DES MATIERES

FIGURES

TABLEAUX

INTRODUCTION

L'ENVA dispose d'un système de gestion des cas cliniques et de facturation baptisé « CLOVIS ». Les avantages de ce système au niveau médical sont indéniables, puisqu'il permet non seulement un suivi étroit du dossier des animaux, ce qui a pour intérêt de mettre directement à la disposition du consultant tous les commémoratifs du patient ; mais représente également une base de données importante permettant la réalisation d'études rétrospectives.

Cet outil trouve ses équivalents sur le marché à destination des cliniques vétérinaires libérales. Ces logiciels, très différents les uns des autres, ont néanmoins leur point commun dans la proposition d'un module de comptabilité assez complet. En effet, cet aspect reste le « nerf de la guerre » pour les vétérinaires libéraux.

Devant ce déficit en logiciels spécialisés dans le suivi clinique, ou plus justement l'inadéquation entre notre attente et les possibilités offertes, il nous a semblé opportun de réaliser une application de base de données dévolue au suivi clinique exclusif des animaux d'une clientèle canine.

La concrétisation de ce projet est désormais possible, avec un minimum de connaissances en programmation, grâce à des logiciels de SGBDR : Système de Gestion Relationnel de Base de Données. Notre choix s'est porté sur Microsoft Access®, puisque cet outil est l'un des plus répandus.

Notre objectif initial a été de créer un logiciel simple, efficace, où un maximum d'informations peut être enregistré sans que la saisie n'en soit trop consommatrice en temps. L'édition de documents types ordonnances ou certificat de vaccination nous a également semblé primordiale.

Dans un premier temps, nous présentons l'outil de SGBDR Microsoft Access, ainsi que le cahier des charges de l'élaboration du logiciel. La deuxième partie est la partie technique, qui expose deux exemples concrets de l'interface et le code qui le sous-tend. Enfin, la troisième partie fait le point sur l'ergonomie du logiciel et sa finalisation. Nous en présentons les défauts et les limites, mais aussi les moyens d'y remédier, et les évolutions possibles de l'outil.

I. PRESENTATION DU SGBDR MICROSOFT ACCESS® ET ELABORATION DU PROJET

A. DEFINITION DU SYSTEME DE GESTION DE BASE DE DONNEES RELATIONNELLE

1) Base de données : définition et histoire succincte

Une base de données, généralement notée par l'abréviation BD, est un ensemble de données structuré et organisé, généralement de grande taille, stocké sur support informatique. Ce stockage permet l'exploitation des données : ajout, suppression, modification, recherche d'information…

Une base de données correspond physiquement à des fichiers enregistrés sur un support magnétique fixe (disque dur ou autre mémoire de masse) ou amovible (disquette, clé USB…).

Il existe plusieurs types de BD [10] :

Bases de données hiérarchiques : historiquement, le premier système de base de données a été conçu pour la gestion des données du programme Apollo de la NASA. Les données étaient structurées dans des hiérarchies, comparables à l'organisation des répertoires sur un PC. Une base de données hiérarchique est une base de données dont le système de gestion lie les enregistrements dans une structure arborescente où chaque enregistrement n'a qu'un seul possesseur.
Par exemple, le *canard* appartient à la famille des *anatidés* qui elle-même appartient à l'ordre des *ansériformes* qui lui-même appartient à la classe des *oiseaux* qui elle-même appartient au sous-embranchement des *vertébrés* qui lui-même appartient au règne *animal*.

Les structures de données hiérarchiques ont été largement utilisées dans les premiers systèmes de gestion de base de données. Elles ont toutefois montré des limites pour décrire des structures complexes, répondre aux besoins réels et suivre l'évolution des systèmes d'information. Comme on le voit dans l'exemple cité plus haut, l'organisation hiérarchique des bases de données est particulièrement adpatée à la modélisation de *nomenclatures*, mais si le principe de relation « 1 vers N » n'est pas respecté (le *canard* n'appartient bien qu'à une seule *famille* mais, par exemple, un *malade* peut être en relation avec plusieurs *médecins*), alors la hiérarchie doit être

transformée en un réseau. Cette évolution nécessaire donnera naissance aux bases de données relationnelles.

Bases de données réseaux : ce modèle de base de données a été inventé par Charles.W. Bachman pour lequel il reçut en 1973 le prix Turing. Le modèle réseau est une manière de représenter les données dans le cadre d'une base de données. Ce modèle est en mesure de lever de nombreuses difficultés du modèle hiérarchique grâce à la possibilité d'établir des liaisons de type 1-N en définissant des associations entre tous les types d'enregistrements.
Ce modèle est une extension du modèle précédent (hiérarchique), les liens entre objets peuvent exister sans restriction. Pour retrouver une donnée dans une telle modélisation, il faut connaître son chemin d'accès (les liens). Ceci rend cependant les programmes encore dépendants de la structure de données.
Bases de données relationnelles : en 1970, au moment où les systèmes basés sur le modèle hiérarchique ou le modèle en réseau étaient en plein développement, CODD [2] proposa de stocker des données hétérogènes dans des tables, permettant d'établir des relations entre elles. De nos jours, ce modèle est extrêmement répandu mais en 1970, cette idée était considérée comme une curiosité intellectuelle. On doutait alors que des tables puissent être jamais gérées de manière efficace par un ordinateur. Ce scepticisme n'a cependant pas empêché CODD de poursuivre ses recherches, et IBM construira rapidement un premier prototype de Système de Gestion de Base de Données Relationnelles (SGBDR).
Depuis les années 1980, cette technologie a mûri et a été adoptée par l'industrie. En 1987, le langage SQL, une extension de l'algèbre relationnelle, est standardisé. A l'heure actuelle, les SGBDR sont présents dans de nombreux logiciels, sont très répandus dans les bases de données et représentent une industrie de plusieurs milliards de dollars.

2) Le logiciel de SGBDR

a) Définition d'un SGBDR

Les Systèmes de Gestion de Base de Données ou SGBD sont les couches logicielles qui permettent l'interaction avec les informations enregistrées dans la BD [12].
Lorsque le SGBD travaille avec une BD de type relationnel, on utilise le terme de Système de Gestion de Base de Données Relationnelle ou SGBDR.
Le SGBDR assure plusieurs fonctions :

- **Ajout de données** : un SGBDR doit permettre l'ajout de données. Pour cela, il est tout d'abord nécessaire de pouvoir décrire les données avec un langage de description de données (LDD). Une fois les données décrites, on peut ajouter des valeurs qui correspondent à la description qu'on en a faite par le biais d'un langage de manipulation de données (LMD).

- **Modification de données** : Les données doivent être modifiables. On doit pouvoir changer la définition des données et les valeurs des données grâce au LDD et LMD respectivement.

- **Recherche de données** : La recherche des données est un point crucial. Il faut que le SGBDR puisse restituer les données rapidement.

b) Langage de description et de manipulation des données

Dans la plupart des SGBDR, le LDD et le LMD sont identiques et basé sur un langage standardisé et normalisé : le Structured Query Language (SQL).

Le SQL est intimement lié au développement du système de BD relationnelle [11].

Lorsque CODD rédigea l'article "Un modèle de données relationnel pour de grandes banques de données partagées" dans la revue Communications of the ACM (Association for Computing Machinery) [2], il ne se doutait pas que son modèle serait rapidement admis comme modèle définitif pour les *systèmes de gestion de base de données* (SGBD). IBM développa alors concomitamment le **Structured English Query Language ("SEQUEL")** (langage d'interrogation structuré en anglais) pour mettre en œuvre le modèle de CODD.

En 1979, Relational Software, Inc. (actuellement Oracle Corporation, détentrice du SGBDR Oracle®) présenta la première version commercialement disponible de SQL, rapidement imité par d'autres fournisseurs.

SQL a été adopté comme recommandation par l'institut de normalisation américaine (ANSI) en 1986, puis comme norme internationale par l'ISO en 1987 sous le nom de *ISO/CEI 9075 - Technologies de l'information - Langages de base de données – SQL* [11].

La norme internationale SQL est passée par un certain nombre de révisions, reprises dans le tableau 1.

Tableau 1 : Révisions de la norme internationale du SQL, d'après [11]

Année	Nom	Appellation	Commentaires
1986	ISO/CEI 9075:1986	SQL-86 ou SQL-87	Édité par l'ANSI puis adopté par l'ISO en 1987.
1989	ISO/CEI 9075:1989	SQL-89 ou SQL-1	Révision mineure.
1992	ISO/CEI 9075:1992	SQL-92 ou SQL2	Révision majeure.
1999	ISO/CEI 9075:1999	SQL-99 ou SQL3	Expressions régulières, requêtes récursives, déclencheurs, types non-scalaires et quelques fonctions orientées objet. (les deux derniers points sont quelque peu controversés et pas encore largement implémentés.)
2003	ISO/CEI 9075:2003	SQL:2003	Introduction de fonctions pour la manipulation XML,"window functions", ordres standardisés et colonnes avec valeurs auto-produites (incluant colonnes d'identité).

Le SQL se décompose en 5 parties :

- Ordres DDL (*Data Definition Language*) : permettant de modifier la structure de la base de données
- Ordres DML (*Data Manipulation Language*) : permettant de consulter / modifier le contenu de la base de donnée
- Ordres DCL (*Data Control Language*) : permettant de gérer les privilèges, c'est-à-dire les utilisateurs et les actions qu'ils peuvent entreprendre

- Ordres TCL (*Transaction Control Language*) : permettant de gérer les transactions, c'est-à-dire rendre atomique divers ordres enchaînés en séquence

- SQL procedural : PSM (*Persistent Stored Module*), CLI (*Call Level Interface*), *Embedded SQL*, ... Qui est un ensemble d'outils pour que SQL s'interface avec des langages hôtes.

Le SQL est un langage devenu au fil de ses révisions très étoffé et complexe. Toute l'intelligence des SGBDR actuels est de proposer une méthode de création et d'interrogation de base de données simple, orientée objet et qui convertit automatiquement les besoins du développeur en ordres SQL. À titre d'illustration, nous ¨proposons en Annexe 1 des exemples de langage SQL.

c) Principe de fonctionnement d'un SGBDR

La BD relationnelle permet de stocker et d'organiser une grande quantité d'information. Les SGBDR permettent de naviguer dans ces données et d'extraire (ou de mettre à jour) les informations voulues au moyen d'une requête.

Les données *apparaissent* comme stockées dans des **tables** qu'on nomme également **relations**.

Les En-têtes de ces tables représentent les données que la BD peut stocker. Ce modèle relationnel conduit à :

- une grande *simplicité d'usage*
- une *transparence pour l'utilisateur* de toute réorganisation technique de la base
- une facilité de *combinaison* du contenu de plusieurs tables (opération *join* ou *jointure*).

Les relations possèdent un certain nombre d'attributs permettant de décrire un enregistrement. La non-duplication (absence de redondance) des enregistrements est assurée par le SGBDR.

Dans les relations, il est possible de définir deux types de clés :

- une clé primaire permet d'identifier un et un seul enregistrement (par exemple le numéro de sécurité sociale). Deux enregistrements ne peuvent donc avoir la même clé primaire.

- une clé étrangère, attribut d'une relation qui est une clé primaire dans une autre table. Elle permet donc de lier deux relations entre elles.

Pour relier les données, les jointures sont utilisées, dont il existe différents types : un-à-un, un-à-plusieurs, …

Lors d'une recherche de données, les opérations sont communiquées sous forme de requêtes aux SGBDR. Une requête se présente sous la forme d'une table, que l'opérateur remplit en indiquant pour quel champ il désire réaliser l'interrogation, et avec quel(s) critère(s). Comme nous l'avons dit plus haut, le SGBDR s'occupe de traduire cette demande en langage SQL.

Dans une base de données relationnelle, le but est de séparer les informations au maximum pour éviter les doublons et la redondance, et d'empêcher la perte de qualité d'information (par exemple, l'adresse d'un fournisseur n'est mise à jour qu'une et une seule fois : la modification sera alors prise en compte sur l'ensemble des courriers).

3) Possibilités offertes par le SGBDR Microsoft Access®

Lors du développement de notre projet, la question s'est posée du choix du SGBDR. Il nous a semblé opportun d'opter pour un logiciel largement diffusé, possédant une notice technique importante ainsi qu'une grande communauté de développeurs, afin de pouvoir en tirer aide et conseils durant le cheminement de l'élaboration du projet.

Microsoft Access® répond parfaitement à ces besoins. En effet, il possède derrière lui de nombreuses années de développement et d'améliorations, et c'est un SGBDR profondément ancré dans le milieu de l'entreprise [1]. De nombreux sites web d'entraide entre développeurs existent et ont permis l'amélioration rapide de notre application. [6, 13]

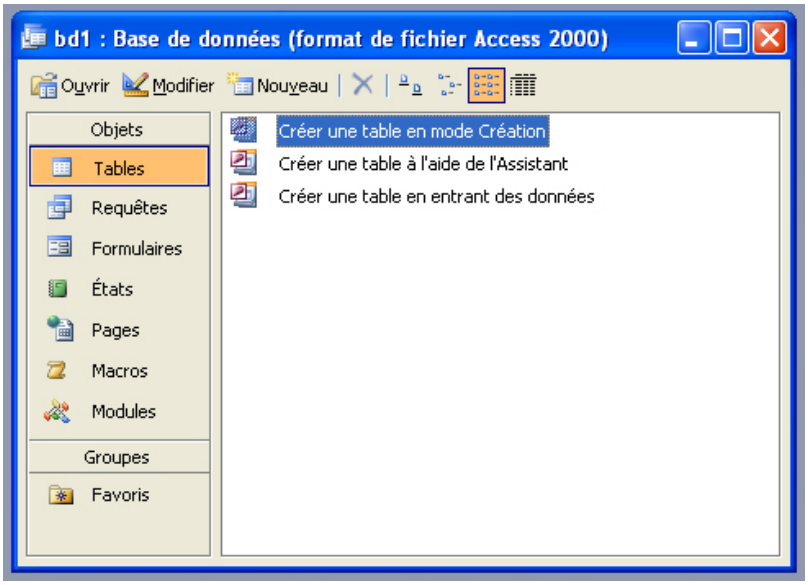
a) Manipulation des données

Comme dit plus haut, les SGBDR servent d'interface entre l'utilisateur et la BD relationnelle, manipulable grâce à un langage d'accès, généralement le SQL.

Microsoft Access® ne déroge pas à la règle puisqu'une interface graphique nous permet de créer facilement les tables dont nous avons besoin pour stocker les données, et les requêtes qui nous servent à interroger ces tables.

La **création d'une table** peut s'effectuer de deux manières différentes : l'opérateur choisit soit de décrire chaque champ de la table, sa taille, le type de données qu'il va contenir… : c'est le mode « Création » ; soit il choisit de partir avec une grille vide, et de rentrer les données au fur et à mesure (figure 1).

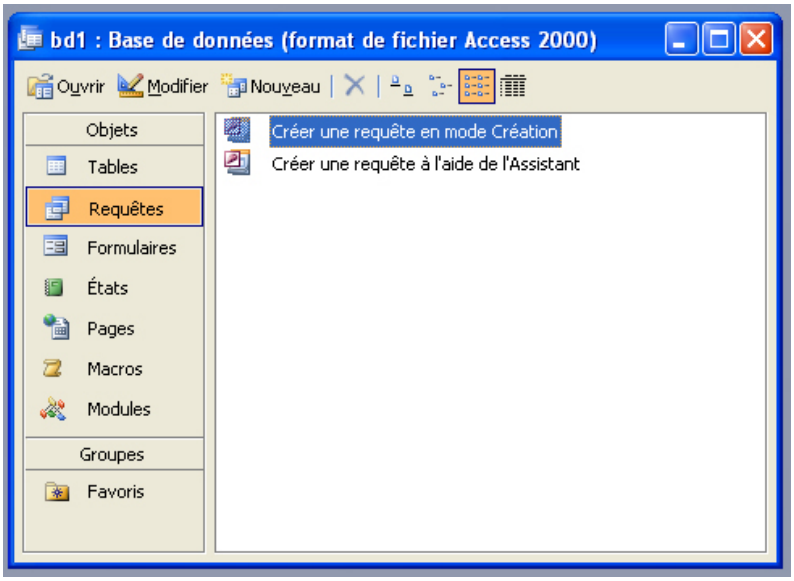
Figure 1 : Copie d'écran de la fenêtre des Tables



L'interrogation des données se fait à partir de **requêtes** : ce sont des grilles que l'opérateur remplit en indiquant quels champs de quelles tables il désire interroger, ainsi que les critères qui doivent être pris en compte (figure 2). Microsoft Access® traduit alors les données entrées dans ces requêtes en SQL pour interroger la table et afficher le résultat de la requête.

L'organisation relationnelle des données implique qu'une requête puisse interroger plusieurs tables qui sont liées entre elles.

Figure 2 : Copie d'écran de la fenêtre des Requêtes



Note : Microsoft Access® inclut des « Assistants » qui permettent de rapidement mettre en œuvre une application de BD sans rien connaître de leur théorie : l'assistant récolte simplement les informations indispensables à la création des tables ou des requêtes et automatise leur création.

b) Possibilité d'interfaçage

Si les fonctions du SGBDR s'arrêtaient là, nous n'aurions à notre disposition que de grands tableaux de données remplis d'informations.

Grâce à son interface graphique, Microsoft Access® permet également d'améliorer la présentation des données, qu'elles soient issues de tables ou de requêtes.

Ces possibilités s'offrent à l'utilisateur sous trois aspects :

- les **formulaires** sont des outils graphiques programmables qui permettent d'améliorer la présentation des données pour la saisie ou l'affichage **à l'écran.** Même s'il est possible d'imprimer un formulaire, son rôle est dévolu à l'interfaçage visuel. Ici aussi, l'opérateur opte pour un développement en mode « Création » où le formulaire est construit de toutes pièces ; ou pour un mode « Assistant », où Microsoft Access® s'occupe pour l'utilisateur de créer le formulaire relatif à une table ou à une requête, en fonction de choix de présentations standards disponibles.

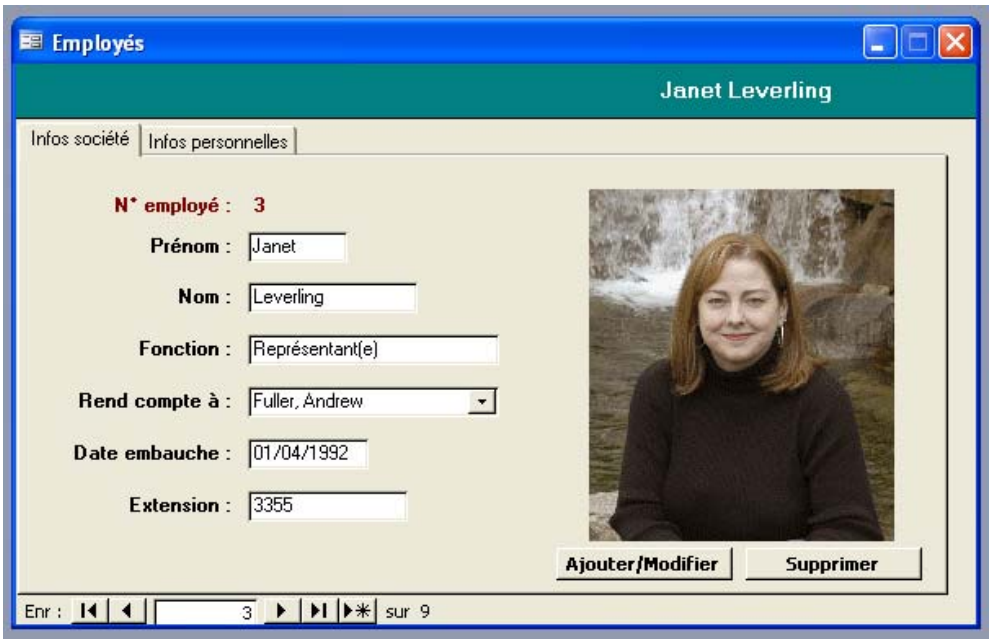
- les **états** ont la même fonction que les formulaires, mais sont utilisés pour réaliser des documents mis en pages en vue d'une impression papier.

- enfin, **les pages d'accès aux données** permettent de mettre au point une interface HTML, utilisable en réseau pour l'accès à une base de données Microsoft Access® via un navigateur internet.

Ces trois objets sont modifiables et personnalisables à volonté. Microsoft Access® est un environnement de développement « orienté objet » : tous les éléments nécessaires à la création de tels formulaires ou états sont pré-programmés et modifiables grâce à des boîtes de dialogue. Nul besoin d'être un programmeur émérite pour réaliser rapidement des effets de présentation graphique à l'allure professionnelle.

La figure 3 montre un exemple de formulaire extrait de la base de données « Les Comptoirs », une base exemple fournie avec Microsoft Access®.

Figure 3 : Copie d'écran du formulaire « Employés » de la base de données exemple « Les Comptoirs »



c) Fonctions avancées

Microsoft Access® possède outre ces fonctions de base de nombreuses autres fonctions avancées, qu'il serait impossible et inutile de toutes décrire ici.

Cependant, il convient d'en évoquer deux dont l'utilisation peut se révéler intéressante pour le développement de notre application.

La programmation grâce au Visual Basic for Applications (VBA) : Microsoft® a développé de nombreuses applications logicielles. Parmi elles, on peut souligner l'importance du Visual Basic® langage de programmation également « orienté objet ». Ainsi, Visual Basic® est capable d'exploiter des bases de données Access® pour créer des applications puissantes. Mais la prise en main d'un tel langage est difficile et longue. Aussi, Access® inclut le VBA, version simplifiée et fournie avec les logiciels de la gamme Office® [1]. Cela apporte principalement deux intérêts au développement :

- la possibilité de réaliser l'automatisation de certaines tâches, ainsi que d'élaborer des fonctions complexes.

- par son partage avec les autres logiciels de la gamme, il est possible d'utiliser VBA pour interfaçer et échanger des données entre Access®, Word®, Excel®, Outlook®, … Cette possibilité de pilotage des autres logiciels est appelée **automation.**

On voit immédiatement l'intérêt de cette fonction dans la réalisation de mailing par courrier papier ou électronique en utilisant les coordonnées clients contenues dans la base de données.

Le travail en réseau : Microsoft Access® offre la possibilité de séparer une base de données en plusieurs « fragments » (c'est-à-dire en plusieurs fichiers). Ainsi, des tables externes appartenant à d'autres bases de données peuvent être liées à la base de données en cours. [7]

On définit donc deux segments de l'application :

- le « back-end », contient toutes les tables et les requêtes. Celles-ci sont liées à une autre base de données :

- le « front-end », qui est vide de tables et de requêtes, mais héberge tous les formulaires et états. Cette partie est l'interface utilisateur de l'application de base de données.

Cette structure de type « client/serveur », permet le travail multi-utilisateur : il suffit de placer le fichier de données sur un ordinateur du réseau (le « serveur »), et de distribuer l'application

front-end à tous les utilisateurs (les « clients »). Ceux-ci se connectent ensuite au fichier de données pour l'exploiter.

Bien sûr, Microsoft Access® possède des sécurités pour éviter que deux utilisateurs différents ne modifient le même enregistrement au même moment.

Des fonctions plus avancées permettent également de gérer des groupes d'utilisateurs avec mots de passe, pour lesquels on peut définir des restrictions d'accès à la base de données (par exemple, les Auxiliaires Vétérinaires ne peuvent pas accéder à la table « Consultations » de la base de données).

Nous reverrons plus en détail ces possibilités lors de l'évocation des tests « grandeur nature » de l'application.

B. CAHIER DES CHARGES DU PROJET

1) Constatation de départ et ciblage des destinataires

L'idée de créer une application de suivi clinique en clientèle canine est partie du constat que généralement, les logiciels disponibles sur le marché ne répondaient pas aux critères fixés pour l'exploitation que nous désirions en faire. Soit leur orientation est trop comptable, soit le système d'interface et de navigation est trop complexe ou trop brouillon, ou pas assez synthétique, soit on ne retrouve pas au niveau médical de système de gestion de cas assez simple, mais tout de même suffisamment élaboré pour y rentrer un grand nombre d'informations sans que la présentation n'en devienne abscons.

L'avantage est également que ce genre d'application réalisée par l'utilisateur principal peut s'adapter en temps réel à l'évolution des besoins ou aux goûts du moment. Cette puissance d'évolution est un atout dans un milieu où la concurrence et les services au client progressent chaque jour.

Cette application s'adresse donc dans un premier temps à notre propre structure vétérinaire. Mais elle est assez simple d'utilisation et polyvalente pour s'adapter à n'importe quelle clinique vétérinaire d'orientation canine.

Les utilisateurs principaux en sont les vétérinaires, mais les activités administratives (impression des Cerfa Rage) ou hygiéniques (prise de poids et notation dans le dossier lors de passage à la clinique) peuvent être remplies par les Auxiliaires vétérinaires.

2) Définition de l'application à réaliser

L'application à réaliser est une application de base de données assurant l'enregistrement et la manipulation :

- des clients
- des animaux qui leurs sont rattachés
- des vaccinations et des consultations appliquées à ces animaux

Cette application est nommée SYSTOLE en rapport avec la phase de travail du muscle cardiaque, puisque SYSTOLE a pour but d'améliorer la qualité du travail au sein de la clinique.

3) Les fonctions essentielles que devra remplir l'application

SYSTOLE doit remplir d'office certaines fonctions indispensables à la gestion de la clientèle et des consultations. En effet, n'oublions pas que son but premier est de faire gagner du temps tout en permettant la constitution d'un véritable « dossier médical » relatif à chaque animal.

Avant toute chose, il nous faut constater que la saisie des éléments de la consultation prend du temps. Afin de ne pas étendre inutilement la consultation, il nous a semblé primordial de réaliser l'automatisation de certaines tâches. Voici donc le cahier des charges auquel l'application devra répondre :

- Interface agréable : c'est le minimum pour utiliser l'application dans des conditions optimales.
- Visualisation des clients : complète et synthétique
- Visualisation des animaux d'un client : disposer d'une liste des animaux correspondant au client sur son écran, ainsi qu'une liste des animaux décédés.
- Visualisation des animaux : sur la fiche animal, disposer de toutes les informations concernant l'animal, et de l'historique de ses vaccinations et de ses consultations.
- Pour chaque prestation réalisée, pouvoir saisir l'intervenant concerné (environnement multi-utilisateur)
- Disposer d'un module d'enregistrement et de visualisation des pesées relatives à un animal.
- Disposer d'une possibilité de visualisation des achats d'aliments liés à un animal.
- Vaccinations : pouvoir saisir toutes les vaccinations de l'animal et les visualiser d'un regard.
- Vaccinations : pouvoir réaliser l'édition imprimante des Cerfas Rage.
- Vaccinations : pouvoir créer un rappel courrier ou le supprimer sur une vaccination en particulier.
- Consultations : pouvoir saisir toutes les consultations de l'animal et les visualiser d'un regard.
- Consultations : disposer d'un espace de saisie large
- Consultations : avoir la possibilité de saisir les médicaments prescrits, avec leur posologie, et d'effectuer l'impression automatique de l'ordonnance à partir de ces données.

- Consultations : avoir la possibilité d'enregistrer des images liées à une consultation en particulier.
- Images : pouvoir visualiser sur l'écran de l'animal toutes les images enregistrées provenant des consultations de cet animal.
- Images : pouvoir attribuer des mots-clés aux images afin d'effectuer une recherche dans cette banque d'information.

Note :

SYSTOLE est actuellement dans sa version 1.3.

Jusqu'à la version 1.2, il incluait également la possibilité de saisir des chirurgies et des examens complémentaires dans une consultation.

Dans un souci de gain de temps de saisie, les confrères utilisateurs ont préféré supprimer l'écran « Chirurgie » afin de porter directement les informations qui leur semblaient importantes dans la zone de saisie globale de la consultation, voire de créer une nouvelle consultation dédiée uniquement à la chirurgie.

Depuis la version 1.21, SYSTOLE inclut aussi la gestion des documents multimédias. De ce fait, avec la possibilité de scanner des documents, le volet « Examens complémentaires » est devenu obsolète, puisqu'un compte-rendu d'examen peut être numérisé et directement versé au dossier de la consultation correspondante.

C. CONCEPTUALISATION DE L'APPLICATION

A présent que nous savons utiliser le SGBDR, que nous avons un cahier des charges précis de ce que nous souhaitons obtenir, il ne reste plus qu'à effectuer les choix techniques à la base de son développement, et à construire l'application.
Cependant, se lancer tête baissée ne peut aboutir qu'à générer des problèmes ; notamment, il est impossible de savoir quelles tables seront construites, avec leurs jointures, et dans quelles tables placer tel ou tel champ de données.
Il faut donc passer par une indispensable étape de conceptualisation.

1) La méthode MERISE

Merise est un acronyme signifiant ***Méthode d'Étude et de Réalisation Informatique par les Sous-Ensembles*** ou ***pour les Systèmes d'Entreprises***.

La méthode MERISE est issue d'une initiative de la Mission informatique du Ministère de l'Industrie qui, de 1977 à 1980, réunit dans un groupe de projet des chercheurs et des praticiens des Systèmes d'Information Informatisés (SII) pour élaborer une méthode unifiée pour la conception de systèmes d'information, avec pour premier objectif de mettre cette méthode en œuvre dans les projets de l'Administration et d'inciter les grandes entreprises à y adhérer. Ce groupe :

- reprend les travaux de recherche de l'équipe d'Hubert Tardieu développés depuis 1974 et qui proposaient un ensemble de modélisations pour la conception d'un système d'information ainsi que des prototypes d'outils,
- élabore une démarche de mise en œuvre issue de la pratique des SII (Etude préalable, Etude détaillée, Etude Technique, Réalisation, …)
- propose un cadre d'organisation et de conduite de projet (maîtrise d'ouvrage / maîtrise d'œuvre, principaux rôles)
- définit les fonctions des outils associés (interface graphique, référentiel, règles et transformations, …)

Les travaux se concrétisent de 1979 à 1981 par des publications de fascicules du Ministère. L'année 1983 marque la diffusion publique de la méthode MERISE ainsi que le début de son

expansion, au travers de la parution du livre « La méthode MERISE, Tome I : Principes et Outils » par TARDIEU et al [8].

La méthode MERISE d'analyse et de conception propose une démarche articulée sur 3 niveaux de préoccupation. Cet étagement en 3 niveaux a pour objectif de *hiérarchiser* les questions auxquelles répondre et de ne pas biaiser les réponses apportées au niveau conceptuel par des choix de technique ou d'organisation (ou de ré-organisation) [4, 8]. Nous allons étudier ces trois niveaux successivement.

a) Le niveau conceptuel

Pour simplifier l'abord de ce niveau, il est possible de le résumer à l'obtention d'un **Modèle Conceptuel des Données** (ou MCD), schéma représentant la structure du système d'information du point de vue des données, c'est-à-dire les dépendances ou relations entre les différentes données du système d'information.

En théorie, à cette étape est également constitué le **Modèle Conceptuel des Traitements** (ou MCT), schéma représentant les traitements, en réponse aux événements à traiter (par exemple la commande d'un client dans une base de données de commande et de facturation). Pour SYSTOLE, nous nous intéresserons uniquement au développement du MCD.

Le MCD repose sur les notions d'entité et de relation :

L'entité : l'entité est définie comme un objet de gestion considéré d'intérêt pour représenter l'activité à modéliser (ex: entité « pays »), chaque entité est porteuse d'une ou plusieurs propriétés simples, dites « atomiques » (ex: code, nom, capitale, population, superficie) dont l'une, unique et discriminante, est désignée comme identifiant (ex: code).

L'entité se décline en occurrences (ex: (fr, France, Paris, 60,4 millions hab., 550 000 km²), (de, Allemagne, Berlin, 82 537 000 hab., 357 027 km²)). Par construction, le MCD impose que toutes les propriétés d'une entité ont vocation à être renseignées (pas de propriété « facultative »).

Le MCD doit, de préférence, ne contenir que le coeur des informations strictement nécessaires pour réaliser les traitements conceptuels : les informations calculées (ex: montant TTC d'une

facture) ou déductibles (ex: densité démographique (population / superficie) ne doivent pas y figurer.

La relation : la relation est un lien sémantique entre une ou plusieurs entités. Elle peut être réflexive, de préférence binaire (ex : une usine « est implantée » dans un pays), parfois ternaire, voire de dimension supérieure. Elle peut également être porteuse d'une ou plusieurs propriétés (ex : « date d'implantation » d'une usine dans un pays)

Cette description sémantique est enrichie par la notion de cardinalité. Celle-ci indique le nombre minimum (0 ou 1) et maximum (1 ou n) de fois ou une occurrence quelconque d'une entité peut participer à une relation (ex: une usine est implantée dans un (cardinal min.) et un seul (cardinal max.) pays ; et réciproquement un pays peut faire l'objet soit d'aucune (cardinal min.) implantation d'usine soit de plusieurs (cardinal max.).

b) Le niveau organisationnel

A ce niveau de préoccupation, les modèles conceptuels sont précisés et font l'objet de choix organisationnels. Un **Modèle Logique des Données** (ou MLD) est construit qui reprend le contenu du MCD précédent, et précise la structure et l'organisation des données telle qu'elles pourront être implémentées. Par exemple, à ce stade, il est possible de connaître la liste exhaustive des tables qui seront à créer dans une base de données relationnelle. Le MLD est donc la traduction exacte de l’ensemble des tables qui vont exister dans la base de données, avec l’ensemble de leurs champs, notamment ceux qui seront impliqués dans des relations.

De la même façon, plus en détail, un **Modèle Logique des Traitements** (ou MLT) peut être construit et fait écho au MCT. Le MLT précise les acteurs et les moyens qui seront mis en œuvre.

La transcription d'un MCD en MLD s'effectue selon quelques règles simples qui consistent d'abord à transformer toute entité en table, avec l'identifiant comme clé primaire, puis à observer les valeurs prises par les cardinalités maximum de chaque association pour représenter celle-ci par l'ajout d'une clé étrangère dans une table existante [3].

c) Le niveau physique

Le niveau physique correspond à l'implémentation du MLD dans le SGBD retenu pour le développement de l'application. Il ne fait donc que traduire le MLD pour qu'il puisse être réalisé avec les outils proposés par le logiciel choisi pour le développement du projet.

C'est à ce niveau de préoccupation que peuvent être créées les tables qui serviront à recevoir les données. A titre d'exemple, nous fournissons dans la figure 4 une copie de l'écran de création de la table « Clients », réalisée selon les choix opérés à l'étape conceptuelle.

2) Ecriture du MCD et du MLD

Le MCD est la transcription schématique des entités et de leurs propriétés, ainsi que des relations qui les unissent.

Dans un premier temps, on spécifie les entités. On écrit ensuite l'ensemble des propriétés qui la caractérisent. Enfin, on choisit un identifiant : soit il fait parti des propriétés déjà identifiées, soit on crée une nouvelle propriété. Il faut préférer les identifiants numériques dans les bases de données car ceux-ci facilitent et accélèrent les recherches. Une fois les entités créées, on écrit les relations qui existent entre elles, avec les cardinalités.

Dans le MCD, la propriété servant d'identifiant est soulignée. La figure 5 présente le MCD final de la base de données de SYSTOLE.

Le MLD est déduit directement à partir du MCD. Le MLD est l'écriture des tables comme elles vont apparaître dans la base de données, avec l'ensemble des champs qu'elles comporteront, en particulier les champs permettant de les relier. Chaque classe d'entité du modèle conceptuel devient une table dans le modèle logique. Les identifiants de la classe d'entité sont appelés clés primaires, tandis que les propriétés deviennent les champs de la table. Les liaisons entre les tables sont réalisées par l'insertion de clés étrangères : c'est une nouvelle propriété de l'entité, qui correspond à la clé primaire de la table liée. Les cardinalités permettent de définir quelles sont les tables qui hériteront de clés étrangères.

Le MLD final est présenté en figure 6. Les clés primaires sont soulignées, les clés étrangères sont en italique.

Figure 4 : Copie de l'écran de création de la table des Clients

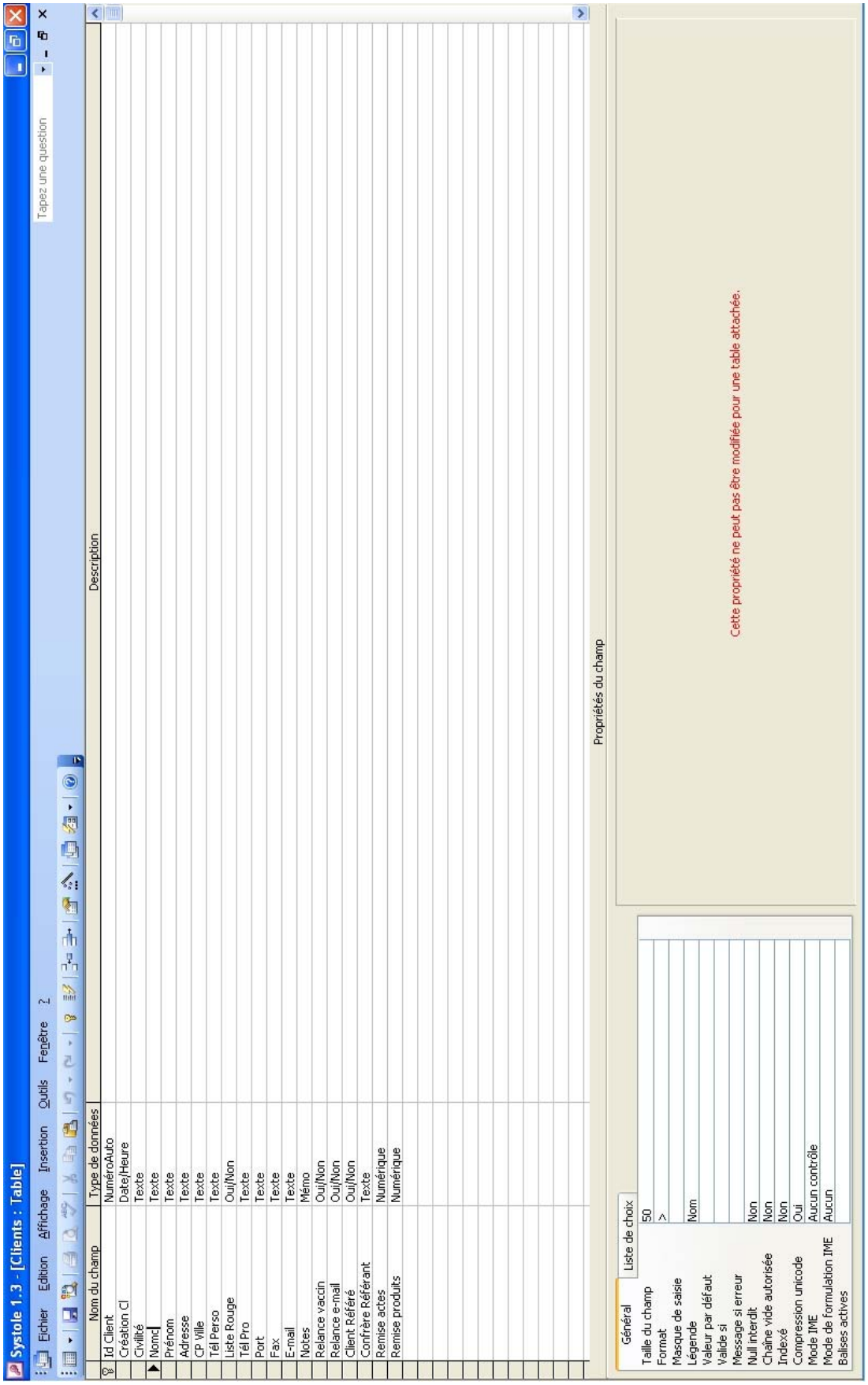


Figure 5 : MCD final

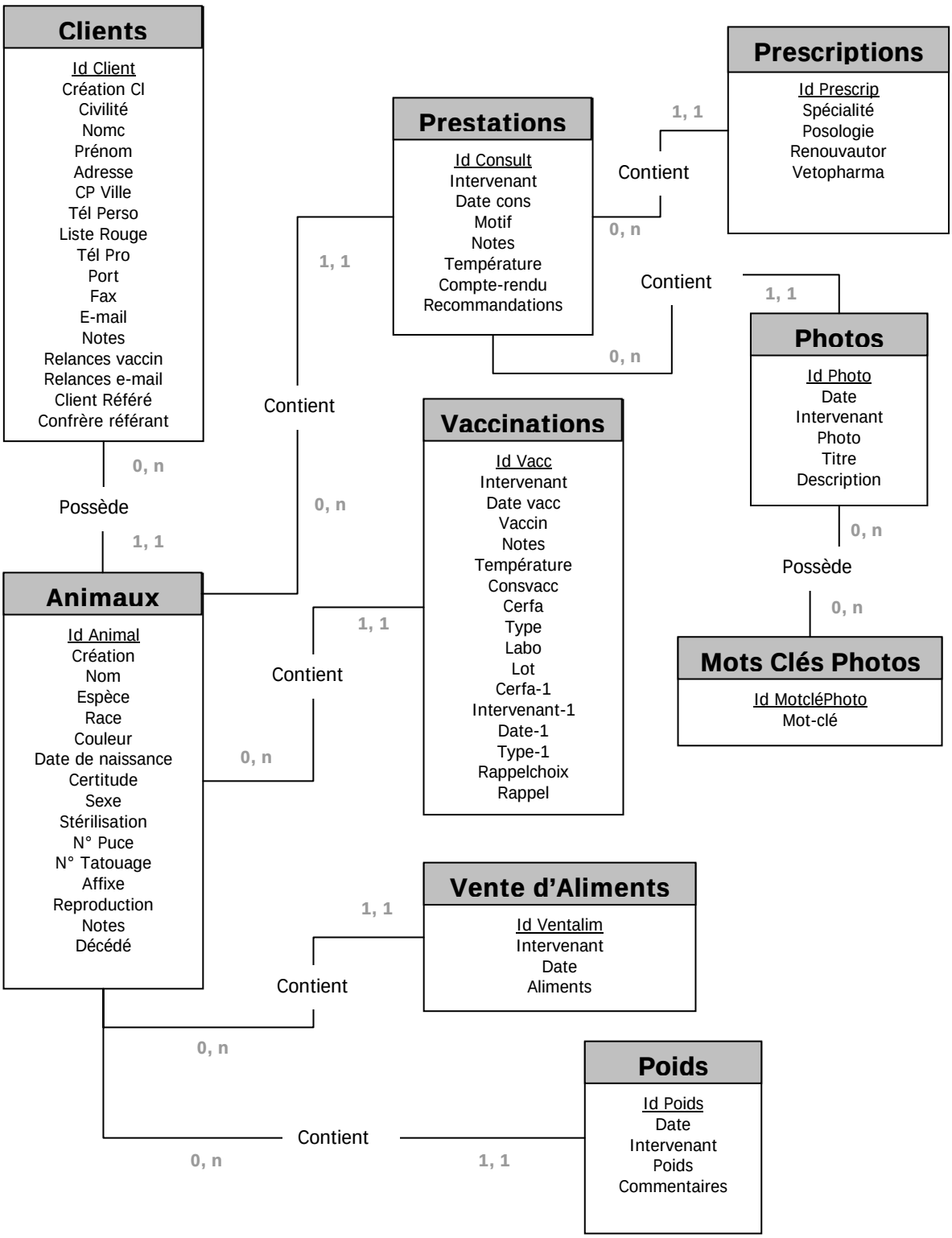
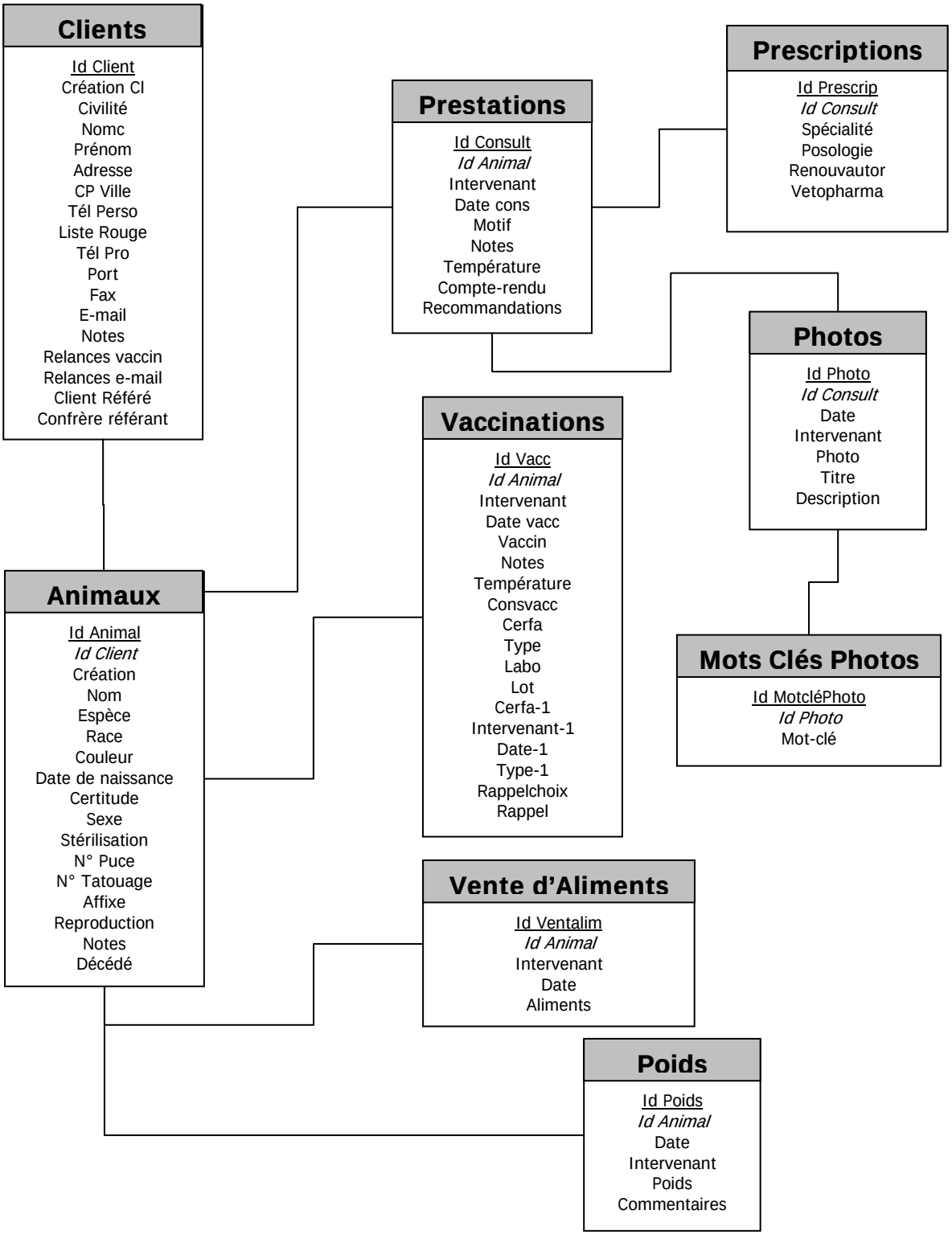


Figure 6 : MLD final



3) *Choix techniques*

a) Création des éléments graphiques de l'interface

La réalisation de l'interface graphique est l'un des points nécessitant le plus de temps dans l'élaboration d'une application informatique. Afin que l'utilisation soit facile et agréable, nous avons décidé d'utiliser dans SYSTOLE des icônes qui, lorsqu'elles sont cliquées, permettent d'accéder aux différents formulaires. Ces éléments visuels rendent l'interface plus intuitive. De plus, une info-bulle s'affiche lorsque le curseur de la souris reste plus de deux secondes sur l'icône, donnant ainsi une information complémentaire sur sa fonction (voir figure 7).

Figure 7 : Copie de l'écran l'affichage d'une info-bulle

Le texte affiché dans l'info-bulle est paramétrable par l'accès aux options du cadre d'image contenant l'icône.

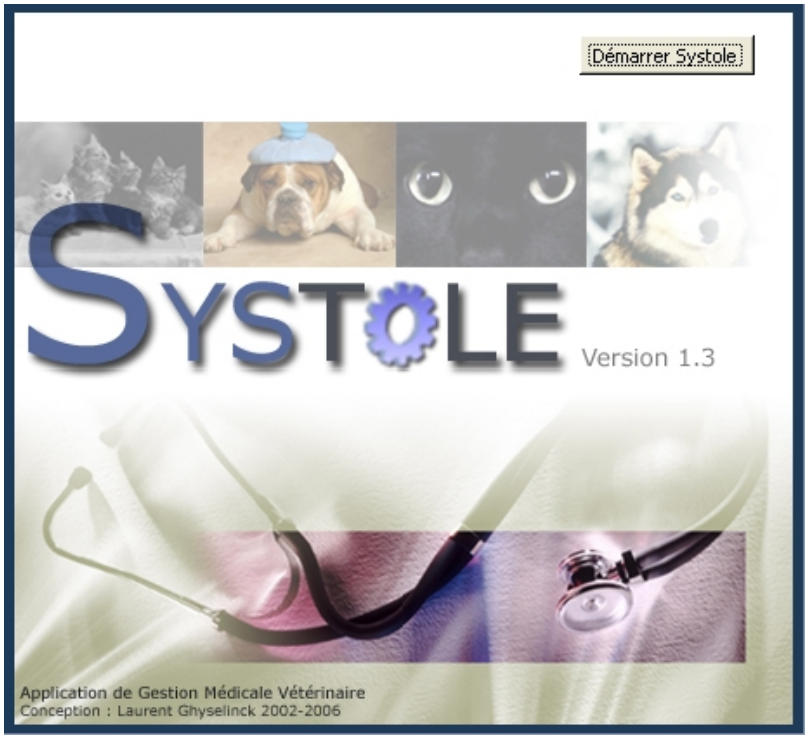
Pour créer les éléments graphiques de l'application, nous avons choisi d'utiliser un logiciel gratuit disponible sur Internet : GIMP [5]. Ce logiciel permet de réaliser des opérations de retouche d'image et de créer des logos ou des icônes à l'allure professionnelle (voir figure 8).

Figure 8 : Copie de l'écran d'accueil de « THE GIMP »

L’utilisation de GIMP a également permis de réaliser un « splash screen » pour SYSTOLE. Il s’agit de l’écran d’accueil de l’application, contenant des informations sur le nom, la version du logiciel ainsi que sur son créateur. C’est un écran d’attente qui s’affiche également pendant ou après que certaines fonctions nécessaires au logiciel soit chargées et mises en place.

La figure 9 montre une copie de l’écran d’accueil de SYSTOLE.

Figure 9 : Copie du spash screen de SYSTOLE



b) Constitution d’un back-end et d’un front-end

Le principe d’une telle application est de **séparer les données (les tables) des autres objets** (les formulaires, les requêtes, les macros, les modules) [7]. L’application contenant les tables (le back-end encore appelé application d’arrière-plan) est stockée sur un dossier accessible en lecture, écriture et modification par l’ensemble des utilisateurs. L’application contenant les autres objets (le front-end encore appelé application frontale ou d’avant-plan) contient les éléments de l’interface utilisateur et se trouve stockée localement sur chaque poste ou dans le dossier personnel de chaque utilisateur. Les avantages d’une telle solution sont nombreux :

- chaque utilisateur peut disposer de **son propre environnement**,

- chaque utilisateur n'a accès qu'aux **seuls objets de base de données dont il a besoin**, ce qui est moins perturbant que de voir une quinzaine de formulaire, une trentaine de requête...,

- les utilisateurs éclairés peuvent **réaliser leurs propres outils** (formulaires, requêtes,…) et être moins dépendant de l'administrateur de bases de données,

- les utilisateurs moins éclairés pourront laisser faire l'administrateur de base de données (ou un développeur d'applications) qui continuera à produire les éléments de l'interface,

- lors d'utilisation en réseau, les données sont centralisées sur le serveur.

Pour SYSTOLE, nous décidons, en prévision de l'utilisation en réseau, et pour faciliter la sauvegarde des données, de créer un fichier « sysdata.mdb », contenant exclusivement les tables de l'application : il s'agit de notre back-end. Pour l'instant, un seul front-end est créé, qui contient les formulaires, les états et certaines requêtes. Il porte le nom « Systole FE Med 1.3.mdb ». C'est ce fichier qui sera placé sur les postes clients.

c) Stockage et utilisation dans l'application des données liées au système

Nous venons de créer les tables qui vont contenir les données. Nous les avons placées dans le back-end. Nous avons créé le front-end et pris les décisions techniques pour la réalisation de l'interface graphique.

Mais pour fonctionner correctement, l'application a besoin de stocker dans la base de données un certain nombre d'informations qui n'apparaissent pas dans le MCD. Ces informations, appelées « données systèmes » sont stockées également dans des tables nommées « tables systèmes ». Ces données sont par exemple : le chemin du fichier « sysdata.mdb », celui où la sauvegarde devra être réalisée, les données concernant les utilisateurs…

A titre d'exemple, la figure 10 montre une copie de la table « Valsys relances vacc » qui sauvegarde l'intervenant responsable de l'édition des relances vaccinales ainsi que la période sur laquelle les relances ont été éditées. Au contraire des autres tables, celle-ci ne contient qu'un seul enregistrement. Il n'y a pas de sauvegarde de l'historique de l'envoi des relances, la table sert

uniquement à stocker les valeurs de l'action en cours (ou de la dernière action dans le cadre d'une nouvelle édition).

Figure 10 : Copie de l'écran de table « Valsys Relances Vacc »

De plus, un certain nombre de tables contiennent les données permettant de créer des listes de choix. Ces listes de choix offrent la possibilité à l'utilisateur d'accélérer la saisie en :

- proposant de choisir directement entre plusieurs occurrences
- proposant d'emblée un éléments de la liste à partir des premières lettres saisies : cette propriété se nomme « auto-étendre ». Il est possible ou non de l'activer dans les propriétés de la liste de choix. De la même manière, on peut forcer l'utilisateur à choisir obligatoirement un élément de la liste, ou lui permettre de saisir un élément personnalisé.

Pour chaque champ proposant une liste de choix multiple, il a été décidé de créer une table permettant de sauvegarder l'ensemble de ces possibilités. Très généralement, ces listes sont directement modifiables par l'utilisateur à travers de petits formulaires accessibles par l'icône se trouvant à côté des liste de choix correspondantes. Ainsi, l'utilisateur peut lui-même adapter le contenu de sa liste de choix à ses habitudes ou aux particularités de son exercice.

La figure 11 propose le formulaire de personnalisation de la liste des races proposées dans le champ « Races » de la Fiche de l'Animal.

Quelques listes ne proposant que deux ou trois occurrences font exception, par exemple le champ « Sexe » de la table « Animal » ne contient que « Mâle » et « Femelle », et il est impossible à l'utilisateur de modifier cette liste.

La figure 12 montre l'utilisation de la liste de choix « Races » dans la Fiche Animal, tandis que la figure 13 montre le réglage des propriétés susmentionnées en mode Création de Formulaire.

Figure 11 : Personnalisation de la liste des races

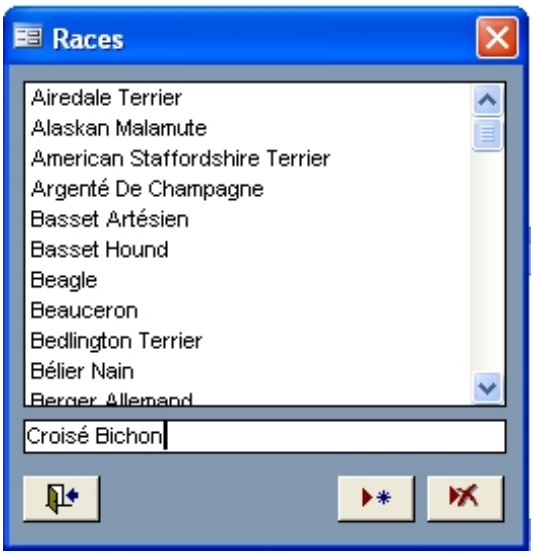


Figure 12 : Utilisation d'une liste de choix dans la Fiche Administrative de l'Animal

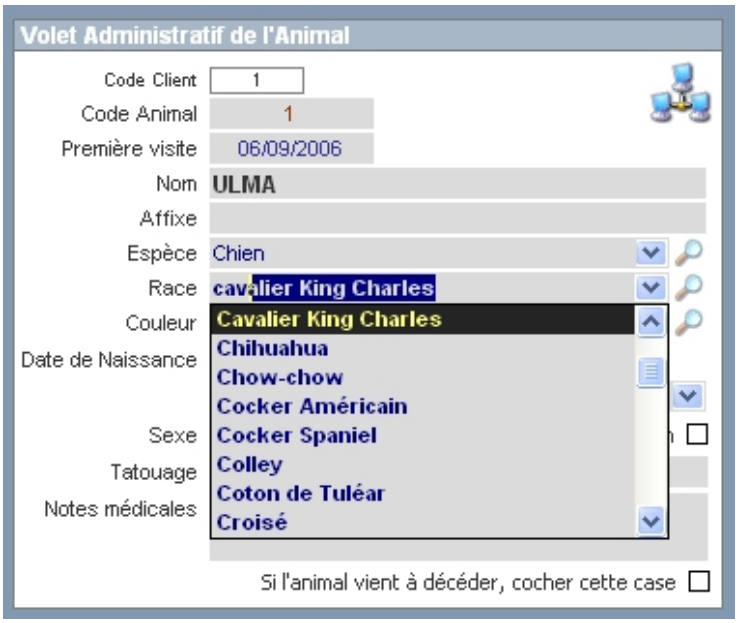
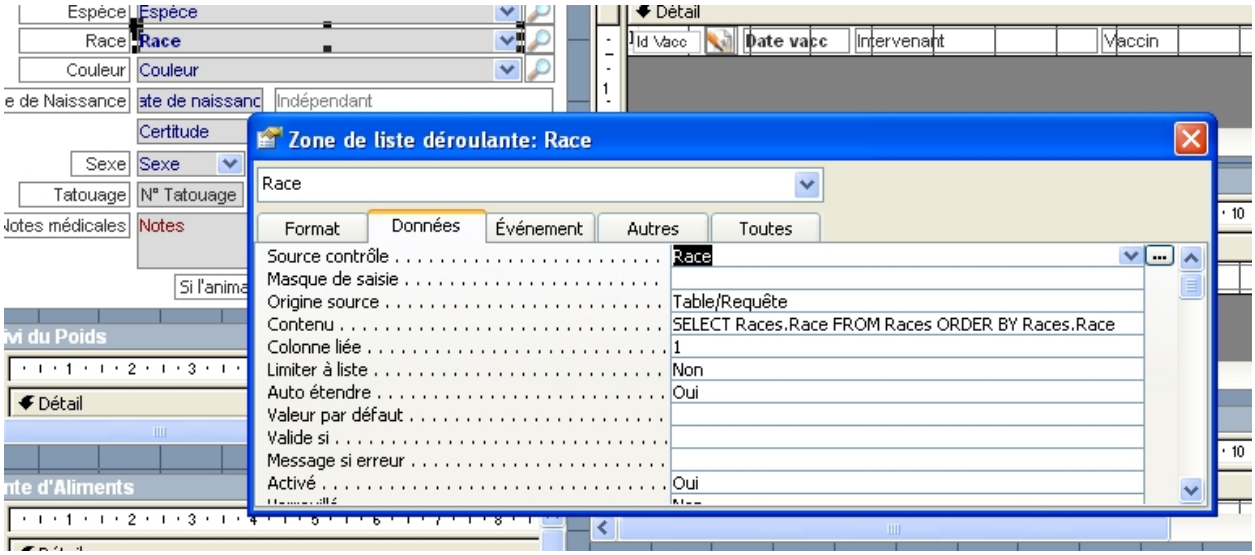


Figure 13 : Utilisation d'une liste de choix dans la Fiche Administrative de l'Animal : réglage des propriétés « Limiter à liste » et « Auto étendre »



Ces listes de choix sont ensuite « remplies » grâce au langage SQL en allant sélectionner, ordonner et trier les données dont nous avons besoin pour afficher la liste des occurences.

Dans l'exemple de notre champ « Race » de la table « Animaux », la liste est constituée à partir de la table « Races » qui maintient la liste des races enregistrées dans la base de données. Il est bien sûr possible d'ajouter, de supprimer ou de modifier cette liste. L'ordre SQL suivant a été utilisé et permet de sélectionner le champ « Races » de la table « Races », et de le trier sur « Races » par ordre croissant :

```
SELECT Races.Race FROM Races ORDER BY Races.Race;
```

Dans la syntaxe du langage SQL chaque ordre se termine par « ; », ce signe de ponctuation permettant d'indiquer la fin de chaque ligne de code. Ce point virgule est requis dans Access mais pas nécessairement dans tous les SGBDR

d) Cas particulier des champs de choix d'un intervenant et des mots-clés

Pour les choix multiples proposés pour les intervenants et les mots-clés, nous voulions la possibilité de hiérarchiser les données et de les catégoriser. En particulier, dans le cas des intervenants, il était intéressant de proposer une alternative à la liste de choix classique de

manière à disposer de catégories regroupant les intervenants « extérieurs », qui pourraient être choisis lors de saisie de cas référés, ou pour des vaccinations effectuées par des confrères.

Pour pallier à cette limite du contrôle « Liste » proposé par Access®, un menu contextuel a été créé pour afficher les différentes possibilités.

La première étape a été de créer une table contenant les intervenants et une clé primaire leur correspondant. Un autre champ appelé « Parent » permet de gérer les catégories : si un intervenant doit appartenir à un autre intervenant, alors on remplit « Parent » avec la clé primaire correspondant à l'intervenant parent. L'écriture d'un code complet dans un module permet :

- de charger et de créer le menu contextuel correspondant au démarrage de SYSTOLE
- de regrouper les fonctions relatives à l'utilisation de ce menu contextuel dans un seul et même endroit.

Le menu contextuel s'affiche automatiquement lorsque l'on clique dans un champ « Intervenant ». Il suffit ensuite de sélectionner l'intervenant correspondant et le code colle automatiquement la valeur choisie dans le champ (voir figure 14).

Figure 14 : Copie d'écran du Menu Contextuel permettant le choix d'un intervenant

Le principe est identique pour la gestion des mots-clés.

Le code complet du module « ChargerIntervenants » et « ChargerMotsClés » est placé en Annexe 2.

II. LA CREATION DE L'INTERFACE : DEUX EXEMPLES PRATIQUES

Dans cette partie, nous allons traiter de la création de l'interface de l'application. Il aurait été bien trop fastidieux et inintéressant de réaliser un catalogue exhaustif de l'interface telle qu'elle a été réalisée. Nous avons donc choisi de traiter deux cas particuliers, en présentant les contraintes liées à leur conception et les choix réalisés pour y apporter des solutions. Nous allons ainsi présenter la conception d'un formulaire de recherche rapide et efficace, puis nous aborderons la création d'états grâce à la fonction d'impression des Cerfas Rage.

A. LE FORMULAIRE DE RECHERCHE

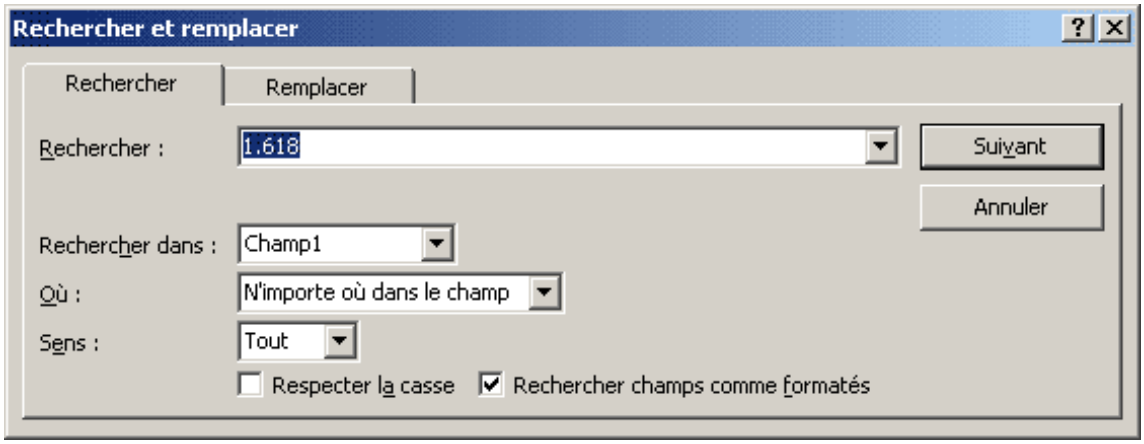
1) Constatations de départ et nécessité de l'outil de recherche

a) Tour d'horizon des techniques de recherche classique sous Access®

Dans une base de données, la recherche d'informations est primordiale. Il faut donc pouvoir accéder facilement et rapidement à un enregistrement particulier pour le consulter, le modifier, ou le supprimer. Access® propose différentes techniques pour effectuer cette recherche.

La plus simple d'utilisation est sans nul doute la commande Rechercher du menu Edition/Rechercher que l'on peut appeler via l'icône Jumelles de la barre d'icônes Base de données ou encore grâce à Ctrl+F (figure 15).

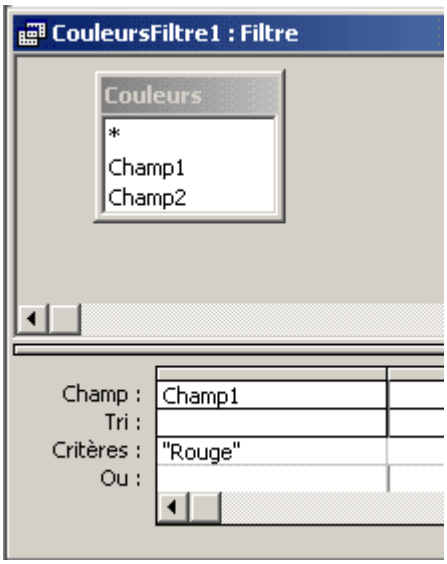
Figure 15 : Copie d'écran de la fenêtre de Recherche d'Access



Celle-ci effectue des recherches séquentielles à partir d'un critère unique. En d'autres termes, elle va parcourir un ensemble d'enregistrement (table ou requête) et s'arrêter sur le prochain enregistrement satisfaisant au critère.

La deuxième possibilité réside dans l'utilisation du filtre (figure 16). Celui-ci, accessible par le menu Enregistrement/Filtre permet de visualiser un groupe d'enregistrements suivant un ou plusieurs critères, un peu à la manière d'une requête.

Figure 16 : Copie d'écran de la fenêtre de définition du filtre



Les boutons 

peuvent également être utilisés pour faire et défaire des filtres. Leur utilisation est simple et intuitive. Le filtrage est intéressant, mais nécessite du temps pour être créé, et que l'utilisateur connaisse bien l'interface d'Access® et possède des notions en base de données, ce qui n'est pas toujours le cas. Le filtre peut en outre être enregistré comme une requête pour être réutilisé par la suite.

Enfin, la création et l'utilisation d'une requête, comme le filtre, permet de visualiser un groupe d'enregistrements d'après un critère. Puissante, elle nécessite malgré tout de solides connaissances en BD et Access®.

b) Contraintes et solutions à apporter

La plupart des utilisateurs de SYSTOLE seront des novices en BD et SQL. Il est ainsi hors de question d'utiliser la création d'une requête, ou même le filtre, afin de réaliser une recherche efficace. L'outil « Rechercher » n'est pas assez puissant : il ne permet de rechercher que sur une seule table et un seul critère.

Il faut donc intégralement créer un formulaire présentant à l'utilisateur des outils facilement compréhensibles afin de filtrer et de rechercher des données.

La fonction principale de l'outil de recherche sera d'extraire un client et/ou un animal particulier afin d'accéder à sa fiche pour la consulter, y saisir ou modifier des données. La recherche doit donc porter à la fois sur la table « Clients » et « Animal ».

Pour faciliter l'emploi de l'outil, nous choisissons de limiter volontairement le nombre de critères utilisables pour les recherches. Nous désirons afficher et discerner des propriétaires différents et leurs animaux. Sont donc sélectionnés le nom, le prénom, l'adresse et la ville du propriétaire, et uniquement le nom de l'animal.

L'écran de recherche devra enfin donner la possibilité d'accéder soit à la Fiche Client, soit à la Fiche Animal de l'enregistrement sélectionné, et d'accéder à une fiche vierge pour la création d'un nouveau client le cas échéant.

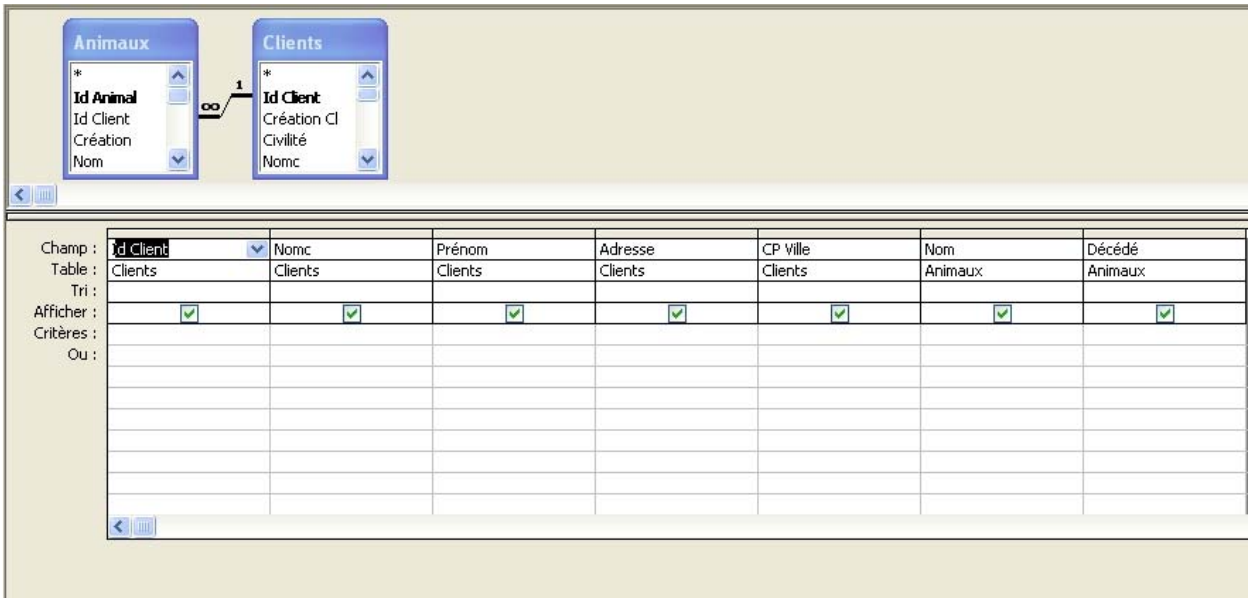
c) Choix techniques pour la création de l'outil de recherche

Puisque la recherche porte sur plus d'une table, nous décidons de créer une requête et d'y inclure l'ensemble des champs sur lesquels nous désirons appliquer une recherche. Rappelons qu'une requête est une grille que nous remplissons afin de faciliter la génération de code SQL. La figure 17 montre une vue de la grille de la requête « Recherche_Client+Animal »

La constitution de la requête suit un schéma précis : en premier lieu, les tables contenant les champs devant intervenir dans la requête sont choisies. Puis, les champs sont inclus un à un dans la grille en précisant leur table d'origine. Enfin, il est possible de choisir un ordre de tri pour le champ, et de choisir si la colonne sera affichée dans la grille résultante. Le champ [Id Client] est

la clé primaire de la table des Clients. Nous décidons de l'inclure dans la requête en envisageant la possibilité d'appeler directement la Fiche du Client via le formulaire de recherche.

Figure 17 : Copie d'écran de la fenêtre de création de requête



Dans une requête classique, il est possible de rentrer les critères que l'on désire utiliser pour la recherche. Dans notre cas, étant donné que les critères changeront à chaque session d'interrogation, aucun n'est saisi.

C'est l'utilisateur qui saisira ses propres critères dans le formulaire, tandis qu'un code Visual Basic® s'occupera d'interroger la requête en utilisant ces éléments.

Une fois la requête constituée, les données sont extraites et organisées. Le résultat de la requête peut être visualisé, tout comme les données contenues dans une table. La figure 18 montre la table de résultats correspondant à notre requête. Dans ce cas, tous les enregistrements de la BD sont visionnés puisque aucun critère n'est utilisé pour restreindre la sélection des enregistrements.

La requête que nous avons créée énumère chaque animal de chaque client. Il est donc normal et logique d'y retrouver plusieurs lignes ayant le même client. Ainsi, si un client possède quatre animaux, il apparaîtra dans quatre lignes de la requête.

Figure 18 : Copie d'écran de la requête « Recherche_Client+Animal »

Id Client	Nom	Prénom	Adresse	CP Ville	Nom	Décédé
1	BIZART	Bernard	20 Residence Florival	59188 SAINT AUBERT	JAZZ	☐
3	PLANQUE	Françoise	1 Chemin Thiossart	59218 VENDEGIES AU BOIS	IRIS	☐
4	DECALLION	Marie Thérèse	8 Rue Paradis	59296 AVESNES LE SEC	JOY	☐
5	BOSCHETTI	Elsa	12 Rue Luthon	59218 NEUVILLE EN AVESNOIS	CHANEL	☐
6	HERBIN	Martine	130 Rue Du Général De Gaule	59730 SOLESMES	SCOTTY	☐
7	DELHAY	Marie Thérese	26 Rue Pasteur	59227 SAULZOIR	SMARTIES	☐
7	DELHAY	Marie Thérese	26 Rue Pasteur	59227 SAULZOIR	OLIVER	☐
7	DELHAY	Marie Thérese	26 Rue Pasteur	59227 SAULZOIR	PRINCESSE	☐
8	VAILLANT	Bernadette	20 Rue Jaures	59227 SAULZOIR	BIBOU	☐
9	GRIFFART	Paulette	84 Rue De Cambrai	59188 VILLERS EN CAUCHIES	SUN	☐
10	POTTIEZ	Sylvie	14 Rue Martin Hecquet	59227 SAULZOIR	OCEANE	☐
11	DELSARTE	Jonathan	48 Rue Peri	59129 AVESNES LES AUBERT	WARRIOR	☐
12	VERQUIN	Anne Marie	130 Rue Du Miroir	59213 BERMERAIN	IGOR	☐
12	VERQUIN	Anne Marie	130 Rue Du Miroir	59213 BERMERAIN	BELIZE	☐
13	BOURDIN	Mickael	12 Rue Karl Marx	59129 AVESNES LES AUBERT	BOUBA	☐

Enfin, nous souhaiterions que l'utilisateur n'ait pas à cliquer sur un bouton afin de lancer sa recherche. Nous optons pour un formulaire dynamique, mis à jour automatiquement à chaque modification d'une des zones de saisie.

2) Création du formulaire de recherche

Pour réaliser notre outil de recherche, nous allons utiliser des contrôles indépendants, c'est à dire qui ne sont pas liés directement à une source de données (une table). Pour un contrôle lié à un champ dune table, un changement de valeur du contrôle implique un changement de valeur dans une table. Notre but est de disposer de contrôles n'étant liés à aucune donnée d'aucune table afin de pouvoir réaliser une saisie des critères de recherche. Ces critères ne sont pas destinés à être enregistrés, mais seront exploités de manière ponctuelle par le code de recherche.

L'idée est de rechercher sur les cinq critères cités précédemment. Nous ferons des recherches sur une combinaison des cinq éléments avec les spécifications suivantes :

- Pour le nom, le prénom, la ville du propriétaire, et le nom de l'animal : critère exact, c'est à dire choix parmi la liste des occurrences enregistrées dans la base de données.

- Pour l'adresse : critère contenu, c'est à dire que la réponse doit contenir le critère (par exemple : « HER » peut représenter « HERGE » ou « HERNANDEZ »).

Pour les critères exacts nous utiliserons comme zone de saisie des listes déroulantes ou Combo Box, et pour le critère contenu une boîte de saisie Text Box.

a) Charte graphique de l’interface et contrôles nécessaires

Le formulaire est fabriqué en mode « Création ». Pour la charte graphique, nous décidons d’utiliser une combinaison de tons bleus et blancs. Ces couleurs sont reposantes, agréables, et permettront d’utiliser le logiciel plusieurs heures sans que la vue ne se fatigue trop. L’ensemble des formulaires de SYSTOLE est basé sur cette charte graphique. Un logo caractéristique est placé dans le coin supérieur gauche du formulaire, dans le but de donner à l’application une identité et une certaine homogénéité.

Les icônes utilisées proviennent de celles disponibles sous Microsoft Windows®. Elles sont modifiées (diminution de taille, changement de couleur, fusion de deux ou de plusieurs icônes…) et la couleur du fond est adaptée à celle du formulaire. L’ensemble du logiciel utilise des icônes récurrentes pour des fonctions similaires, de telle sorte que l’apprentissage par les utilisateurs est simplifié.

On choisit pour ce formulaire l’option « Fenêtre indépendante ». Cela signifie que le formulaire s’ouvrira en premier plan, dans une fenêtre à ses dimensions.

L’interface maintenant définie, il ne nous reste qu’à créer les contrôles dont nous aurons besoin. Habituellement, les contrôles indépendants peuvent être placés sur un formulaire également indépendant.

Pour les besoins des fonctions d’affichage du formulaire, nous allons néanmoins le rendre dépendant de la requête de recherche que nous avons créé. En effet, nous aimerions que la liste des animaux vivants et décédés du client soit également affichée, telles qu’elles le sont sur la Fiche Administrative du Client.

Pour ce faire, le moyen le plus simple d’y parvenir est d’utiliser les propriétés des **sous-formulaires**. Ceux-ci agissent comme des formulaires imbriqués dans le formulaire parent, et permettent par une liaison grâce aux clés primaires et étrangères d’afficher des enregistrements provenant d’une autre table. Cet affichage est conditionné par l’enregistrement affiché sur le formulaire parent.

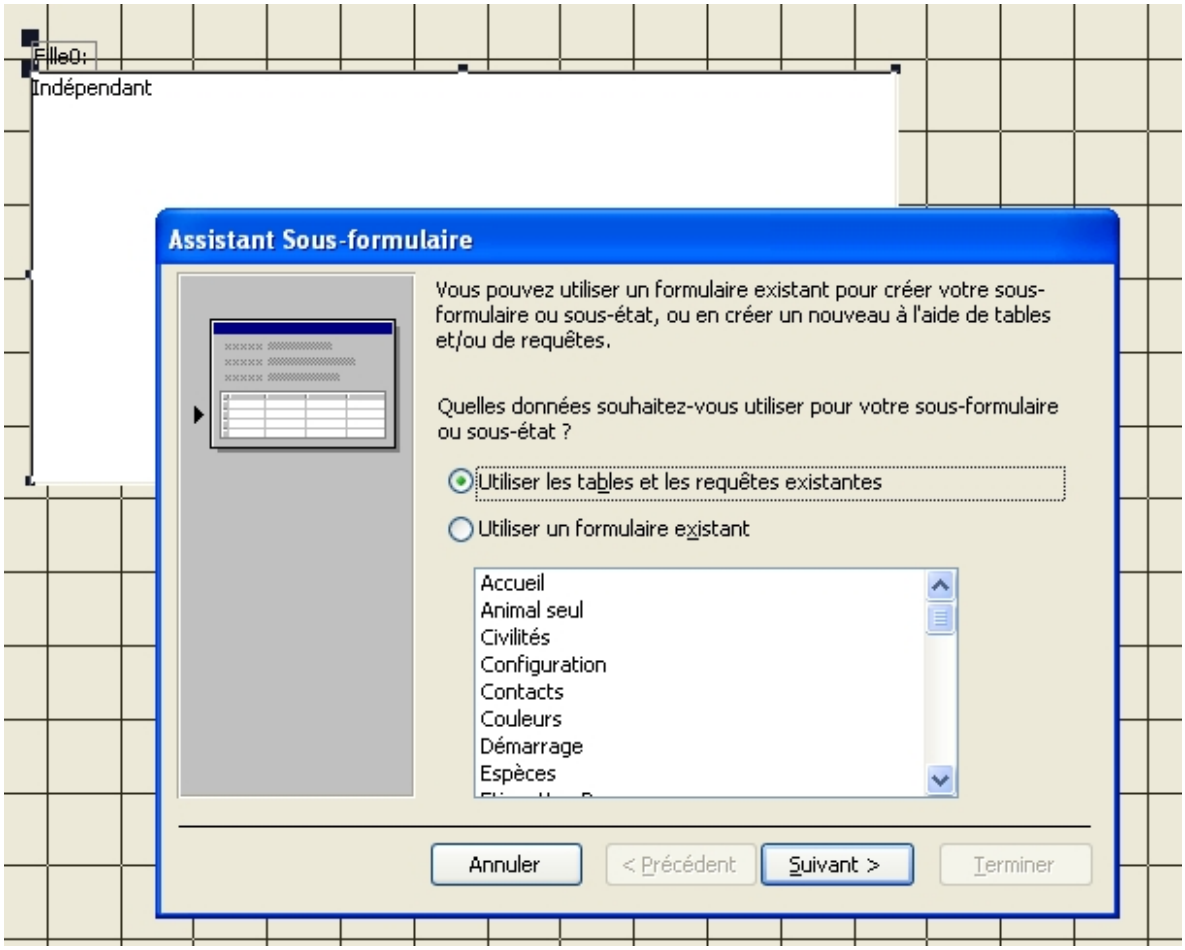
Dans notre exemple, nos sous-formulaires sont construits sur des ordres SQL permettant de sélectionner les animaux vivants d’une part, et les décédés d’autre part.

Puis, il suffit de créer indépendamment des formulaires permettant l'affichage de ces données.

Enfin, en mode « Création », on inclut une plage de sous-formulaire au formulaire père. Cela a pour effet d'ouvrir une boîte de dialogue, comme le montre la figure 19. Il est alors possible de choisir d'afficher un formulaire existant ou d'en créer un nouveau.

Dans notre cas, nos formulaires sont déjà disponibles. Access® réalise alors automatiquement la liaison entre le formulaire père et le formulaire fils sur la base de la clé [Id Client].

Figure 19 : Copie d'écran de la boîte de dialogue de l'Assistant Sous-formulaire



Le choix des noms des contrôles est toujours très important. Nous décidons d'utiliser les trois premières lettres pour rappeler le type de contrôle : **txt** pour Text Box (zone de texte), **cmb** pour Combo Box (liste déroulante), **chk** pour Check Box (case à cocher), **lbl** pour Label (étiquette) et **lst** pour List Box (zone de liste).

Nous utiliserons les cases à cocher pour déterminer si la recherche en cours utilisera ou non le critère. Par exemple, si la case chkNom est cochée, la zone de saisie correspondante est affichée et l'utilisateur doit saisir le nom qu'il recherche ; si la case chkAnimal est décochée, l'utilisateur n'effectue pas de sélection sur le nom de l'Animal. Pour les zones de listes déroulantes, nous utiliserons la propriété « auto-étendre » décrite plus haut.

Une étiquette « lblstats » est créée, elle servira à afficher les statistiques de la recherche, c'est-à-dire le nombre d'enregistrements extraits correspondants aux critères demandés.

Enfin, les résultats de la recherche seront affichés dans une zone de liste « lstResults ».

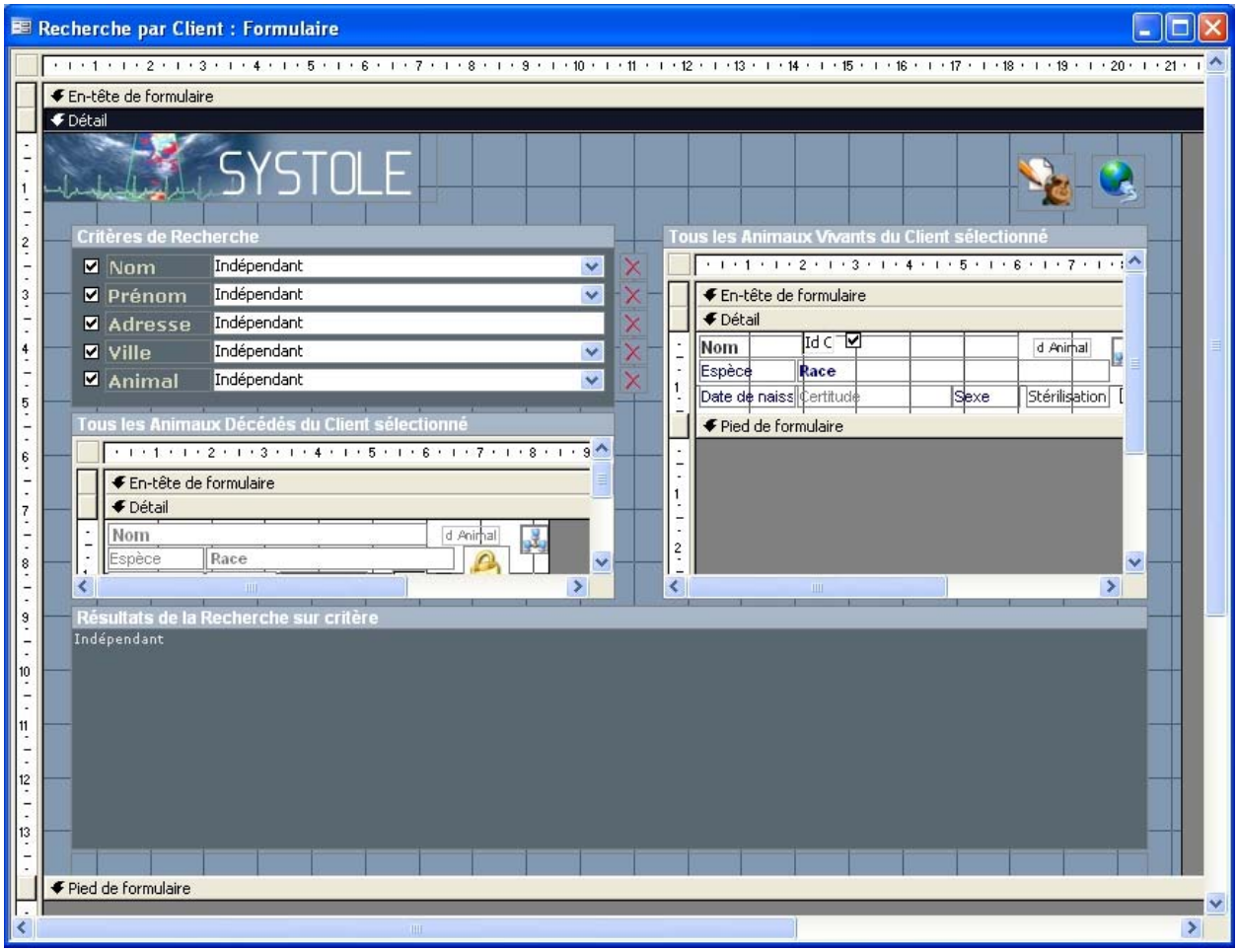
Tous les contrôles utilisés sur le formulaire sont repris dans le tableau 2.

Tableau 2 : Récapitulatif des contrôles créés et utilisés sur le formulaire de recherche

TextBox	ComboBox	CheckBox	Image	Label	ListBox
	cmbRechNom	chkNom	imgnom		
	cmbRechPrénom	chkPrénom	imgprenom		
txtRechAdresse		chkAdresse	imgadresse		
	cmbRechCPVille	chkCPVille	imgville		
	cmbRechAnimal	chkAnimal	imganimal		
				lblStats	
					lstResults

La figure 20 montre le formulaire terminé en mode « Création », avec tous les éléments et les contrôles en place.

Figure 20 : Copie d'écran du formulaire de recherche finalisé



Maintenant que notre formulaire est terminé, il ne reste plus qu'à créer le code sous-jacent à son fonctionnement correct. Ce code aura deux fonctions : tout d'abord, gérer l'interface et ensuite accéder aux données de la requête.

b) Code Visual Basic® associé à l'interface

Le code doit d'abord prendre en charge l'affichage des zones de saisies et des icônes de vidange en fonction des cases à cocher s'y rapportant. Lorsque l'utilisateur coche la case, la zone et l'icône doivent s'afficher, et inversement disparaître si la case est décochée. Le formulaire doit également être mis à jour après chaque modification. C'est le rôle de l'appel du sous-programme

« RefreshQuery » dont nous allons parler plus bas. Voici le code associé à l'événement « SurClick » des différentes cases à cocher :

```
Private Sub chkNom_Click()

If Me.chkNom Then
   Me.cmbRechNom.Visible = True
   Me.imgnom.Visible = True
Else
   Me.cmbRechNom.Visible = False
      Me.imgnom.Visible = False
End If

RefreshQuery

End Sub
```

```
Private Sub chkPrénom_Click()

If Me.chkPrénom Then
   Me.cmbRechPrénom.Visible = True
   Me.imgprenom.Visible = True
Else
   Me.cmbRechPrénom.Visible = False
      Me.imgprenom.Visible = False
End If

RefreshQuery

End Sub
```

```
Private Sub chkAdresse_Click()

If Me.chkAdresse Then
   Me.txtRechAdresse.Visible = True
   Me.imgadresse.Visible = True
Else
   Me.txtRechAdresse.Visible = False
      Me.imgadresse.Visible = False
End If

RefreshQuery

End Sub
```

```
Private Sub chkCPVille_Click()

If Me.chkCPVille Then
   Me.cmbRechCPVille.Visible = True
   Me.imgville.Visible = True
Else
   Me.cmbRechCPVille.Visible = False
      Me.imgville.Visible = False
```

```
End If

RefreshQuery

End Sub
```

```
Private Sub chkAnimal_Click()

If Me.chkAnimal Then
    Me.cmbRechAnimal.Visible = True
    Me.imganimal.Visible = True
Else
    Me.cmbRechAnimal.Visible = False
        Me.imganimal.Visible = False
End If

RefreshQuery

End Sub
```

Les icônes « img » permettent d'effacer rapidement par un click de souris le contenu de la zone de saisie correspondante. Là aussi, après chaque utilisation, la procédure « RefreshQuery » est appelée afin que les données du formulaire se mettent à jour.

```
Private Sub imgadresse_Click()
Me.txtRechAdresse.Value = ""
RefreshQuery
End Sub
```

```
Private Sub imgnom_Click()
Me.cmbRechNom.Value = ""
RefreshQuery
End Sub
```

```
Private Sub imgprenom_Click()
Me.cmbRechPrénom.Value = ""
RefreshQuery
End Sub
```

```
Private Sub imgville_Click()
Me.cmbRechCPVille.Value = ""
RefreshQuery
End Sub
```

```
Private Sub imgAnimal_Click()
Me.cmbRechAnimal.Value = ""
RefreshQuery
End Sub
```

Nous devons maintenant écrire les instructions SQL qui permettront aux Combo Box de se remplir. Ces instructions sont placées dans la propriété « Contenu » de la liste déroulante (voir figure 21). La figure 22 présente l'assistant accessible par le bouton « … ». Cet assistant permet de créer facilement le SQL par sélection des éléments qui nous intéressent, directement comme dans une requête.

Figure 21 : Copie d'écran des paramètres de la zone de liste déroulante « cmbRechAnimal »

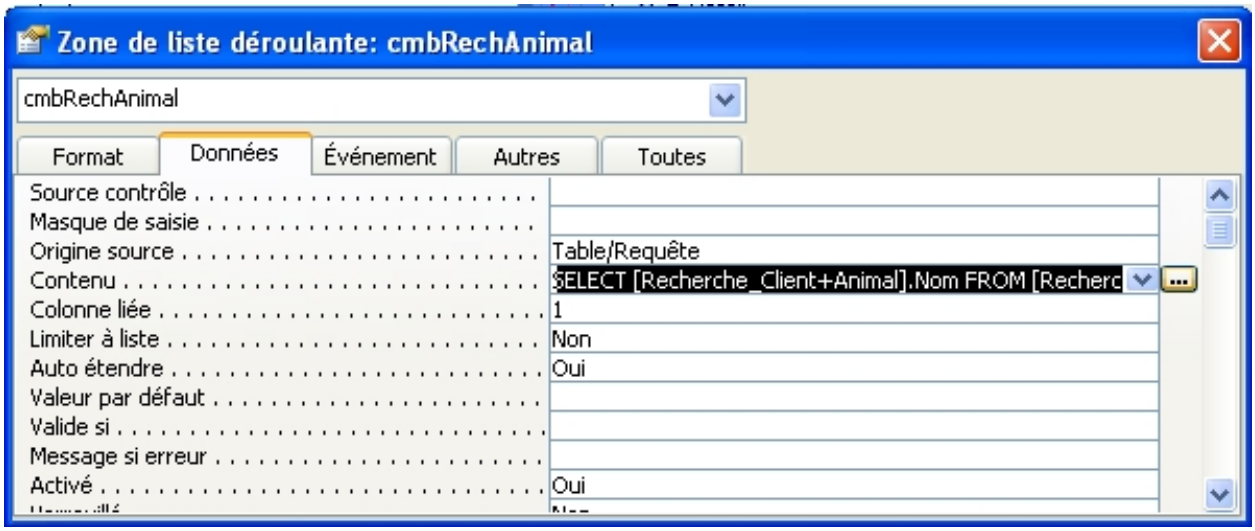
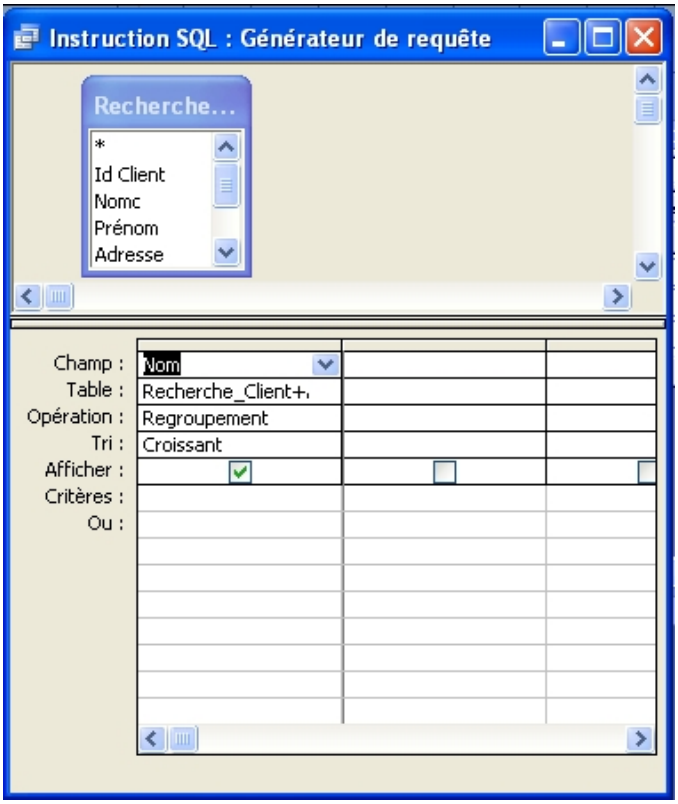


Figure 22 : Copie d'écran de l'assistant contenu SQL de la zone de liste déroulante « cmbRechAnimal »



Voici ces instructions :

- pour le Nom : SELECT Clients.Nomc FROM Clients GROUP BY Clients.Nomc ORDER BY Clients.Nomc;
- pour le Prénom : SELECT Clients.Prénom FROM Clients GROUP BY Clients.Prénom ORDER BY Clients.Prénom;
- pour la Ville : SELECT Clients.[CP Ville] FROM Clients GROUP BY Clients.[CP Ville] ORDER BY Clients.[CP Ville];
- pour le Nom de l'Animal : SELECT [Recherche_Client+Animal].Nom FROM [Recherche_Client+Animal] GROUP BY [Recherche_Client+Animal].Nom ORDER BY [Recherche_Client+Animal].Nom;

L'instruction GROUP BY permet de ne placer dans la liste qu'une seule fois chaque occurrence de la base. Par exemple, si dix personnes portent le nom « DUPONT », l'occurrence « DUPONT » n'apparaîtra qu'une fois dans la liste. L'instruction ORDER BY permet de classer les occurrences par ordre alphabétique.

Note : l'ordre SQL pour le Nom de l'Animal effectue la recherche dans la requête « Recherche_Client+Animal », tandis que pour les données du client, cette recherche est effectuée sur la table « Clients ». Ceci provient du fait qu'historiquement, la première version du formulaire ne permettait pas la recherche sur le nom de l'animal couplé au client. Pour remédier à cette lacune, la requête « Recherche_Client+Animal » a été créée et les manipulations réalisées sur elle. En conséquence, il est possible de remplacer la table « Clients » par la requête dans les ordres SQL permettant de remplir les zones de listes déroulantes correspondant au Client.

Il ne nous reste plus qu'à appeler le sous-programme « RefreshQuery » lorsque les zones de saisie sont modifiées afin d'appliquer automatiquement la recherche et de mettre à jour le formulaire. Voici le code correspondant :

```
Private Sub cmbRechAnimal_BeforeUpdate(Cancel As Integer)
 RefreshQuery
End Sub
```

```
Private Sub cmbRechNom_BeforeUpdate(Cancel As Integer)
 RefreshQuery
End Sub
```

```
Private Sub cmbRechPrénom_BeforeUpdate(Cancel As Integer)
 RefreshQuery
End Sub
```

```
Private Sub cmbRechCPVille_BeforeUpdate(Cancel As Integer)
 RefreshQuery
End Sub
```

```
Private Sub txtRechAdresse_BeforeUpdate(Cancel As Integer)
 RefreshQuery
End Sub
```

Comme nous l'avons défini plus haut, nous désirons que notre recherche s'effectue automatiquement à chaque changement des contrôles du formulaire. Ecrire le code correspondant pour chaque contrôle modifié aurait été bien trop fastidieux. L'idée est d'écrire un sous-programme standard appelé à chaque modification.

C'est le rôle du sous-programme « RefreshQuery ». Nous venons de voir comment ce sous-programme est appelé après la modification de chaque contrôle. Nous allons à présent expliquer son fonctionnement.

Avant tout, faisons une brève présentation de la syntaxe des ordres SQL utilisés dans ce code.

SELECT [obligatoire] liste des champs séparés par une virgule, au besoin renommés par un alias grâce à l'instruction "As".

*Les champs choisis vont être les colonnes de la requête. Pour sélectionner tous les champs dans une requête sur une seule table : SELECT *. Pour sélectionner tous les champs dans une requête sur plusieurs tables : SELECT Table1.**

Par exemple : SELECT NomFamille As NomF, PrenomUsuel As Prenom, DateNaissance

FROM [obligatoire] table sur laquelle porte la requête, au besoin elle aussi renommée par un alias grâce à l'instruction "As".

Par exemple : FROM tblPersonnel la requête porte sur la table "tblPersonnel"

WHERE [Facultatif] liste des conditions séparées par un opérateur logique "And" ou "Or".

Les conditions sont exprimées : Champ = Valeur

Exemples :

WHERE tblPersonnel.NomFamille = 'MARTIN'

Ici, nous sélectionnons les personnes qui portent le nom de Martin (Claude Martin et Jean Martin)

WHERE tblPersonnel.NomFamille = 'MARTIN' And tblPersonnel.PrenomUsuel Like 'C'*

Ici nous sélectionnons les personnes qui portent le nom de Martin ET dont le prénom comment par un C (Claude Martin, mais pas Jean Martin)

WHERE tblPersonnel.NomFamille = 'MARTIN' Or tblPersonnel.PrenomUsuel Like 'C'*

Ici nous sélectionnons les personnes qui portent le nom de Martin OU dont le prénom comment par un C (Claude Martin et Jean Martin mais aussi Christian Janvier)

Nous utilisons ces instructions afin de réaliser la requête de base. Tout d'abord, le sous-programme est nommé, puis les variables « SQL » et « SQLWhere » sont déclarées en tant que chaînes de caractères (« String »).

```
Private Sub RefreshQuery()
Dim SQL As String
Dim SQLWhere As String
```

On attribue à la variable « SQL » la valeur suivante :

```
SQL = "SELECT [Id Client], [Nom], [Nomc], [Prénom], [Adresse], [CP Ville] FROM [Recherche_Client+Animal] Where [Recherche_Client+Animal]![Id Client] <> 0 "
```

Cette instruction sélectionne l'ensemble des enregistrements de la requête, en imposant comme condition que le champ [Id Client] ne soit pas nul. L'utilisation de la condition WHERE est ici sans intérêt, mais ceci à l'avantage de l'inclure dans la chaîne SQL afin de pouvoir ajouter nos critères.

Le but de la suite du code est d'inspecter chaque case à cocher. Si la case est cochée, alors le code rajoute la valeur de la zone de saisie correspondante en tant que condition à la requête déjà formulée (c'est le rôle de la ligne SQL = SQL & « … »). Si la case est décochée, rien n'est

ajouté. Pour les critères exacts, on utilise la syntaxe « = « valeur » ». Pour les critères non exacts, on utilise « Like « valeur » ».

```
If Me.chkAnimal Then
   SQL = SQL & "And [Recherche_Client+Animal]![Nom] = '" & Me.cmbRechAnimal & "' "
End If
If Me.chkNom Then
   SQL = SQL & "And [Recherche_Client+Animal]![Nomc] = '" & Me.cmbRechNom & "' "
End If
If Me.chkPrénom Then
   SQL = SQL & "And [Recherche_Client+Animal]![Prénom] = '" & Me.cmbRechPrénom & "' "
End If
If Me.chkAdresse Then
   SQL = SQL & "And [Recherche_Client+Animal]![Adresse] like '*" & Me.txtRechAdresse & "*' "
End If
If Me.chkCPVille Then
   SQL = SQL & "And [Recherche_Client+Animal]![CP Ville] = '" & Me.cmbRechCPVille & "' "
End If
```

La ligne suivante permet d'extraire la chaîne de caractère se trouvant après WHERE, afin de l'utiliser plus loin dans la fonction DCount qui nous permettra de compter le nombre d'enregistrements retournés par la requête avec les critères proposés. Cette chaîne est nommée « SQLWhere »

```
SQLWhere = Trim(Right(SQL, Len(SQL) - InStr(SQL, "Where ") - Len("Where ") + 1))
```

Enfin, nous appliquons à la requête un dernier tri par ordre alphabétique croissant des noms d'animaux.

```
SQL = SQL & "ORDER BY [Recherche_Client+Animal].Nom;"
```

La dernière étape du code affecte à l'étiquette « lblStats » le nombre d'enregistrements correspondant à la requête par rapport au nombre d'enregistrement total de la requête.

```
Me.lblstats.Caption = "Il y a actuellement " & DCount("*", "[Recherche_Client+Animal]", SQLWhere) & "
enregistrements sur " & DCount("*", "[Recherche_Client+Animal]") & " Animaux correspondant à votre
Requête"
```

Pour terminer, la chaîne SQL est affectée à la propriété « Contenu » de la zone de liste « lstResults », puis celle-ci est mise à jour, c'est-à-dire que le nouvel ordre SQL défini est exécuté et son résultat affiché. Le sous-programme est alors fermé.

```
Me.lstResults.RowSource = SQL
Me.lstResults.Requery
End Sub
```

3) Finalisation et utilisation du formulaire de recherche

a) Gestion des paramètres d'ouverture.

Nous souhaiterions qu'à l'ouverture du formulaire, aucun critère ne puisse venir filtrer la table de manière parasite. De plus, le contrôle « lblStats » devrait afficher uniquement le nombre total d'enregistrement dans la base de données.

Pour y parvenir, nous pouvons utiliser l'événement « Sur Chargement » attaché au formulaire. Le code entré dans cette section est exécuté à chaque chargement de la fenêtre.

```
Private Sub Form_Load()
Dim ctl As Control

For Each ctl In Me.Controls
   Select Case Left(ctl.name, 3)
      Case "chk"
         ctl.Value = 0
      Case "lbl"
         ctl.Caption = "Il y a actuellement " & DCount("*", "[Recherche_Client+Animal]") & " Animaux
dans la base de Systole."
      Case "txt"
         ctl.Visible = False
         ctl.Value = ""
      Case "cmb"
         ctl.Visible = False
      Case "img"
         ctl.Visible = False

   End Select
Next ctl
```

Cette partie de code est composée d'une boucle For…Next qui s'exécute un nombre déterminé de fois. En l'occurrence, ici, on définit la variable « ctl » en tant que contrôle. La boucle va examiner chaque contrôle présent sur le formulaire, et lui appliquer éventuellement un statut de démarrage.

Le principe est d'utiliser le nom que nous avons donné aux contrôles. La fonction **Left(ctl.name, 3)** permet d'extraite les trois premières lettres du nom de chaque contrôle.

Ensuite, selon le résultat (fonction SELECT CASE), un statut est appliqué au contrôle. Ainsi :

- chaque contrôle « chk » se voit attribuer l'état « décoché »
- le contrôle « lbl » affiche « Il y a actuellement x enrgistrements dans la base de Systole »
- le contrôle « txt » est rendu invisible et on lui attribue comme valeur une chaîne vide
- les contrôles « cmb » et « img » sont rendus invisibles

De plus, la zone de liste « lstResults » est remplie avec tous les enregistrements présents dans la requête, puis l'ordre SQL est appliqué :

```
Me.lstResults.RowSource = "SELECT [Id Client], [Nom], [Nomc], [Prénom], [Adresse], [CP Ville] FROM [Recherche_Client+Animal] ORDER BY [Recherche_Client+Animal.Nomc];"
Me.lstResults.Requery
```

Enfin, l'ouverture du formulaire de recherche se fait toujours à partir de la Fiche Administrative du Client. De manière à ce que la liste se repositionne automatiquement sur l'enregistrement sélectionné à l'ouverture du formulaire, une portion de code est écrite pour récupérer la valeur [Id Client] affichée. Puis, celle-ci est réinjectée dans la zone de liste. Enfin, le sous-programme est fermé.

```
Dim rs As Object

    Set rs = Me.Recordset.Clone
    rs.FindFirst "[Id Client] = " & Str(Nz(Me![lstResults], 0))
    If Not rs.EOF Then Me.Bookmark = rs.Bookmark

End Sub
```

La figure 23 montre l'aspect du formulaire de recherche lors de son démarrage. Son affichage correspond effectivement aux conditions données par le code attribué à l'événement « Sur Chargement ».

b) Accès aux Fiches Client et Animal à partir du formulaire

Bien qu'il remplisse les tâches pour lesquelles nous l'avons créé, ce formulaire de recherche ne serait pas très utile si il ne nous permettait pas de pouvoir accéder aux Fiches enregistrées.

Pour ce faire, deux possibilités s'offrent à l'utilisateur :

- soit cliquer sur l'icône  située dans la liste des Animaux. Il s'agit de l'icône de connexion, qui permet de naviguer entre Client et Animal. Elle possède dans ses propriétés un code associé à l'événement SurClick qui ouvre la Fiche Administrative de l'Animal correspondante, et qui ferme automatiquement le formulaire de recherche.

- soit réaliser un double click sur la ligne de la zone de liste qui nous intéresse, auquel cas c'est la Fiche Administrative du Client qui sera ouverte.

Le code suivant est associé à l'événement SurDoubleClick de la zone lstResults. Il ferme également le formulaire de recherche afin d'éviter toute interférence avec d'autres fenêtres :

```
Private Sub lstResults_DblClick(Cancel As Integer)
DoCmd.OpenForm "Fiche Administrative Client", acNormal, , "[Id Client] = " & Me.lstResults
DoCmd.close acForm, "Recherche par Client"
End Sub
```

La méthode « Open Form » permet d'ouvrir un formulaire existant, et d'y afficher un enregistrement particulier dont on connaît la clé primaire. En l'occurrence, on utilise la colonne [Id Client] de la requête « Recherche_Client+Animal ».

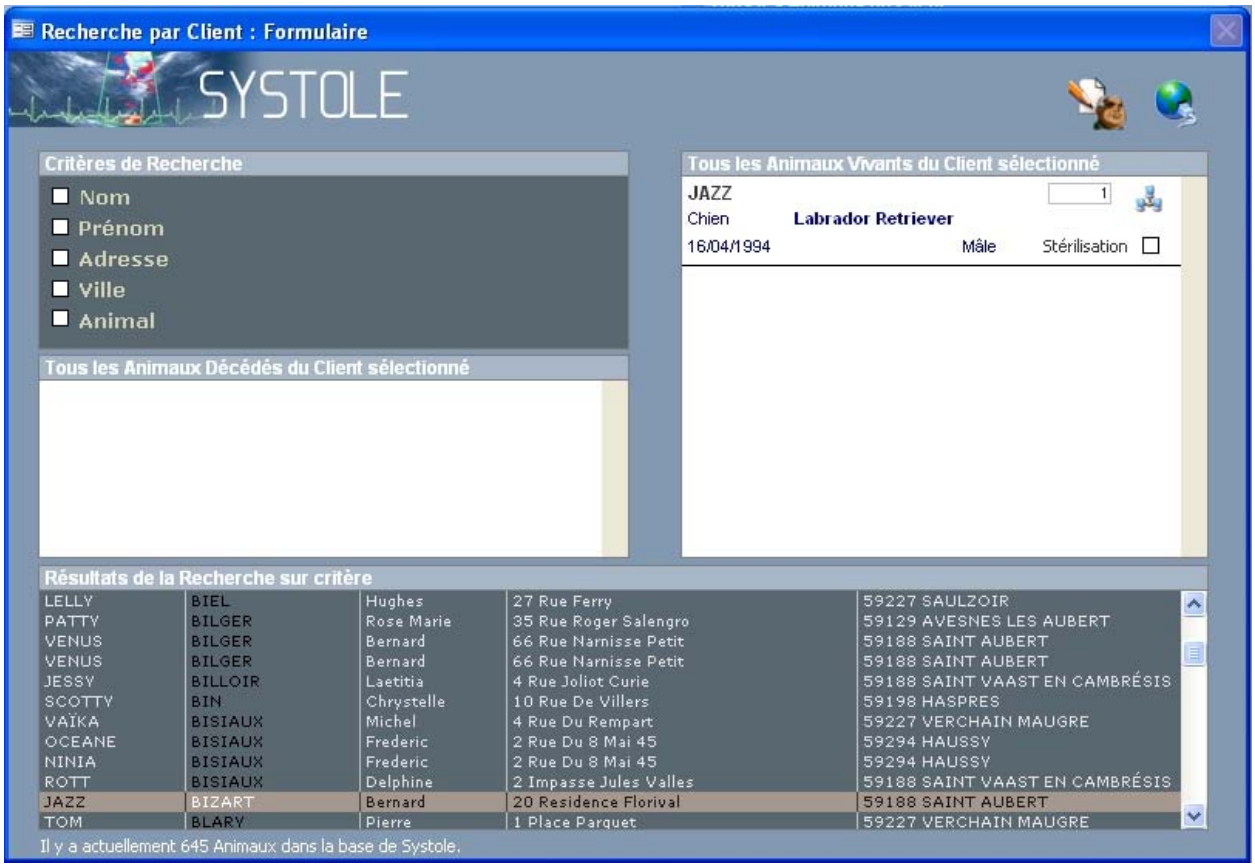
Cette méthode nécessite que la propriété « Colonne liée » de la zone de liste lstResults soit égale à un (dans la requête, [Id Client] est la colonne numéro un).

Enfin, deux autres icônes permettent de compléter les possibilités offertes par le formulaire :

- l'icône  ferme le formulaire de recherche et charge une Fiche Client vierge. Ainsi, si le client n'existe pas dans la base de données, on peut démarrer sa création immédiatement. Ceci est également permis par l'utilisation de la méthode « OpenForm » qui possède une option permettant de démarrer sur un nouvel enregistrement.

- l'icône  permet de ferme le formulaire de recherche, sans effectuer aucune autre opération.

Figure 23 : Copie d'écran du formulaire de Recherche dans l'état au démarrage



B. LES ETATS D'IMPRESSION DES CERFAS RAGE

1) Constatations de départ et nécessité de l'outil

a) Rappels brefs sur la législation de la Rage en France

Depuis 2001, la France est déclarée indemne de Rage au sens de l'Office International des Epizooties, c'est-à-dire que notre pays n'a pas connu de cas de Rage autochtone dans les deux années précédant cette date [9].

Dans une zone indemne, comme l'est la France métropolitaine actuellement, la vaccination contre la Rage des Carnivores Domestiques n'est plus obligatoire. Elle le reste cependant pour certains individus dans certaines conditions :

- Chiens et chats **importés en France**, ou y revenant avec leur maître (règlement CE N° 998/2003) ou dans un cadre commercial (Décision de la Commission 2004/595).
- Chiens et chats **introduits en Corse ou dans un département d'Outre-mer**, en provenance de la France continentale (arrêté ministériel paru au Journal officiel du 10 décembre 1991).
- Chiens et chats en provenance de France et devant circuler avec leur maître (règlement CE N° 998/2003) ou dans un cadre commercial (arrêté ministériel du 20 mai 2005) **dans les pays de la Communauté européenne**.
- **Lévriers** engagés dans des **courses publiques** (arrêté ministériel du 22 janvier 1985)
- Chiens et chats introduits dans un **camping** ou un **centre de vacances** sur tout le territoire national (arrêté ministériel du 22 janvier 1985).
- Chiens **dangereux** ($1^{ère}$ et $2^{ème}$ catégories) (loi du 6 janvier 1999).

La vaccination anti-rabique des animaux de compagnie ne peut être effectuée que par des vétérinaires détenteurs du mandat sanitaire. Elle est réalisée à l'aide d'un vaccin inactivé**,** en une injection de primo vaccination si il est adjuvé, en deux injections à un mois d'intervalle si il est non adjuvé.

A l'heure actuelle, les vaccins classiquement utilisés sont tous adjuvés : la primo vaccination, réalisée une seule fois, fait l'objet de la délivrance d'un certificat de couleur bleue, dont la validité courre de un mois après la date de l'injection et pendant un an.

Les vaccinations suivantes, dites « de rappel » nécessitent une seule injection effectuée moins d'un an après la primo vaccination. Chacune d'elle est attestée par la délivrance d'un certificat de couleur rose. Le rappel prend effet le jour de son établissement, il est valable un an, jour pour jour.

Les certificats doivent comporter certaines informations obligatoires : le nom et l'adresse du vétérinaire sanitaire réalisant la vaccination, les coordonnées complètes du propriétaire et de l'animal, ainsi que son numéro d'identification (tatouage ou puce électronique), la date de la vaccination, le laboratoire fournisseur et le numéro de lot du vaccin.

b) Conséquences sur le cahier des charges de la gestion de l'impression des Cerfa

La plupart des données nécessaires à l'établissement du Cerfa Rage sont déjà contenues dans SYSTOLE et ne nécessitent donc pas d'être enregistrées. Par contre, pour les informations pouvant changer d'une année sur l'autre, une saisie lors de l'établissement du certificat sera obligatoire.

Nous désirons aussi que SYSTOLE garde une trace du numéro de Cerfa utilisé pour l'animal. En effet, même si il n'est pas obligatoire à l'impression (ce numéro est déjà attribué et imprimé en haut à droite des formulaires), il peut nous être utile pour l'établissement de duplicata en cas de perte des documents de l'animal par le propriétaire.

Pour l'impression en format A4, nous choisissons d'utiliser les Cerfas founis par les éditions du Point Vétérinaire, et la mise en page sera réalisée par rapport à leur tabulation.

Lors de la vaccination, il est nécessaire ici de distinguer deux cas de figure :

- le certificat établi est un certificat de primo vaccination : dans ce cas, seuls le numéro de Cerfa, le numéro de lot du vaccin et le laboratoire fournisseur sont nécessaires comme informations. Une seule mise en page est utilisée.

- le certificat établi est un certificat de rappel : aux informations suscitées s'ajoutent le numéro de Cerfa et la date de vaccination de l'année précédente, la nature du Cerfa de l'année précédente (Rose ou Bleu), et le nom du vétérinaire sanitaire ayant réalisé la vaccination l'année précédente. En fonction du type de Cerfa de l'année précédente (Rose ou Bleu), deux mises en page différentes peuvent être utilisées.

Au total, donc, trois états devront être créés : un pour les certificats de primo vaccination, un pour les certificats de rappel suite à une primo vaccination, et un pour les certificats de rappel suite à un rappel.

Enfin, pour éviter toute manipulation fastidieuse, en fonction des choix réalisés dans le type de Cerfa de l'année en cours et précédente, un code Visual Basic® devra déterminer quel type de mise en page utiliser pour l'impression.

2) Réalisation des états d'impression des Cerfas

Les données saisies dans le formulaire de consultation vaccinale sont enregistrées, comme indiqué dans le MLD, dans une table nommée « Vaccinations ».

Cependant, l'établissement des Cerfas Rage nécessite de faire appel à des données issues de plusieurs tables différentes (Clients, Animaux, Vaccinations). Pour cette raison, nous décidons de créer une Requête nommée « Vaccinations : Cerfa » qui assurera le regroupement des différentes informations nécessaires. Les états que nous allons créer se référeront à cette Requête.

Le problème suivant est de pouvoir sélectionner l'enregistrement exact à imprimer. Il serait fâcheux d'imprimer le certificat du Chien « B » appartenant à Monsieur « Y », lorsque c'est le Chien « A » de Monsieur « X » qui vient en consultation.

Afin de régler facilement cet écueil, nous décidons d'utiliser le principe déjà présenté plus haut des sous-formulaires. Access® permet d'inclure dans un état un sous-état, relié à l'état principal par une clé.

Ainsi, le principe est de créer trois états indépendants, possédant un champ invisible qui récupère la clé de la vaccination en cours. Il suffit alors d'inclure un sous-état particulier relié à l'état parent par cette clé. C'est ce sous-état qui contient les différentes zones de textes tabulées en fonction du Cerfa utilisé.

La mise en page a été réalisée grâce aux règles situées dans la fenêtre du mode création de l'état. Celles-ci ont permis de positionner les zones de texte en première approche par rapport aux

postions vides sur les Cerfas. Pour terminer, un affinement successif est réalisé à la main si la position des zones n'est pas parfaite.

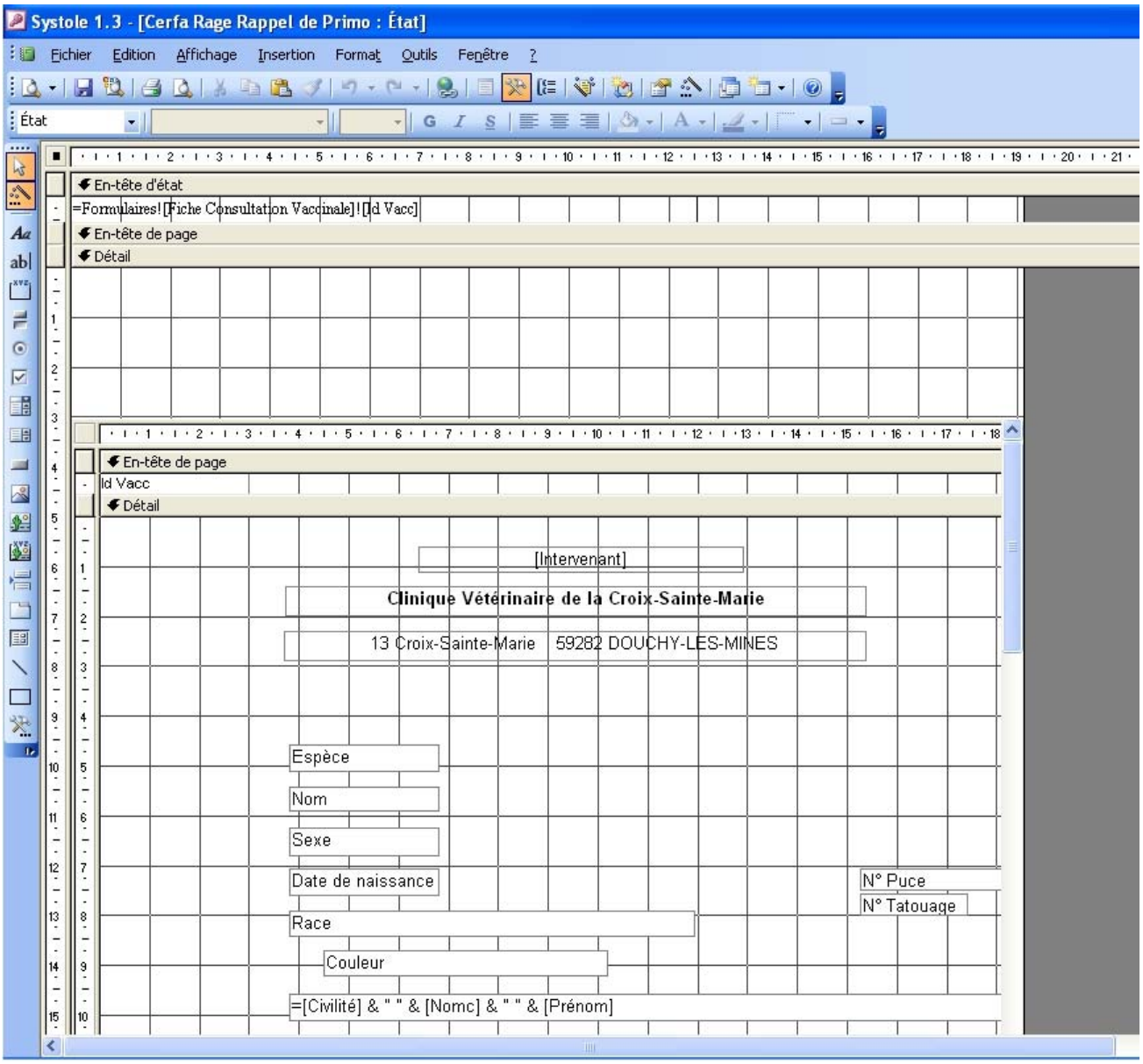
Les champs de date de validité de la vaccination sont obtenus par calcul : il suffit d'ajouter le nombre de jours voulus à [Date vacc] (date de vaccination). Ainsi, pour le certificat de primo vaccination, l'injection est valable de [Date vacc]+31 à [Date vacc]+365.

Nous créons donc trois états et trois sous états :

- Un état « Cerfa Rage Primo » incluant un sous-état « Sous-état : Rédaction Rage Primo »
- Un état « Cerfa Rage Rappel de Primo » incluant un sous-état « Sous-état : Rédaction Rage Rappel de Primo »
- Un état « Cerfa Rage Rappel de Rappel » incluant un sous-état « Sous-état : Rédaction Rage Rappel de Rappel »

La figure 24 montre l'état « Cerfa Rage Rappel de Primo » en mode création.

Figure 24 : Copie de l'écran de création de l'état « Cerfa Rage Rappel de Primo »



3) Réalisation du code et de l'interface permettant le routage des états

Maintenant que les états ont été créés, il ne reste plus qu'à écrire l'interface et le code qui permettront de sélectionner quel modèle devra être imprimé.

Dans l'écran de consultation vaccinale, nous allons donc mettre en place un onglet consacré à la saisie des informations nécessaires à l'établissement des Cerfas. Cet onglet est présenté en figure 25.

Figure 25 : Copie d'écran de l'onglet de saisie des informations nécessaires à l'établissement du Cerfa Rage

Afin d'obtenir une saisie facilitée, nous choisissons d'utiliser un contrôle de groupe pour le choix du type de Cerfa : cela signifie que seule l'option Primo vaccination ou Rappel peut être sélectionnée. Une valeur numérique correspond à ces deux options : le primo vaccin est associé au nombre « 1 », le rappel au nombre « 2 ». C'est cette valeur numérique qui est enregistrée dans la table de données correspondante.

Les champs de saisie correspondant aux informations du Cerfa de l'année précédente ne sont nécessaires que pour éditer des Cerfas de Rappel. Aussi, dans un souci d'allègement de l'interface et de guide pour l'utilisateur, ces champs sont masqués dans le cas d'un certificat de primo vaccination. La figure 26 montre l'écran lors de saisie d'un primo vaccin, la figure 27 montre l'écran lors de saisie d'un rappel.

Figure 26 : Copie d'écran de l'onglet de saisie des informations nécessaires à l'établissement du Cerfa Rage de primo vaccination

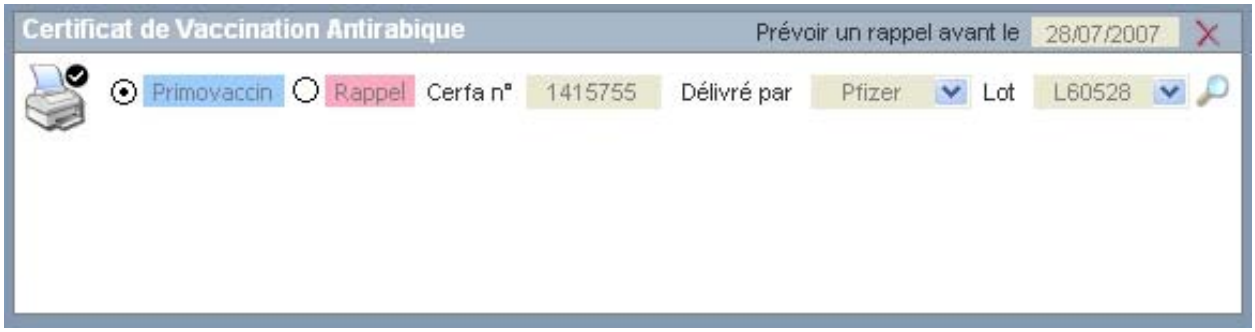


Figure 27 : Copie d'écran de l'onglet de saisie des informations nécessaires à l'établissement du Cerfa Rage de rappel

Le laboratoire fournisseur et le numéro de lot sont choisis dans une liste de choix multiples paramétrables. Le vétérinaire ayant établi le certificat l'année précédente est choisi par liste en menu contextuel.

La programmation de ces éléments a été exposée en première partie.

Une fois les données saisies, il ne reste plus qu'à imprimer le Cerfa. C'est le but de la présence de l'icône d'imprimante  . En cliquant sur cette icône, on appelle la procédure Visual Basic® permettant de sélectionner l'état à afficher. Tout d'abord, la procédure est nommée. Puis, l'enregistrement en cours est enregistré et le formulaire est actualisé, de manière à ce que l'état puisse trouver la référence [Id Vacc].

```
Private Sub open_cerfarab_Click()
On Error GoTo Err_open_cerfarab_Click

DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , acMenuVer70
DoCmd.DoMenuItem acFormBar, acRecordsMenu, 5, , acMenuVer70
```

Ensuite, on définit la variable stDocName en tant que chaîne de caractère. Il sera assigné à cette variable le nom de l'état à afficher en fonction du résultat du test.

```
Dim stDocName As String
```

Vient ensuite la séquence de test : « primorap » correspond au groupe de choix du type de Cerfa, et « cerfatprev » au groupe de choix du type de Cerfa de l'année précédente. Une boucle If … Then, est utilisée : elle permet de soumettre un élément à un test logique.

```
    If primorap = "1" Then stDocName = "Cerfa Rage Primo"
    If primorap = "2" And cerfatprev = "1" Then stDocName = "Cerfa Rage Rappel de Primo"
    If primorap = "2" And cerfatprev = "2" Then stDocName = "Cerfa Rage Rappel de Rappel"
```

Ainsi, si « primorap » a la valeur 1, il s'agit d'un Cerfa de primo vaccination, et l'état ouvert est « Cerfa Rage Primo ». Si « primorap » a la valeur 2 (rappel), on examine la nature du Cerfa de l'année précédente pour déterminer quel état sélectionner.

```
DoCmd.OpenReport stDocName, acPreview
```

Enfin, le document assigné à la variable « stDocName » est ouvert grâce à la méthode OpenReport, en mode « Aperçu avant Impression » (acPreview). La fin du code clôt la procédure et gère l'affichage d'un message en cas d'erreur d'exécution.

```
Exit_open_cerfarab_Click:
    Exit Sub

Err_open_cerfarab_Click:
    MsgBox "Avant d'imprimer le Cerfa, rentrez d'abord les données le concernant."
    Resume Exit_open_cerfarab_Click
End Sub
```

III. UTILISATION DU LOGICIEL : LIMITES ET PERSPECTIVES

A. FINALISATION DU LOGICIEL ET UTILISATION EN SITUATION DE TERRAIN

1) Installation du logiciel et prise en charge réseau et multi-utilisateur

L'application n'est pas indépendante en tant que telle. Il lui faut l'environnement Microsoft Access® pour fonctionner.

Ce choix a été réalisé sur la base du contexte d'évolution continue de l'application : en effet, à chaque nouveau besoin de changement, il nous est possible d'intervenir immédiatement sur la structure et le fonctionnement du logiciel. Ce choix était logique et sûr tant que quelques utilisateurs au grand maximum travaillaient sur SYSTOLE.

Mais le passage en réseau a augmenté et a diversifié le nombre des personnes utilisant l'application.

Il nous a donc fallu sécuriser un peu plus la structure de la base de données, tout en ayant la possibilité d'accéder toujours aussi facilement à la partie programmation.

Cette « sécurisation » utilise deux artifices :

- Tout d'abord, il a été retiré de l'application la possibilité de supprimer des fiches clients ou animaux. Ceci élimine déjà la possibilité de perdre accidentellement des données précieuses. La suppression ne peut être réalisée que par le concepteur, en éliminant directement de la table des données l'enregistrement concerné.
- Ensuite, l'écran de gestion de la base de données est masqué dès le démarrage de l'application grâce à un code exécuté au chargement. Cela évite que les utilisateurs manipulent par erreur les données, les requêtes, ou la structure des formulaires ou des états.

La fonction demarrage_systole écrite en VBA assure cette charge. Elle est placée dans un module de code nommé « Démarrage ». Ce code est exécuté au démarrage par une macro nommée « autoexec ». Access® est conçu pour démarrer d'office toute macro nommée « autoexec » dès son chargement. Voici le code de la fonction de démarrage qui permet en outre de charger les modules « ChargerMotsClés » et « ChargerIntervenant » pour la création des menus contextuels des intervenants et des mots clés :

```
Option Compare Database
Option Explicit

Function demarrage_systole()
On Error Resume Next

' masque la fenêtre de la base de donnees
    DoCmd.SelectObject acTable, , True
    DoCmd.RunCommand acCmdWindowHide

'cache la barre d'état
    Application.SetOption "Afficher Barre d'état", False

'Charge les mots clés (fonction du module "Menu hiérarchique")
    ChargerMotsClés
    DoEvents 'pause

'Charge les intervenants (fonction du module "Menu hiérarchique")
    ChargerIntervenant
    DoEvents 'pause

'ouvre le formualire d'accueil
    DoCmd.OpenForm "Démarrage"

'If Err.Number <> 0 Then MsgBox "Erreur n°" & Err.Number & " au cours du chargement de la procédure
de démarrage." _
                    & vbCrLf & vbCrLf & Err.Description, vbExclamation, "Erreur au démarrage..."
End Function
```

2) Tests lors du développement

A chaque phase du développement de SYSTOLE, et à chaque ajout ou modification d'un module, nous effectuions des tests destinés à essayer de prendre en compte toutes les situations de saisie.

Ces tests nous ont permis :

- de mettre en place à chaque mise à jour une version du logiciel exempte de bugs majeurs
- de détecter les moments clés de la saisie, là où une erreur relationnelle pouvait arriver (par exemple, une consultation qui n'enregistrerait pas le numéro de l'animal auquel elle correspond)
- de réaliser une interface conviviale ayant pour but de guider l'utilisateur dans sa saisie.

Ces tests consistaient simplement à utiliser la nouvelle fonction proposée :

- d'abord dans un contexte normal pour apprécier la facilité et la fluidité d'utilisation,
- puis dans un contexte anormal, pour apprécier la sécurité du système.

Ainsi, SYSTOLE propose autant que faire se peut une sécurisation et une échappatoire possible pour l'utilisateur en cas d'erreur de manipulation.

3) Tests en grandeur nature

a) Utilisation en poste unique de l'application

Les tests grandeur nature ont simplement été réalisés lors de la mise en place du logiciel dans notre clinique. Il a été décidé une période d'essai de un an à compter de janvier 2005 où l'utilisation du logiciel serait limitée à la saisie des cas médicaux, sans utilisation de la fonction d'édition des Cerfas Rages et des relances vaccinales (fonctions par ailleurs assurée depuis plusieurs années au sein de notre structure par Winvet®, logiciel conçu également sous Microsoft Access® par le Dr Jean-Marc LAUWERIE).

Cette période d'essai en situation réelle a permis :

- de corriger quelques bugs mineurs (mises à jour de listes d'affichage ne s'effectuant pas à la fermeture d'un formulaire par exemple)
- de faire évoluer l'application bien plus vite par les propositions d'amélioration données par les confrères.

Au bilan, l'année 2005 fut profitable à l'application qui se montra rapidement indispensable, en facilitant la transmission des informations pour le suivi des cas.

En janvier 2006, le module d'enregistrement des vaccins et d'impression des Cerfas Rage lui est adjoint, ainsi qu'un module de suivi du poids.

b) Utilisation en réseau

Conjointement à ce passage à l'enregistrement des vaccinations, SYSTOLE est également passé d'une version monoposte à une version réseau.

Actuellement, l'application fonctionne sur un réseau composé de deux ordinateurs fixes, parfois complété d'un ordinateur portable. L'architecture est basée sur un réseau WIFI en configuration Ad Hoc, c'est-à-dire que chaque poste se connecte l'un à l'autre pour former la trame du réseau, sans qu'une distinction Client / Serveur soit réalisée.

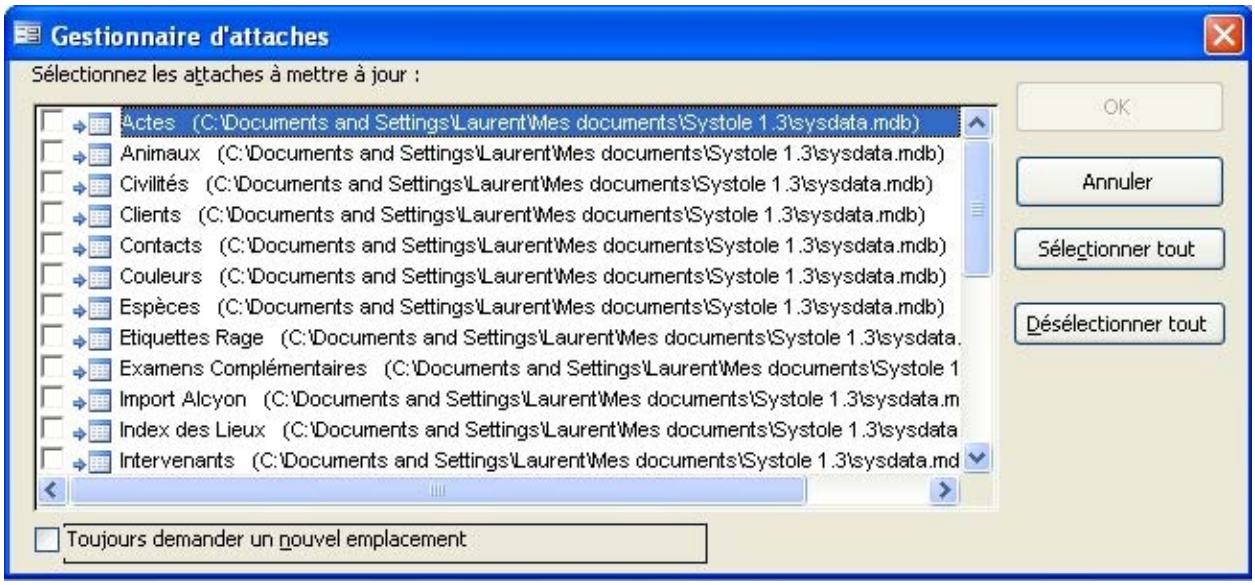
Un ordinateur central joue le rôle de « serveur ». Sur son disque dur se trouve un dossier « Partage Réseau », qui stocke la partie back-end de l'application. Chaque poste de travail dispose en plus sur une zone non partagée du disque de la partie front-end qui est reliée au back-

end via le réseau. Ainsi, chacun peut travailler simultanément sur les données sans interférer avec l'activité des autres postes.

Deux points importants sont à soulever :

- La liaison aux données est effectuée à l'installation de l'application grâce au Gestionnaire de tables liées d'Access®. Ce module permet d'indiquer au logiciel où se trouvent les données qui doivent être exploitées par la partie front-end de l'application (voir figure 28).

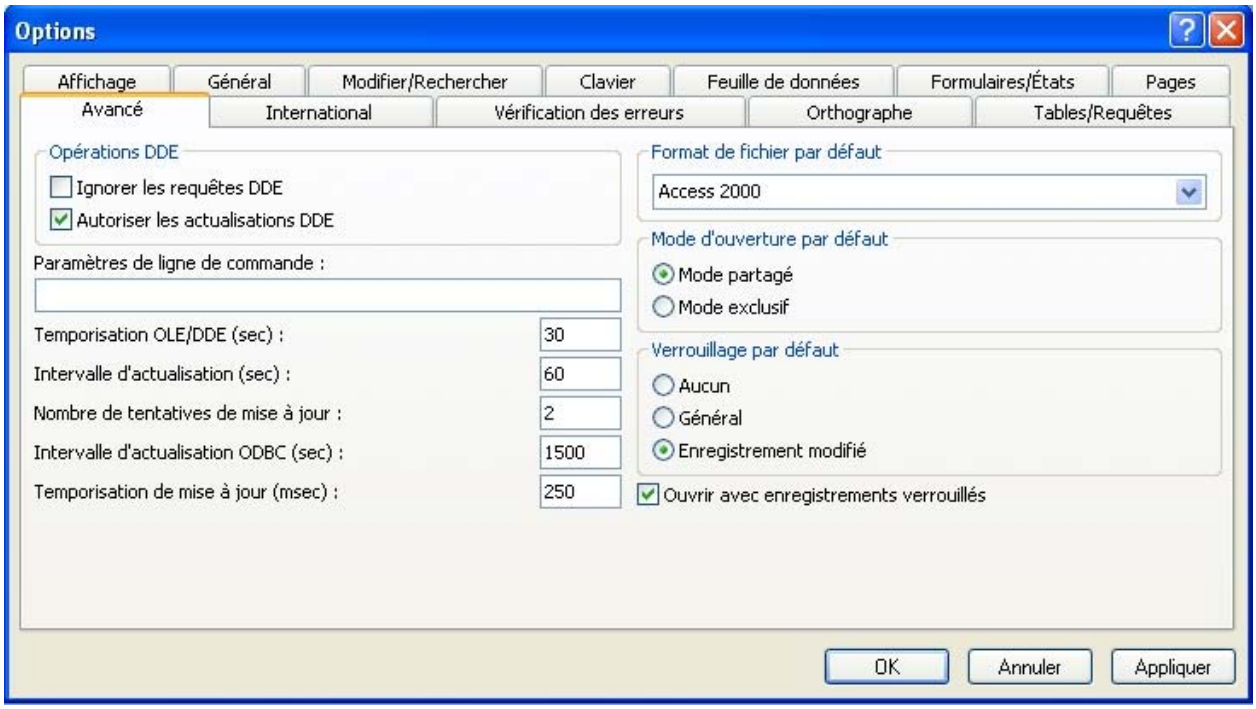
Figure 28 : Copie d'écran du Gestionnaire d'attache des tables liées



- Lors de l'installation de SYSTOLE en réseau, il faut être sûr qu'un enregistrement ne puisse pas être modifié par deux personnes en même temps, auquel cas des données pourraient être perdues. Dans l'espace de configuration d'Access®, une fonction de protection des enregistrements en réseau existe et doit être configurée pour éviter ce désagrément (voir figure 29). Ainsi, pour le « Verrouillage par défaut », il existe plusieurs possibilités. L'option « Aucun » : ici, en cas de conflit de modification, le dernier utilisateur a le pouvoir de décider ou non si ses modifications sont à sauvegarder, un message d'avertissement l'informe. L'option « Général » est non recommandée car si un utilisateur modifie un enregistrement d'une table, il bloque l'accès à la table complète pour les autres utilisateurs du réseau. Enfin, l'option « Enregistrement modifié » : l'enregistrement en cours de modification est bloqué par l'utilisateur auteur de la modification tant qu'il n'aura pas été enregistré. Les autres utilisateurs devront attendre la

fin de la modification pour accéder à cet enregistrement. Ce mode de verrouillage est le plus intéressant car il évite que deux utilisateurs ne modifient en même temps le même enregistrement. C'est cette dernière possibilité qui a été choisie pour l'utilisation de SYSTOLE en réseau.

Figure 29 : Copie d'écran de l'onglet « Avancé » des Options d'Access



c) Mise en place d'une méthode de sauvegarde des données

En situation de terrain, il est impératif que le back-end puisse être sauvegardé afin de ne perdre aucune donnée. En effet, il serait catastrophique pour l'activité clinique de perdre tout le fichier client, les animaux, les vaccinations et les consultations afférentes.

Ainsi, SYSTOLE intègre une possibilité de sauvegarde sous la forme de la création d'une archive Zip du fichier sysdata.mdb. Sur l'écran de Fiche Client, l'icône  placée en bas à droite permet, lorsqu'elle est cliquée, de lancer la procédure de sauvegarde. Voici le code attaché à l'événement « Click » :

```
Private Sub savedata_Click()
On Error GoTo Err_savedata_Click

Dim Msg, Style, Title, Response
Msg = "Avant de sauvegarder la Base de Systole, assurez-vous que le périphérique de stockage soit inséré et prêt."    ' Définit le message.
Style = vbOKCancel + vbQuestion    ' Définit les boutons.
Title = "Quitter et Sauvegarder"    ' Définit le titre.
' Affiche le message.
Response = MsgBox(Msg, Style, Title)

If Response = vbOK Then
Dim z As ZipClass

Set z = New ZipClass
z.AddFile Me.ssformsave!Datapath
z.WriteZip Me.ssformsave!Savepath & "backup" & Format(Date, "ddmmyyyy") & ".zip", True
Set z = Nothing

DoCmd.Quit
Else: DoCmd.CancelEvent

End If

Exit_savedata_Click:
    Exit Sub

Err_savedata_Click:
    MsgBox Err.Description
    Resume Exit_savedata_Click

End Sub
```

Le fichier de sauvegarde ainsi obtenu est nommé backupjjmmaaaa.zip, où « jjmmaaaa » représente la date courante du moment où est réalisée la sauvegarde.

Pour fonctionner, cet outil nécessite la présence de deux modules de classe et d'une Dynamic Link Library (DLL). Ces trois fichiers doivent être placés dans le répertoire courant de Systole.

Un module de classe est un module comprenant du code Visual Basic® pouvant être appelé de manière externe via une autre partie de code. Un module de classe contient généralement tout le code correspondant à l'exécution d'une fonction particulière.

Pour notre exemple, nous trouvons dans le répertoire de SYSTOLE : zlib.dll, et les deux modules de classe : ZipFile.cls et ZipClass.cls (voir figure 30). Les modules de classe doivent de plus être déclarés dans la section « Modules » d'Access® pour pouvoir être utilisés (voir figure 31).

Figure 30 : Copie d'écran du répertoire de SYSTOLE

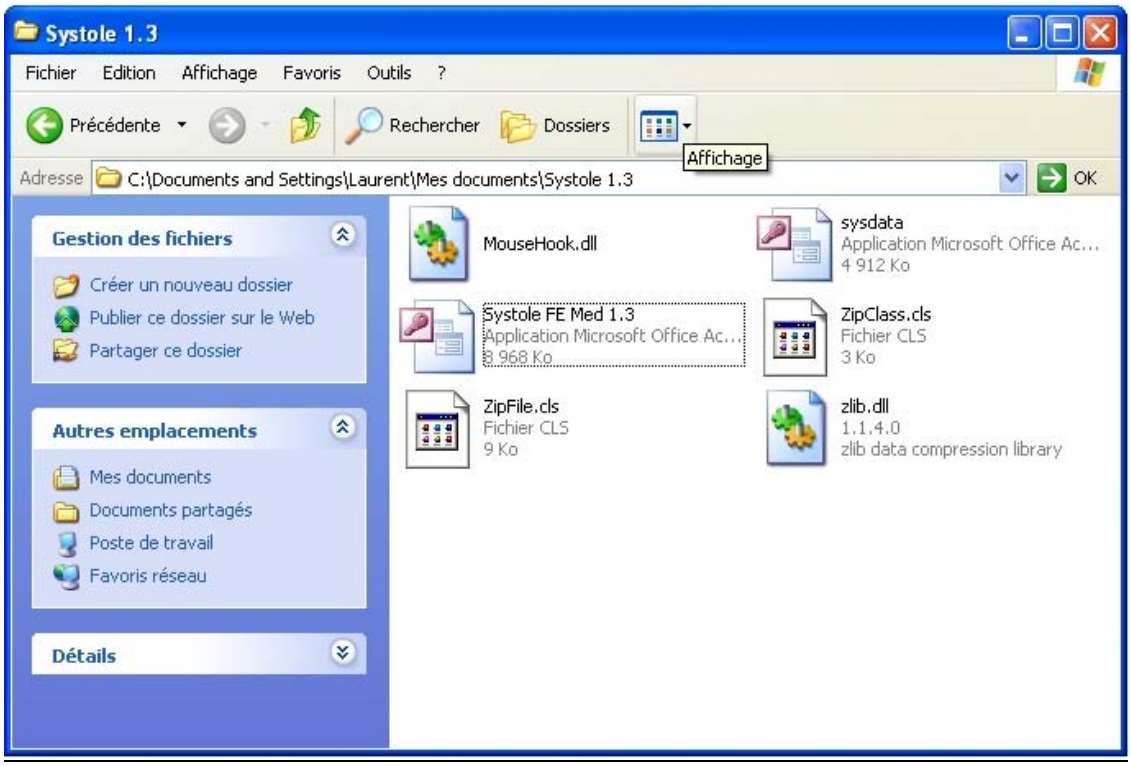
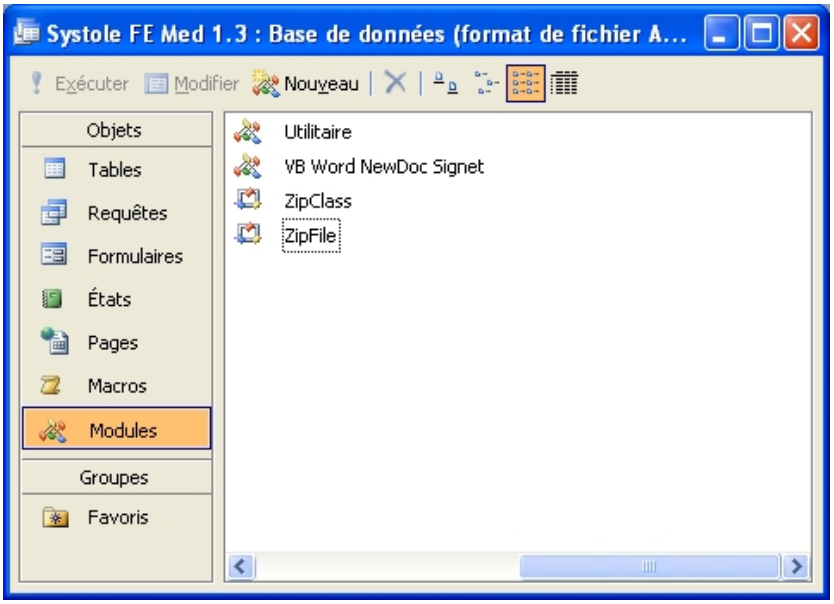


Figure 31 : Copie d'écran de l'onglet des Modules, montrant les modules de classes ZipClass et ZipFile déclarés



B. LES LIMITES DE LA VERSION ACTUELLE DE L'APPLICATION

1) Limites propres du logiciel

A l'heure actuelle, SYSTOLE convient bien à l'utilisation courante d'une structure vétérinaire de petite à moyenne taille (de deux à quatre vétérinaires travaillant en même temps, plus deux à trois postes d'ASV), situation de travail pour laquelle il est pleinement fonctionnel.
Par contre, sa sécurisation et la gestion plus pointue d'un environnement Multi-Utilisateur deviendra indispensable à l'avenir, car c'est encore son point faible.
Dans sa version 1.3, SYSTOLE ne permet pas d'effectuer de recherche sur les mots-clés attribués aux photographies. Cependant, cette limite est relative puisqu'il suffit de dupliquer et de modifier légèrement l'écran de recherche des clients et des animaux afin de combler cette lacune.
Enfin, il devient de plus en plus clair que le marketing des services se dirige vers une proposition de tout un panel d'options qui pourront être choisies à l'avenir par les propriétaires de nos patients. Il faut ainsi tenir compte :

- de la démocratisation de l'Internet
- de l'avancée considérable de la place des aliments physiologiques et diététiques dans nos cliniques et de la demande croissante en conseil alimentaire
- de l'évolution démographique de la population canine et féline (vieillissement)
- du développement des mutuelles pour animaux de compagnie
- du temps que peut accorder chaque propriétaire à son animal, et de l'utilité des rappels par courrier ou e-mail (vermifuge, anti-parasitaire externe, courbe de glycémie…)

Dans ces conditions, il devient important de pouvoir emmagasiner un maximum d'informations concernant nos patients et leurs maîtres, de manière à pouvoir cerner leurs besoin et leur proposer les services qui leurs sont les plus adaptés.
Cette évolution de notre profession et de notre manière d'exercer doit être suivie par SYSTOLE.

2) Limites liées au concepteur

Les limites d'évolution de SYSTOLE sont également liées au concepteur du logiciel. En effet, il nous a fallu déjà beaucoup de temps afin d'acquérir les savoirs et les compétences nous ayant

permis de concevoir une application stable et fonctionnelle, et cela conjointement à notre activité professionnelle.
C'est surtout donc la contrainte de temps qui ralentit l'évolution du logiciel.
Notre progression dans l'utilisation d'Access® et dans la programmation en Visual Basic® a été malgré tout grandement facilitée par l'existence sur Internet d'une grande communauté de développeurs.
Denier point : dans une logique de développement personnalisé en temps réel, la vision qu'a le développeur de son application vient souvent en opposition des besoins exprimés par le groupe des utilisateurs, ce qui nécessite souvent, telle Pénélope sur son ouvrage, de défaire ce qui avait pris bien du temps à être fait. Mais l'évolution harmonieuse et l'adéquation de SYSTOLE à des besoins de terrain sont à ce prix.

3) Limites liées aux utilisateurs

Enfin, les limites du logiciel tiennent également, mais dans une moindre mesure, aux utilisateurs.
Au début de l'utilisation de SYSTOLE, ses fonctions étaient simples, voire même simplistes. Puis, se complexifiant et gagnant en nombre d'utilisateurs, son fonctionnement en est devenu moins évident.
Pourtant, nous n'avons pas réalisé jusqu'à récemment de manuel décrivant toutes les possibilités offertes.
Il est plus que probable que l'utilisation de SYSTOLE serait plus qu'efficace si tous les utilisateurs disposaient d'un mode d'emploi auquel se référer.
Cette situation n'est pour l'instant pas catastrophique, mais il convient d'y remédier dès maintenant dans l'optique de l'évolution de l'application.
Ainsi, nous avons débuté la rédaction d'une notice technique, portant essentiellement sur les nouvelles fonctions disponibles lors des mises à jour.
A titre d'exemple, nous fournissons en annexe 3 l'aide concernant le module de suivi du poids.

C. PERSPECTIVES D'EVOLUTION

1) Amélioration de la sécurité et environnement Multi-Utilisateur

A l'avenir, nous devrons certainement faire face à une augmentation du nombre potentiel d'utilisateurs de SYSTOLE. D'autre part, actuellement, aucune restriction d'accès à l'une quelconque des parties de l'application n'est possible selon le statut de l'utilisateur.

L'amélioration de la sécurité ne pourra passer que par une gestion des utilisateurs et de leurs permissions d'accès.

Access® permet de facilement mettre en place ce type de restrictions d'accès, grâce à la possibilité de créer des utilisateurs et des groupes d'utilisateurs (figure 32) [7].

Figure 32 : Copie de l'écran des utilisateurs et des groupes dans Access

Ainsi, pour chaque personne accédant à la base de données, une identification avec un nom de connexion et un mot de passe peut-être délivré. En fonction du statut de l'utilisateur, des autorisations d'accès sont délivrées par l'Administrateur de la BD.

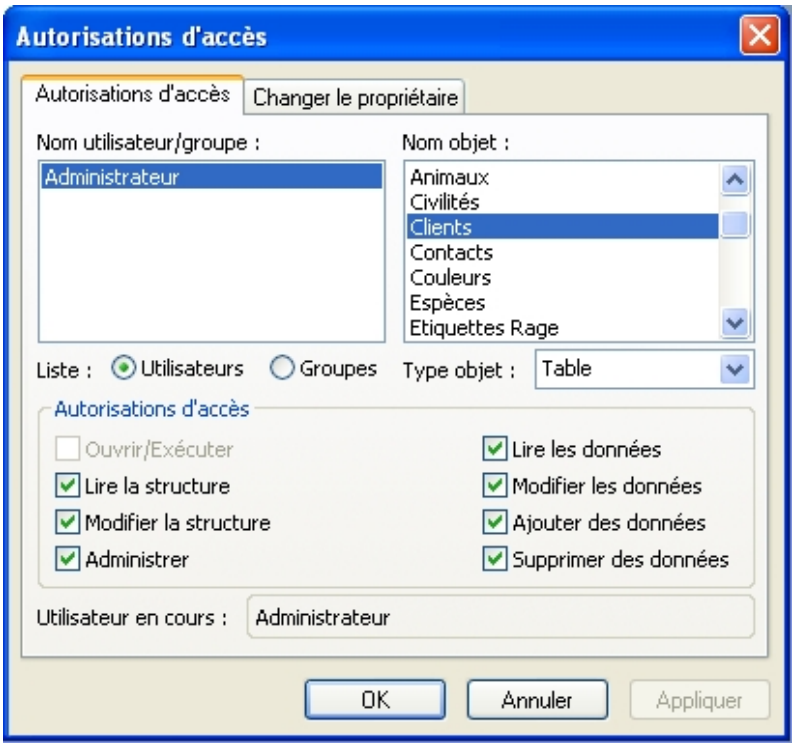
On distingue celles portant sur la structure même des objets (conception) :

- **Ouvrir/exécuter** : ne concerne que les formulaires : l'utilisateur peut ouvrir le formulaire et exécuter le code lié au formulaire

- **Lire la structure** : l'utilisateur peut observer la structure de la table ou de tout autre objet mais ne peut pas la modifier
- **Modifier la structure** : l'utilisateur peut ajouter de nouveaux champs, en supprimer voire même supprimer la table mais ne peut pas accorder d'autorisations à d'autres utilisateurs
- **Administrer** : mêmes droits que « modifier la structure » avec en plus le droit d'accorder des autorisations d'accès à d'autres utilisateurs.

L'autre catégorie concerne l'utilisation normale de l'objet : lire, modifier, ajouter ou supprimer les données (figure 33).

Figure 33 : Copie de l'écran des autorisations d'accès des utilisateurs et des groupes dans Access



Malgré l'aspect intéressant de cette gestion des utilisateurs, son utilisation requiert néanmoins une certaine maîtrise du logiciel.

En effet, les informations sur les utilisateurs et les groupes sont rassemblées sous l'appellation **groupe de travail.** Physiquement, ce groupe de travail correspond à l'existence d'un fichier portant l'extension « .mdw ». Par défaut, il existe un fichier nommé « system.mdw » qui reprend

les informations concernant les utilisateurs configurés au démarrage d'Access®. Il ne contient qu'un seul utilisateur nommé « *administrateur* » rattaché au groupe d'utilisateurs *Administrateurs.* Cet utilisateur n'a aucun mot de passe. C'est pourquoi un utilisateur faisant implicitement référence à « system.mdw » a tous les pouvoirs (il peut créer de nouvelles bases, ajouter, modifier ou supprimer tout ce qu'il veut). Pour garder cette liberté, il ne faut donc surtout pas gérer les utilisateurs en utilisant « system.mdw » faute de quoi, les bases de données actuelles ou futures pourraient ne plus fonctionner.

Quelles sont les liens unissant une base Access et les fichiers « .mdw » ? Lorsque Access® est démarré, le programme lit les informations du fichier « .mdw » « actif ». En ouvrant ensuite une base (ayant l'extension « .mdb »), Access® utilise les informations contenues dans le fichier « .mdw ». S'il trouve un utilisateur unique avec un nom autre qu'*administrateur* ou différents utilisateurs, il ouvre une boîte de dialogue d'ouverture de session. Ensuite, à chaque fois que l'utilisateur ouvre un objet (table, formulaire, requête,…) il vérifie les autorisations d'accès de l'utilisateur sur cet objet. Ainsi, si les utilisateurs sont gérés au niveau d'un fichier « .mdw », les autorisations d'accès aux objets de la base de données sont elles, en revanche, prises en charge par la base de données (fichier « .mdb »)

Ceci dit, si on ne peut pas utiliser, par prudence, le fichier « system.mdw » peut on en utiliser un autre ? Pour gérer la sécurité, il faut effectivement créer d'abord un groupe de travail. Une fois le groupe de travail créé, il faut **joindre** ce groupe de travail. Lorsqu'un groupe de travail est joint (la procédure est expliquée plus bas), toutes les bases de données ouvertes feront référence à ce groupe de travail. Les conséquences **après jonction** sont importantes :

1. les bases nouvellement créés comporteront les utilisateurs du groupe de travail actif (le retour à une utilisation du groupe de travail par défaut rendra ces bases totalement inutilisables).

2. les bases anciennes ouvertes sous le nouveau groupe de travail ne fonctionneront plus (ou risqueront de ne plus fonctionner) avec le groupe par défaut « system.mdw » (c'est pourquoi avant de sécuriser une base, il faut en faire une copie de sécurité).

Donc, **PAR PRECAUTION**, lorsque le travail sur une base sécurisée est terminé, il faut joindre le groupe de travail par défaut « system.mdw », cette opération ayant pour but de restaurer

l'usage normal d'Access et d'éviter ainsi d'ouvrir une base non sécurisée avec n'importe quel fichier de groupe de travail.

Cette lourdeur de gestion du groupe de travail pourra éventuellement trouver sa solution :

- soit dans une programmation en Visual Basic®, auquel cas la tâche serait automatisée à l'ouverture et à la fermeture d'Access.

- Soit dans la réalisation d'une application indépendante, n'utilisant plus Access pour s'exécuter. Il existe pour les applications réalisées un programme nommé *runtime* permettant un lancement de manière autonome, même si Access n'est pas installé sur la machine.

Au stade où en est le développement, aucune décision n'a encore été prise, les autorisations d'accès n'étant toujours pas définies pour les différents utilisateurs, mais il va sans dire que cet axe de développement sera l'un des points important de l'expansion de SYSTOLE.

2) Développement de la version 2.0

Pour répondre aux besoins énoncés plus haut en terme de sécurité et de volume d'informations, nous avons décidé de démarrer le développement de la version 2.0 de SYSTOLE.

Cette version possèdera une interface remaniée qui se veut plus synthétique.

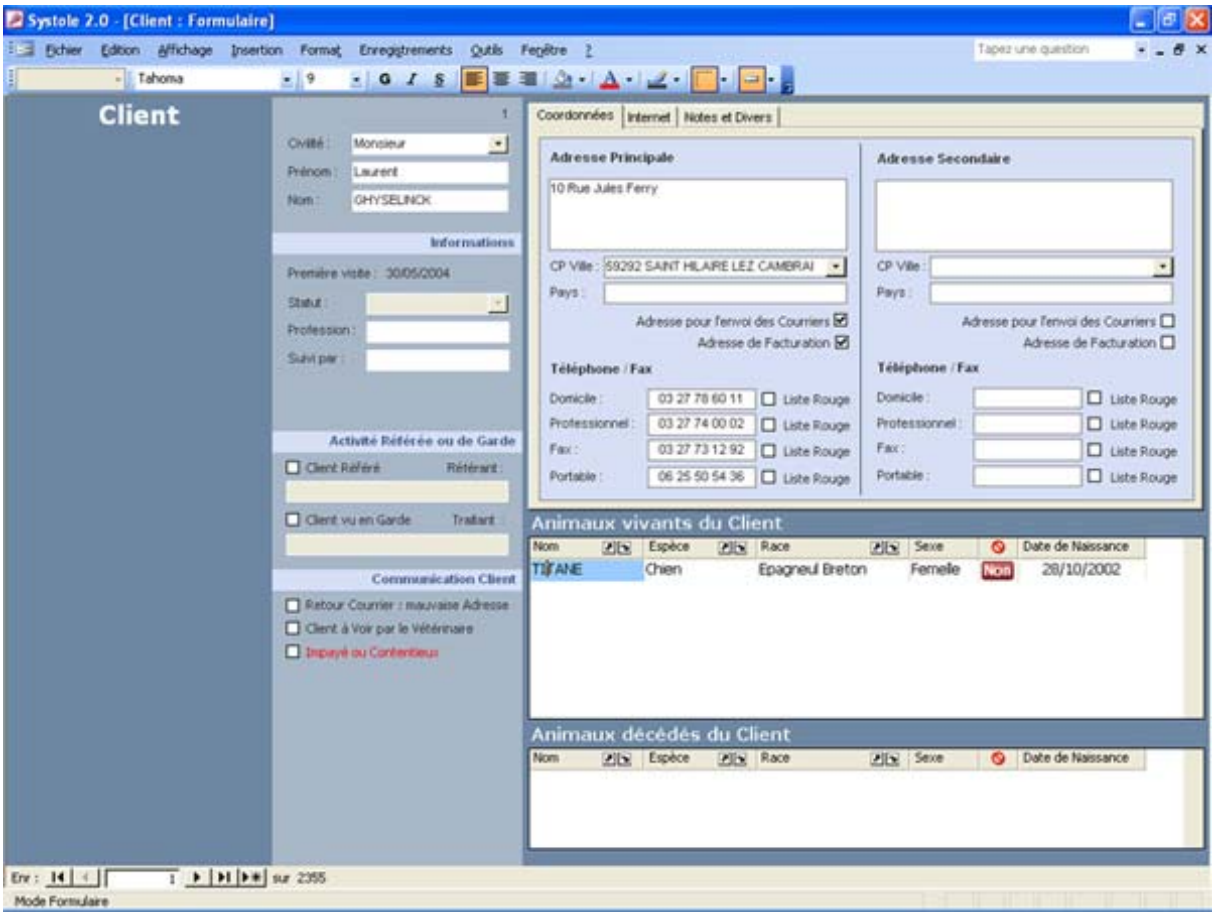
Un grand nombre de fonctions de saisie automatique sera mis en place également (utilisation de mots-clés dans le compte-rendu clinique, de phrases et d'expression types dans l'élaboration des ordonnances).

La gestion des passeports européens et de l'administration de produits stupéfiants, oubliés dans les versions 1.x sera également présente.

L'ouverture vers les nouveaux moyens de communication est enfin une priorité de cette version de l'application, le but étant de proposer aux clients des rappels réguliers par courrier électroniques ou mini-messages sur téléphone portable, de points de prévention tels que les traitements anti-parasitaires internes ou externes, les vaccins, le brossage des dents…

A titre d'exemple, la figure 34 propose une capture de l'écran rassemblant les informations concernant le client. On pourra constater que dans le principe, les grandes lignes de SYSTOLE 1 sont conservées.

Figure 34 : Copie d'écran de la Fiche Administrative Client prévisionnelle de Systole 2.0



3) Mise à disposition des confrères via Internet

Pour clore ces perspectives d'évolution de SYSTOLE, nous envisageons comme possibilité la mise en ligne de l'application ainsi que de son code source, sous forme de freeware.

Le but en serait :

- de créer une véritable communauté d'utilisateur, génératrice elle-même de demande et de proposition en terme d'évolution

- de rallier à ce projet des développeurs vétérinaires pour enrichir SYSTOLE et en faire un véritable logiciel de Gestion de Clinique Vétérinaire Open Source.

Si ce projet devait se réaliser, il passerait de manière obligatoire par l'utilisation d'un moteur de SGBDR et d'un langage de programmation également gratuit et open source. Une réflexion menée sur la question nous permet d'ores et déjà d'envisager l'utilisation du couple PHP – MySQL pour réaliser ce projet. De plus, d'autres vétérinaires informaticiens ont déjà réalisés des travaux axés sur ce type d'association [2].

Freeware : consiste à mettre gratuitement à la disposition du public une version complète et fonctionnelle d'un logiciel, avec ou sans ses sources.

CONCLUSION

Notre objectif initial était de créer une application de suivi clinique simple pour une Clinique Vétérinaire canine. Elle devait remplir certaines fonctions, mais sa finalité était d'apporter une amélioration dans le suivi clinique des animaux de la clientèle, sans être trop gourmande en temps de saisie et en permettant d'automatiser certaines tâches.

Après plus de trois ans de formation autodidacte en Base de Données et langage Visual Basic®, et de développement concomitant, et plus d'un an de test en grandeur nature au sein d'une Clinique, nous avons obtenu une application à l'interface agréable et ergonomique, au fonctionnement fluide, qui a su faire sa place et se montrer indispensable quotidiennement.

Mais le développement ne s'arrête pas ici. Le fait de pouvoir modifier l'application à loisir et en temps réel permet la correction des défauts lorsqu'ils se présentent, et l'utilisation par plusieurs personnes différentes fait inévitablement surgir des besoins supplémentaires.

SYSTOLE n'est donc pas une application aboutie, mais en devenir permanent, qui ne demande qu'à s'enrichir et à évoluer.

Ce manuel de thèse explique notre démarche dans la réalisation d'un tel travail. Il sera utile, nous l'espérons, à tous ceux qui voudront, à leur tour, s'attaquer à un projet similaire ou rejoindre peut-être le projet SYSTOLE.

BIBLIOGRAPHIE

1. Access. Microsoft Corporation, **2003**.

2. CODD EF., A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 1970, **13,** 377-387

3. GAVIGLIO V. *Création d'un outil informatique d'archivage et d'aide à la réalisation des cas cliniques à l'ENVA*. Thèse Méd. Vét., Alfort, **2004**, 74p.

4. GRUAU C. Conception d'une base de données. *In* : *Cyril Gruau – MERISE – Club d'entraide des développeurs francophones.* [en-ligne], **17 octobre 2005** (modifié le 13 juillet 2006), : Le Club des Développeurs [http://cyril-gruau.developpez.com/uml/tutoriel/ConceptionBD/] (consultée le 2 août 2006).

5. KIMBALL S., MATTIS P., GNU Image Manipulation Program (GIMP) 2.2.11, **2005**

6. Le Club des Développeurs. *Les meilleurs cours et tutoriels Access – Club d'entraide des développeurs francophones* [en-ligne], [http://access.developpez.com/], (consulté le 22 septembre 2004).

7. REDONNET F. *Bases Access en Réseau – présentation du thème* [en-ligne], Mise à jour le **24 Juin 2005** [http://frederic.redonnet.free.fr/cours/formationaccess/perfectaccess/multi1.htm], (consulté le 6 Juillet 2005).

8. TARDIEU H., ROCHFELD A., COLLETI R. *La méthode Merise - Tome 1 : Principes et outils*. Paris : Editions d'organisation, **1983**, 328 p.

9. TOMA B. *La Rage* .Polycopié. Ecoles Nationales Vétérinaires françaises, Unité Pédagogique de Maladies Contagieuses. **2006**, 73p.

10. Wikipedia. *Base de Données - Wikipedia* [en-ligne], [http://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es], (consulté le 2 Avril 2006).

11. Wikipedia. *Structured Query Language - Wikipedia* [en-ligne], [http://fr.wikipedia.org/wiki/SQL], (consulté le 6 Avril 2006).

12. Wikipedia. *Système de Gestion de Base de Données - Wikipedia* [en-ligne], [http://fr.wikipedia.org/wiki/Syst%C3%A8me_de_gestion_de_base_de_donn%C3%A9es], (consulté le 25 Avril 2006).

13. YAHOO! GROUPES. *access-Fr : Développer sous Access* [Groupe de discussion en-ligne], Création le **4 juin 2000** [http://fr.groups.yahoo.com/group/access-Fr/], (consulté le 12 Janvier 2006).

ANNEXES

Annexe 1 : Exemples de commandes en langage SQL

Exemples d'ordres DDL

Création de table :

```
CREATE TABLE TABLE1 (Champ1 Integer, Champ2 integer, Champ3 Date, Champ4 Date);
```

Modification de table :

```
# On ne met des parenthèses que lorsque l'on ajoute une colonne
ALTER TABLE TABLE1 ADD COLUMN (Champ5 Integer);
ALTER TABLE TABLE1 MODIFY COLUMN Champ5 Long;
ALTER TABLE TABLE1 DROP COLUMN Champ5;
```

Suppression de table :

```
DROP TABLE TABLE1 ;
```

Exemples d'ordres DML

Consultation d'un enregistrement

```
SELECT {expressions} FROM {tables} WHERE {prédicats} GROUP BY {expressions} HAVING {condition} ORDER BY {expressions};
SELECT (name, service) FROM employees WHERE (statut='stagiaire') GROUP BY (name,service) ORDER BY name;
```

Par exemple, si on a une tables *employes*, contenant 2 champs *nom* et *salaire*, la requête permettant de trouver le nom des salariés touchant plus de 1500€ par mois classés par salaire est la suivante :

```
SELECT nom FROM employes WHERE salaire > 1500 ORDER BY salaire;
```

Exemples de requêtes pour afficher les enregistrements d'une table TABLE1 de clé primaire champ1 non présents dans une seconde table TABLE2 :

```
SELECT * FROM TABLE1  WHERE TABLE1.champ1 NOT IN (SELECT champ2 FROM TABLE2)
```

(Logique d'exclusion : NOT + enumeration IN : À éviter pour de bonnes performances)

```
SELECT * FROM TABLE1 WHERE TABLE1.champ1 IN (SELECT champ1 FROM TABLE2 MINUS SELECT champ2 FROM TABLE2 ) (Logique ensembliste MINUS + enumeration IN )

SELECT * FROM TABLE1 , (SELECT champ1 FROM TABLE2 MINUS SELECT champ2 FROM TABLE2) TMP WHERE TABLE1.champ1 = TMP.champ1 (Logique ensembliste : MINUS + jointure : = )

SELECT * FROM TABLE1 LEFT OUTER JOIN TABLE2 ON TABLE1.champ1 = TABLE2.champ2 WHERE TABLE2.champ2 IS NULL (Logique de jointure complète à gauche et incomplète à droite + restriction : IS NULL) (Syntaxe SQLANSI Compatible Oracle à partir de 9i)
```

```
SELECT * FROM TABLE1, TABLE2 WHERE TABLE1.champ1 = TABLE2.champ2(+) AND TABLE2.champ2 IS NULL (Variation Spécifique Oracle sur la syntaxe de la jointure gauche LEFT OUTER )
```

NB : Dans la mesure du possible on privilégiera les opérateurs d'égalité et ensemblistes aux opérateurs d'exclusion <>, NOT IN qui n'utilisent que très rarement les index

Ajout d'un enregistrement :

```
INSERT INTO employees (name, service) values ('Martin','comptabilité');
```

Ajout de données à partir des informations d'une autre table :

```
INSERT INTO TABLE1 ( champ1, champ2, champ3) Select (Val1, val2, val2+val3/2) from TABLE2;
```

Mise a jour de données:

```
UPDATE employees SET service='accueil' Where name = 'Martin';
```

Mise à jour d'une table à partir des informations d'une autre table :

```
UPDATE TABLE1
SET ( champ3, champ4 ) =
(SELECT val32, val13 FROM TABLE2 WHERE TABLE1.CHAMP1 = TABLE2.CHAMP2)
WHERE ( CHAMP1 ) IN ( SELECT CHAMP2 FROM TABLE2);
```

Suppression d'un enregistrement :

```
DELETE FROM employees WHERE name = 'Alpha';
```

```
'*****Pour Access 2000 il faut :******
'  VisualBasic For Applications
'  Microsoft Access 9.0 Object Library
'  Microsoft DAO 3.6 Object Library
'  Microsoft Office 9.0 Object Library
'  Microsoft Word 9.0 Object Library
'  OLE Automation
'  Microsoft Visual Basic for Applications Extensibility 5.3

Option Compare Database
Option Explicit
Const NomMenu = "Intervenant"
Dim Intervenant
'*************** Barre de progression  *********************
Dim Compteur As Long, TotalIntervenants As Integer, ProgressIntervenants   As
Integer
'compte les intervenants chargés dans le menu
'       le total des intervenants à charger
'       le pourcentage d'intervenants chargés
'*************************************************************


Sub ChargerIntervenant()
On Error GoTo Err_ChargerIntervenant
'Charge le Menu hiérarchique des intervenants contenus dans la table
"Intervenant"
'Cette fonction est lancée au chargement du formulaire d'accueil

'*************** Barre de progression  *********************
DoCmd.OpenForm "ProgressBar"
DoEvents
Compteur = 0
TotalIntervenants = DCount("Index", "Intervenants")
'*************************************************************

Dim Menu As CommandBar
For Each Menu In Application.CommandBars
   If Menu.name = NomMenu Then Menu.Delete
Next Menu
Set Menu = Application.CommandBars.Add(NomMenu, msoBarPopup, , True)
ChargerIntervenant1 0, 1

'*************** Barre de progression  *********************
   DoCmd.Close acForm, "ProgressBar"
'*************************************************************
Set Menu = Nothing

Exit_ChargerIntervenant:
Exit Sub
Err_ChargerIntervenant:
MsgBox "Cher Confrère, " & vbCrLf & _
    "Une erreur s'est produite lors du chargement des mots clés. Il se peut
que la table des mots clés " & _
    "n'ait pas été trouvée. Cliquez sur Configuration et " & _
    "vérifiez à l'onglet [Source des données] que le chemin d'accès à vos
données soit valable." & vbCrLf & vbCrLf & _
```

```
    "Description de l'erreur :" & vbCrLf & _
    Err.Description, vbOKOnly + vbExclamation, _
    "Erreur dans la fonction ChargerMotsClés du module Menu hiérarchique"
Resume Exit_ChargerIntervenant
End Sub


Private Sub ChargerIntervenant1(ByVal iParent As Long, ByVal iMaxDepth As
Long)
On Error GoTo Err_ChargerIntervenant1
Dim SousMenu As CommandBarPopup, OptionMenu As CommandBarControl

If iMaxDepth > 20 Then MsgBox "Vous ne pouvez pas avoir plus de 20 niveaux
dans le menu des mots clés.": Exit Sub
Dim TypeMenu
Dim rs As Recordset
Set rs = Currentdb.OpenRecordset("SELECT * from Intervenants where parent=" &
Str$(iParent) & " ORDER BY Affichage;")
If rs.RecordCount > 0 Then

    rs.MoveLast
    rs.MoveFirst
   Do While Not rs.EOF
   '****************** Barre de progression  **********************
   Compteur = Compteur + 1
   ProgressIntervenants = 100 * Compteur / TotalIntervenants
   Forms!ProgressBar!Pcent = Compteur & " / " & TotalIntervenants
   DoEvents
   'Forms!ProgressBar!txtProgression = ProgressMots & "%"
   'DoEvents
   Forms!ProgressBar!txtProgression.Width = 43 * ProgressIntervenants
'*************************************************************
Dim Childs As Long
    Childs = DCount("Index", "Intervenants", "parent=" & rs!index)
      If Childs = 0 Then
         TypeMenu = msoControlButton
      Else
         TypeMenu = msoControlPopup
      End If

      If rs!Parent <> 0 Then
         Set SousMenu = Application.CommandBars.FindControl(msoControlPopup,
, "index" & rs!Parent)
         Set OptionMenu = SousMenu.Controls.Add(TypeMenu, , , , True)
         If SousMenu.Controls.Count <= 2 Then OptionMenu.BeginGroup = True
      Else
         Set OptionMenu =
Application.CommandBars(NomMenu).Controls.Add(TypeMenu, , , , True)
      End If
      OptionMenu.Caption = rs!Affichage

      If Childs > 0 Then
         OptionMenu.Tag = "index" & rs!index
         Set OptionMenu = OptionMenu.Controls.Add(msoControlButton, , , ,
True)
         OptionMenu.Caption = "Sélectionner " & rs!Affichage
      End If
      OptionMenu.OnAction = "ColleSelection"
      OptionMenu.Tag = rs!Affichage
      ChargerIntervenant1 rs!index, iMaxDepth + 1
      rs.MoveNext
```

```
   Loop
End If
rs.Close
Set rs = Nothing
Set OptionMenu = Nothing
Set SousMenu = Nothing
Exit_ChargerIntervenant1:
Exit Sub
Err_ChargerIntervenant1:
MsgBox "Cher Confrère," & vbCrLf & _
    "Une erreur s'est produite lors du chargement des mots clés." & vbCrLf &
Err.Description, _
    vbOKOnly + vbExclamation, _
    "Erreur dans la fonction ChargerMotsClés1 du module Menu hiérarchique"
Resume Exit_ChargerIntervenant1

End Sub

Function AfficherMenu()
On Error GoTo Err_AfficherMenu

Application.CommandBars(NomMenu).ShowPopup

Exit_AfficherMenu:
Exit Function

Err_AfficherMenu:
MsgBox "Cher Confrère, " & vbCrLf & Err.Description & _
    " dans la fonction AfficherMenu du module Menu Hiérarchique."

Resume Exit_AfficherMenu

End Function

Sub ColleSelection()
  Intervenant = Application.CommandBars.ActionControl.Tag
    End Sub

Sub GetIntervenant()
'Fonction utilisée dans les formulaires pour attribuer un mot clé à un
patient ou une photo.
'Colle le mot clé selectionné dans le contrôle actif du formulaire.
On Error GoTo Err_GetIntervenant

    Intervenant = Null
    Application.CommandBars(NomMenu).ShowPopup
    If Not IsNull(Intervenant) Then
        Screen.ActiveControl = Intervenant
        DoCmd.RunCommand acCmdSaveRecord
    End If
Exit_GetIntervenant:
Exit Sub
Err_GetIntervenant:
MsgBox "Cher Confrère, " & Err.Description & " dans la fonction GetMotCle du
module Menu Hiérarchique."
Resume Exit_GetIntervenant

End Sub
```

```
'*****Pour Access 2000 il faut :******
'  VisualBasic For Applications
'  Microsoft Access 9.0 Object Library
'  Microsoft DAO 3.6 Object Library
'  Microsoft Office 9.0 Object Library
'  Microsoft Word 9.0 Object Library
'  OLE Automation
'  Microsoft Visual Basic for Applications Extensibility 5.3

Option Compare Database
Option Explicit
Const NomMenu = "MenuHierarchique"
Dim MotCle
'*************** Barre de progression  *********************
Dim Compteur As Long, TotalMots As Integer, ProgressMots   As Integer
'compte les mots clés chargés dans le menu
'       le total des mots clés à charger
'       le pourcentage de mots chargés
'*********************************************************


Sub ChargerMotsClés()
On Error GoTo Err_ChargerMotsClés
'Charge le Menu hiérarchique des mots clés contenus dans la table "MotsClés"
'Cette fonction est lancée au chargement du formulaire d'accueil

'*************** Barre de progression  *********************
DoCmd.OpenForm "ProgressBar"
DoEvents
Compteur = 0
TotalMots = DCount("index", "MotsClés")
'*********************************************************

Dim Menu As CommandBar
For Each Menu In Application.CommandBars
   If Menu.name = NomMenu Then Menu.Delete
Next Menu
Set Menu = Application.CommandBars.Add(NomMenu, msoBarPopup, , True)
ChargerMotsClés1 0, 1

'*************** Barre de progression  *********************
   DoCmd.Close acForm, "ProgressBar"
'*************************************************************
Set Menu = Nothing

Exit_ChargerMotsClés:
Exit Sub
Err_ChargerMotsClés:
MsgBox "Cher Confrère, " & vbCrLf & _
    "Une erreur s'est produite lors du chargement des mots clés. Il se peut
que la table des mots clés " & _
    "n'ait pas été trouvée. Cliquez sur Configuration et " & _
    "vérifiez à l'onglet [Source des données] que le chemin d'accès à vos
données soit valable." & vbCrLf & vbCrLf & _
    "Description de l'erreur :" & vbCrLf & _
    Err.Description, vbOKOnly + vbExclamation, _
    "Erreur dans la fonction ChargerMotsClés du module Menu hiérarchique"
DoCmd.Close acForm, "ProgressBar" 'fermeture du formulaire ProgressBar
Resume Exit_ChargerMotsClés
End Sub
```

```
Private Sub ChargerMotsClés1(ByVal iParent As Long, ByVal iMaxDepth As Long)
On Error GoTo Err_ChargerMotsClés1
Dim SousMenu As CommandBarPopup, OptionMenu As CommandBarControl

If iMaxDepth > 20 Then MsgBox "Vous ne pouvez pas avoir plus de 20 niveaux
dans le menu des mots clés.": Exit Sub
Dim TypeMenu
Dim rs As Recordset
Set rs = Currentdb.OpenRecordset("SELECT * from MotsClés where parent=" &
Str$(iParent) & " ORDER BY libellé;")
If rs.RecordCount > 0 Then

    rs.MoveLast
    rs.MoveFirst
   Do While Not rs.EOF
'****************** Barre de progression  **********************
   Compteur = Compteur + 1
   ProgressMots = 100 * Compteur / TotalMots
   Forms!ProgressBar!Pcent = Compteur & " / " & TotalMots
   DoEvents
   'Forms!ProgressBar!txtProgression = ProgressMots & "%"
   'DoEvents
   Forms!ProgressBar!txtProgression.Width = 43 * ProgressMots
'*************************************************************
    Dim Childs As Long
    Childs = DCount("index", "motsclés", "parent=" & rs!index)
      If Childs = 0 Then
         TypeMenu = msoControlButton
      Else
         TypeMenu = msoControlPopup
      End If

      If rs!Parent <> 0 Then
         Set SousMenu = Application.CommandBars.FindControl(msoControlPopup,
, "Index" & rs!Parent)
         Set OptionMenu = SousMenu.Controls.Add(TypeMenu, , , , True)
         If SousMenu.Controls.Count <= 2 Then OptionMenu.BeginGroup = True
      Else
         Set OptionMenu =
Application.CommandBars(NomMenu).Controls.Add(TypeMenu, , , , True)
      End If
      OptionMenu.Caption = rs!libellé

      If Childs > 0 Then
         OptionMenu.Tag = "Index" & rs!index
         Set OptionMenu = OptionMenu.Controls.Add(msoControlButton, , , ,
True)
         OptionMenu.Caption = "Sélectionner " & rs!libellé
      End If
      OptionMenu.OnAction = "SelectOptionMenu"
      OptionMenu.Tag = rs!libellé
      ChargerMotsClés1 rs!index, iMaxDepth + 1
      rs.MoveNext
   Loop
End If
rs.Close
Set rs = Nothing
Set OptionMenu = Nothing
Set SousMenu = Nothing
Exit_ChargerMotsClés1:
```

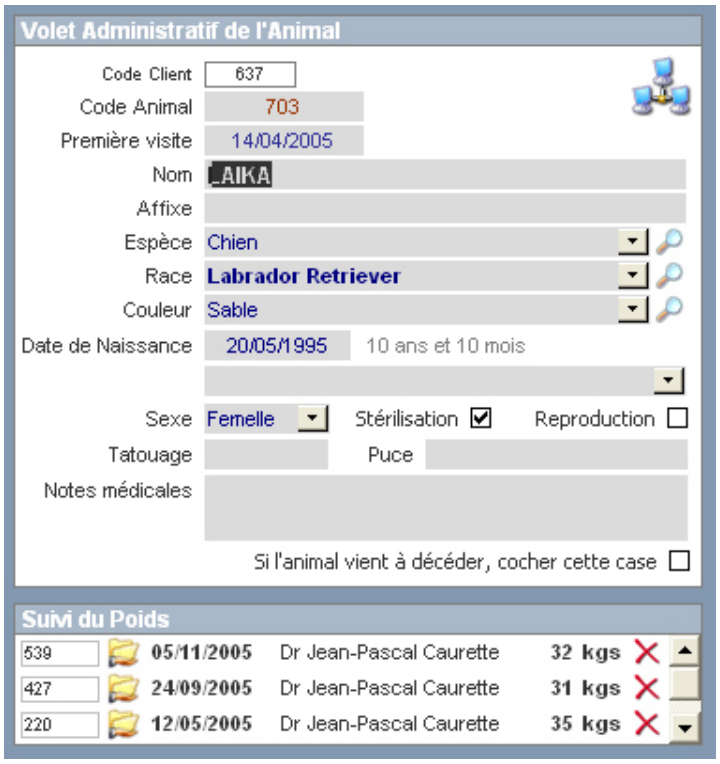
```
Exit Sub
Err_ChargerMotsClés1:
MsgBox "Cher Confrère," & vbCrLf & _
    "Une erreur s'est produite lors du chargement des mots clés." & vbCrLf &
Err.Description, _
    vbOKOnly + vbExclamation, _
    "Erreur dans la fonction ChargerMotsClés1 du module Menu hiérarchique"
Resume Exit_ChargerMotsClés1

End Sub

Function AfficherMenu()
On Error GoTo Err_AfficherMenu

Application.CommandBars(NomMenu).ShowPopup

Exit_AfficherMenu:
Exit Function

Err_AfficherMenu:
MsgBox "Cher Confrère, " & vbCrLf & Err.Description & _
    " dans la fonction AfficherMenu du module Menu Hiérarchique."

Resume Exit_AfficherMenu

End Function

Sub SelectOptionMenu()
  MotCle = Application.CommandBars.ActionControl.Tag
End Sub

Sub GetMotCle()
'Fonction utilisée dans les formulaires pour attribuer un mot clé à un
patient ou une photo.
'Colle le mot clé selectionné dans le contrôle actif du formulaire.
On Error GoTo Err_GetMotCle

    MotCle = Null
    Application.CommandBars(NomMenu).ShowPopup
    If Not IsNull(MotCle) Then
        Screen.ActiveControl = MotCle
        DoCmd.RunCommand acCmdSaveRecord
    End If

Exit_GetMotCle:
Exit Sub
Err_GetMotCle:
MsgBox "Cher Confrère, " & Err.Description & " dans la fonction GetMotCle du
module Menu Hiérarchique."
Resume Exit_GetMotCle

End Sub

Sub AjoutMotCle()
'Fonction utilisée dans le formulaire "Ajouter un mot clé"
'pour remplir la table des mots clés sans erreur de parents.
'Colle le mot clé selectionné dans le champ du mot cle parent ainsi que son
index et son parent.
'Attribue l'index de ce mot clé au champ "Parent" du futur mot clé.
On Error GoTo Err_AjoutMotCle
Dim NewParent As Long
```

```
Dim FormAjoutMotClé As Form
Set FormAjoutMotClé = Forms![Ajouter un mot clé]
Dim rs As Recordset
MotCle = Null
Application.CommandBars(NomMenu).ShowPopup
If Not IsNull(MotCle) Then
    Set rs = Currentdb.OpenRecordset("Select * FROM MotsClés WHERE Libellé =
" & Chr(34) & MotCle & Chr(34))
    If rs.RecordCount > 0 Then
        FormAjoutMotClé![MotCleRef] = rs!libellé
        FormAjoutMotClé![IndexRef] = rs!index
        FormAjoutMotClé![ParentRef] = rs!Parent

        'Si le contrôle "index" est nul, c'est qu'aucun mot-clé enregistré
n'est affiché :
        'modification de parent et de niveau
        If (IsNull(FormAjoutMotClé![index])) Then
            FormAjoutMotClé![Parent] = rs!index

        'Sinon la fonction va modifier le parent du mot-clé affiché,
        'ce qui va le déplacer dans le sous menu du mot-clé selectionné.
        Else
            Dim chMsg As String
            chMsg = "ATTENTION" & vbCrLf & _
            "Vous allez déplacer le mot-clé: [" & FormAjoutMotClé!libellé & _
            "] et son sous-menu dans le sous-menu du mot-clé: [" & rs!libellé
& "]." & vbCrLf & vbCrLf & _
            "Voulez-vous continuer ?"
            If MsgBox(chMsg, vbYesNo + vbExclamation + vbDefaultButton2,
"Déplacement d'un mot-clé") = vbYes Then
                NewParent = rs!index
                'Fonction ParentOK pour vérifier que le mot clé ne va pas
être déplacé dans son sous-menu
                If Not ParentOk(NewParent, FormAjoutMotClé!index) Then
                MsgBox "Il est impossible de déplacer un mot-clé dans son
propre sous-menu.", _
                vbOKOnly + vbCritical, "Déplacement impossible"
                Else
                'Déplacement du mot clé
                FormAjoutMotClé!Parent = NewParent
                End If
            End If
        End If
    End If
    rs.Close
    Set rs = Nothing
End If
Set FormAjoutMotClé = Nothing
Exit_AjoutMotCle:
Exit Sub
Err_AjoutMotCle:
MsgBox "Cher Confrère, " & Err.Description & " dans la fonction AjoutMotCle
du module Menu Hiérarchique."
Resume Exit_AjoutMotCle

End Sub

Private Function ParentOk(ByVal ChercherIndex As Long, ByVal EviterIndex As
Long) As Boolean
    If ChercherIndex = EviterIndex Then ParentOk = False: Exit Function '
propre parent
```

```
    Dim ParentChercherIndex As Variant
    ParentChercherIndex = DLookup("Parent", "MotsClés", "index=" &
Str$(ChercherIndex))
    If IsNull(ParentChercherIndex) Then ParentOk = True: Exit Function '
Racine trouvée sans problèmes
    ParentOk = ParentOk(ParentChercherIndex, EviterIndex) ' remonte au parent
End Function
```

AJOUT, MODIFICATION ET SUIVI DU POIDS D'UN ANIMAL

Disponible depuis la version 1.3 au 20 mars 2006



Suivi du Poids de l'Animal

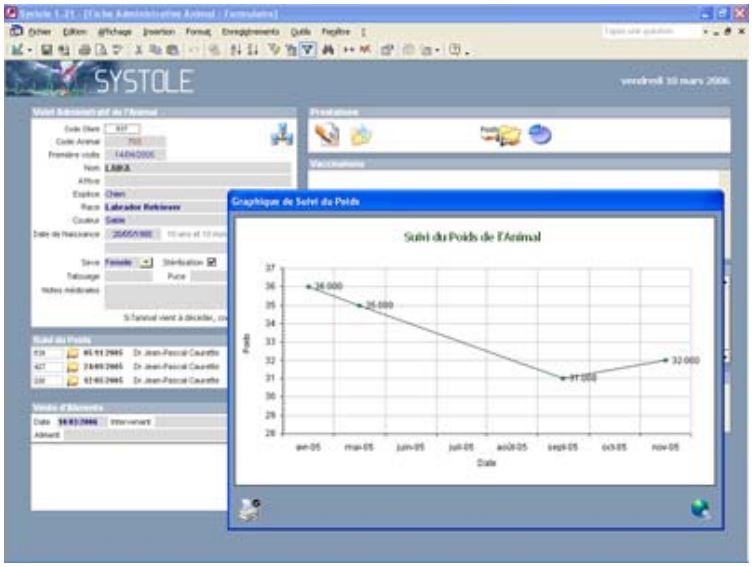
Le suivi du poids est important dans l'activité quotidienne, aussi bien pour évaluer la croissance que l'amaigrissement lors d'un régime ou d'une pathologie chronique. Systole intègre un module de suivi de poids avec représentation graphique de l'évolution pondérale.

Deux modes de suivi sont disponibles :

- un historique des pesées est présent sous le Volet Administratif de l'Animal, dans la Fiche Administrative Animal. Il indique dans l'encadré le numéro système de la pesée, la date de la pesée, l'intervenant ayant réalisé la pesée, et le poids. L'icône «Dossier » à gauche de la date permet d'accéder à la fiche de pesée, de visualiser l'éventuel commentaire ou de modifier la peser. L'icône « Supprimer » à l'extrême droite permet de supprimer la pesée.
- Un graphique reprenant l'évolution du poids au fil du temps.

Accéder au Graphique de Suivi Pondéral

Pour accéder au Graphique de Suivi du Poids, il suffit de cliquer sur l'Icône  dans la barre « Prestations ». La fenêtre du graphique s'affiche alors.



Le Graphique du Suivi Pondéral donne l'évolution du poids en fonction du temps. Lors de la visualisation de cet écran, il n'est plus possible d'intervenir sur la Fiche de l'Animal.

Pour sortir du Graphique, il suffit de cliquer sur l'Icône  en bas à droite de la fenêtre du Graphique. Le Graphique se ferme, et on revient à la Fiche Administrative de l'Animal.

Il est possible **d'imprimer le Graphique**, afin de le donner au propriétaire. Pour cela, il suffit de cliquer sur l'Icône  en bas à gauche de la fenêtre.

Attention : le graphique s'imprime immédiatement, sans aperçu, il est impossible d'annuler cette demande.

Créer une Nouvelle Pesée

Dans Systole, les pesées sont enregistrées et sont reliées à l'Animal, et non aux Prestations. Cela permet un suivi du poids séparé : à l'occasion de la délivrance d'un vermifuge ou d'un anti-parasitaire, les ASV peuvent faire peser et enregistrer la valeur. De même, il est possible de proposer au propriétaire de passer de temps à autre à la Clinique pour réaliser un suivi pondéral lors d'un schéma d'amaigrissement. Ainsi, le module Pesée de Systole permet de réaliser un suivi étroit en terme de poids.

Pour enregistrer une Nouvelle Pesée, il suffit de cliquer sur l'icône 

dans l'onglet « Prestations » de la Fiche Administrative de l'Animal. La fenêtre suivante apparaît :

La date du jour s'affiche automatique. Il est possible de la modifier.
Le champ « Commentaires » permet d'enregistrer des remarques sur la pesée ou l'évolution du poids de l'animal.
Le champ « Poids », à fond jaune, vous permet de saisir le poids de l'animal. **Attention, si vous rentrer des grammes, il faut impérativement utiliser la virgule comme séparateur.**
Le champ « Intervenant » vous permet de saisir la personne qui réalise la pesée : on clique sur le champ, un menu contextuel apparaît : il suffit de cliquer sur la personne concernée.

Une fois la fiche remplie, on clique sur l'Icône  pour enregistrer la saisie, revenir à la fiche de l'Animal et mettre à jour l'Historique des pesées.

Pour Annuler la saisie en cours et revenir à la fiche de l'Animal, on clique sur l'Icône  .

Visualiser ou Modifier une Pesée

Pour Visualiser ou Modifier une Pesée, il suffit d'afficher l'écran de visualisation en cliquant sur l'Icône  dans la fenêtre d'Historique des poids.

La fenêtre de visualisation de la pesée s'affiche. On peut modifier le champ désiré. Une fois le travail terminé, on clique sur l'Icône  pour enregistrer et revenir à la fiche de l'Animal.

Supprimer une Pesée

Dans la fenêtre d'Historique des Poids, on clique sur l'Icône  en regard de la Pesée que l'on désire supprimer. Systole demande une confirmation avant de supprimer l'enregistrement.