



Interrogation floue de Bases de données: extension de iSQLf

BARTEL Samuel
2005-2006

Tuteur: Olivier PIVERT
Laboratoire Lannionnais d'Informatique
pivert@enssat.fr

Abstract

Since fuzzy logic has proven to be a very useful tool for representing human knowledge by means of mathematical expressions and because database are widely used in computer science, research has been conducted to extend database functionalities with a fuzzy querying system. The architecture studied by the Badins team consists in using a regular database management system (DBMS) and developping an additionnal layer playing the role of an interface between the DBMS and a fuzzy SQL language.

A project aiming to implement this interface and based on theoritical work done by the Badins team on the derivation of fuzzy queries into boolean queries used by all the main DBMS currently available has started six years ago and a prototype has been built. Since the year 2000 web architecture works allowing to process queries without necessarily being logged on a network at ENSSAT.

The next step is to improve human computer interface, especially for error management and for the interface which is not enough ergonomic and to add new functionalities for nested queries.

Keywords

Fuzzy queries, query language, database, CGI, Web server, lexical analyser.

Table des matières

Glossaire	p5
Introduction	p8
1. Description du sujet	p9
1.1 Cadre du projet	p9
1.2 Contexte	p9
1.2.1. Présentation	p9
1.2.2. Modélisation d'un terme flou	p10
1.2.3. Requêtes floues	p11
1.2.4. Présentation du logiciel iSQLf	p12
2. Objectif	p16
2.1 Etat des lieux et Objectif initial	p16
2.2 Problèmes rencontrés	p16
2.2.1. Méthode de travail prévue initialement	p16
2.2.2. Interface pour utilisateurs experts	p17
2.2.3. Interface pour utilisateurs non experts	p17
2.2.4. Bugs persistants	p19
2.2.5. Problèmes de compilation	p20
2.3 Refonte des objectifs	p21
3. Refonte de l'interface Web	p22
3.1 Interface pour utilisateurs experts	p22
3.2 Interface pour utilisateurs non experts	p22
4. Correction des bugs persitants	p30
4.1 Problèmes liés à la gestion du seuil	p30
4.2 Chargement des requêtes sauvegardés	p30
4.3 Opérateur NOT pour des requêtes classiques	p31
4.4 Interdire le flou sur des attributs de type chaine de caractères	p31
4.5 Problème lié aux attributs présents dans la clause WHERE	p31
4.6 Opérateur “=” associé à des termes non flous	p32
5. Perspectives	p33
5.1 Amélioration de l'interface	p33
5.2 Amélioration des performances	p34
6. Planning récapitulatif du travail effectué	p36
Conclusion	p37
Références	p38
Annexes	p39
A)requete.c	p39

B)chReq1.c	p40
C)chReq2.c	p41
D)chReq3.c	p45
E)chReq4.c	p49
F)chReq5.c	p56
H)perfreq.c	p59
I)derivation.c	p67
K)moteur.c	p96
L)scn1.l	p102

Glossaire

CGI

Common Gateway interface. Interface du serveur HTTP composée par un ensemble de programmes qui permettent à l'utilisateur de disposer de documents web réalisés de façon dynamique sur requête du navigateur. Les scripts CGI s'exécutent sous forme de processus indépendants, lancés par le serveur HTTP

HTML

Hyper Text Markup Language. Langage de description des pages web.

HTTP

Hyper Text Transport Protocol. Protocole permettant le transport des pages web.

iSQLf

Le logiciel développé pour l'interrogation flexible de bases de données par l'équipe Badins

Javascript

Langage de programmation s'intégrant à du code HTML afin de rendre dynamique la page Web générée (On parle alors de DHTML).

Lex et Yacc

Langages de description d'analyseurs lexicaux et syntaxiques. Utilisés dans le logiciel iSQLf pour définir la sur-couche SQLf du langage SQL.

Pro*C

Langage de programmation formant une sur-couche du langage C. Il permet l'interrogation d'un SGBD via des requêtes SQL intégrées à du code développé en C.

Serveur HTTP

Serveur web. Logiciel fournissant un accès à des pages web, via le protocole HTTP. Le serveur web utilisé ici par iSQLf est un serveur Apache.

SGBD

Système de gestion de bases de données constitué d'un ensemble de programmes, il assure la gestion et l'accès à une base de données. Il permet l'ajout de données, la modification de données, ainsi que la recherche des données dans une ou plusieurs bases de données. Le SGBD utilisé par iSQLf est Oracle.

SQL

Vient du sigle acronyme Sequel (Structured English QUery Language). Langage permettant la définition de relations et de contraintes, et la manipulation des données d'un SGBD relationnel.

SQLf

Surcouche du langage SQL mise au point par l'équipe du projet Badins. Elle intègre la définition d'un environnement flou.

Sqlplus

logiciel utilisé pour interroger une base de données avec le langage SQL via un terminal sous Unix.

Tristan

Nom du serveur local à l'ENSSAT utilisé pour le développement, l'hébergement et l'exécution d'iSQLf. Le serveur Apache notamment est installé sur un compte utilisateur sur Tristan et est accessible par défaut sur le port 1212 à l'adresse : <http://tristan.enssat.fr:1212/sqlf>. Ce port est, pour des raisons de sécurité, inaccessible en dehors du réseau local. Il est cependant possible de demander à l'administrateur du réseau de l'ENSSAT de permettre, de façon ponctuelle, l'ouverture de ce port afin de permettre des démonstrations du logiciel à l'extérieur de l'ENSSAT.

Unix

Système d'exploitation très répandu dans le domaine des stations de travail et des serveurs. Tristan est sous Unix.

Oracle

Système de gestion de bases de données relationnelles couramment utilisé dans les applications sur différentes plate-formes.

XUL

XUL, est un langage basé sur XML pour décrire une interface graphique, utilisé dans Firefox et les autres logiciels Mozilla.

Jflex

Flex(Java Fast Lex) est un générateur d'analyseur lexical, il est prévu pour fonctionner avec CUP ou tout seul.

CUP

Cup est un générateur d'analyseur LALR pour Java (équivalent à yacc).

ANTLR

ANTLR (ANother Tool for Language Recognition) est un utilitaire qui permet de construire un analyseur syntaxique à partir d'une grammaire spécifique. ANTLR peut gérer du code en Java ou en C++.

Introduction

Ce projet a pour cadre l'interrogation flexible de bases de données. Un logiciel a été développé depuis quelques années dans le cadre des activités de l'équipe Badins, groupe de recherche de l'IRISA. On s'intéresse à l'application de la théorie des ensembles flous dans le domaine des bases de données afin d'offrir à l'utilisateur des outils lui permettant de formuler des requêtes comportant des termes imprécis. Le logiciel développé, iSQLf, se présente comme un système d'interrogation imprécise, et dont l'objectif est de fournir un ensemble de réponses discriminées. Un degré de satisfaction est ainsi donné à l'utilisateur pour chaque nuplet retenu relativement à la requête.

L'évolution de requêtes floues est un problème complexe. L'approche retenue consiste à utiliser un système de gestion de base de données (SGBD) classique et à développer une couche supplémentaire qui joue le rôle d'interface. Une première version d'un tel prototype a été réalisée en 1997-1998. Depuis 2000, l'architecture a été revue et un serveur Web a été mis en oeuvre afin de rendre disponible l'application à partir de tout système connecté à Internet. Il s'agit maintenant d'améliorer le serveur existant en rendant son accès le plus intuitif possible ainsi que de corriger les bugs présents qui empêchent la poursuite du développement.

Dans ce document, vous seront d'abord présentés le projet iSQLf et son contexte. Ensuite nous nous intéresserons à l'état des lieux du projet à la rentrée 2005. Puis nous aborderons les différents travaux réalisés durant cette année. Enfin vous seront présentés les perspectives d'évolution du projet iSQLf pour l'année prochaine.

1. Description du sujet

1.1 Cadre du projet

Le logiciel iSQLF est réalisé dans le cadre du projet Badins du laboratoire Lannionnais d'informatique (L.L.I.), groupe de recherche de l'IRISA. Afin de mieux situer le contexte d'études de ce projet, voici quelques informations extraites du site internet de l'ENSSAT concernant le L.L.I.

La création de l'ENSSAT a conduit à la création d'un pôle universitaire en informatique commun à l'I.U.T. et à l'ENSSAT. Les activités sont regroupées autour de deux thèmes fondateurs : l'imprécision dans les systèmes d'information et le dialogue homme-machine qui prolongera les recherches précédentes autour de la parole.

Le projet Badins (Bases de Données et Interrogation Souple) se situe à la croisée de deux domaines de recherches : les bases de données et la représentation de connaissances imparfaitement connues. Globalement, les axes de recherche développés dans ce projet visent à relaxer l'hypothèse de connaissance parfaite qui sous-tend l'utilisation des SGBD actuels. Le projet aborde donc un certain nombre d'aspects des SGBD en les considérant dans le cadre de l'imprécis.

1.2 Contexte

1.2.1 Présentation

Le logiciel iSQLf doit permettre de conjuguer les bases de données et la représentation de notions imprécises. Plus précisément ce logiciel sera capable au final de réaliser les mêmes opérations que celles possibles pour un SGBD classique, tout en introduisant en plus la possibilité d'interroger ceux-ci à partir de questions imprécises (ou floues)

Quel est l'avantage d'ajouter de l'imprécision? Cela évite une trop haute sensibilité au choix des paramètres, mais permet aussi de filtrer les résultats selon le degré de correspondance à une requête posée au système ou d'introduire un critère de préférence. Ceci n'est pas possible avec un système classique basé sur la logique booléenne qui permet uniquement de répondre oui ou non à une question.

Exemple : prenons une requête afin d'observer les différences entre un système classique utilisant la logique booléenne et une requête utilisant du flou. La question posée en Français est:

“Quels sont les restaurants pas chers et proches de Lannion?”

1. Les deux termes “pas chers” et “proches” sont incompréhensibles par un SGBD classique. Si l'on veut répondre à cette requête en utilisant la logique booléenne, l'utilisateur devra donc interroger la BDD en transformant ces termes en conditions booléennes. L'utilisateur peut alors restreindre ses critères de recherche mais prend alors le risque de n'obtenir que peu voir aucune réponse ou alors il les élargit et prend le risque d'avoir un nombre important de réponses sans aucune préférence entre celles-ci.

<i>Terme</i>	<i>“pas chers”</i>	<i>“proches de Lannions”</i>	<i>Nombre de réponses</i>
Restriction	<15 €	<5 Km	Faible à nul
Élargissement	<30 €	< 25 Km	Elevé

Les requêtes en SQL correspondantes seront de la forme suivante:

```
select nom from Restaurants where prix < 15 and distance < 5;
```

Il est donc difficile de trouver une requête adaptée au résultat souhaité, et de plus, dans les deux cas, toutes les réponses sont placées sur un même pied d'importance : par exemple, un restaurant à 15 € et situé à 6 km ne sera pas mis en avant par rapport à un restaurant à 30 € et situé à 20 km

2. Si l'on utilise maintenant des termes flous, l'utilisateur peut poser sa question comme il la formule en français. L'autre avantage est que pour chaque réponse obtenue, un coefficient de correspondance au souhait de l'utilisateur sera présent ce qui permet de discriminer les réponses. La requête floue aura la forme suivante:

```
select nom from Restaurants
where prix = “pas chers” and distance = “proche de Lannion”
```

Puisque ces requêtes sont incompréhensibles de manière directe par le SGBD, l'utilisateur doit au préalable définir ce qu'il entend par “pas cher” et “proche de Lannion”. Ce sont des notions subjectives dont la traduction dépend de la personne mais également du contexte dans lequel on se place. En effet, ce qui est cher pour une personne ne l'est pas forcément pour une autre par exemple. De même ce qui est cher pour un restaurant ne l'est pas pour une voiture. Chaque utilisateur possède ainsi son propre contexte flou où chaque terme défini est associé à un attribut et à une base précis

Pour résumer, nous pouvons dire que l'utilisation de termes flous est plus souple et plus proche du raisonnement humain: on ne cherche plus à savoir si des éléments satisfont ou pas une condition, mais à quel degré ils la satisfont. Nous allons maintenant voir comment modéliser un terme flou.

1.2.2 Modélisation d'un terme flou

Un terme flou correspond à un adjectif de la langue naturelle. Pour modéliser un terme flou, une fonction trapézoïdale est utilisée (cf figure1). Il reste pour l'utilisateur du terme flou à définir les paramètres caractérisant cette fonction.

Une fonction trapézoïdale étant définie par le quadruplet ((A,0);(B,1);(C,1);(D,0)), les valeurs de A,B,C,D sont à renseigner.

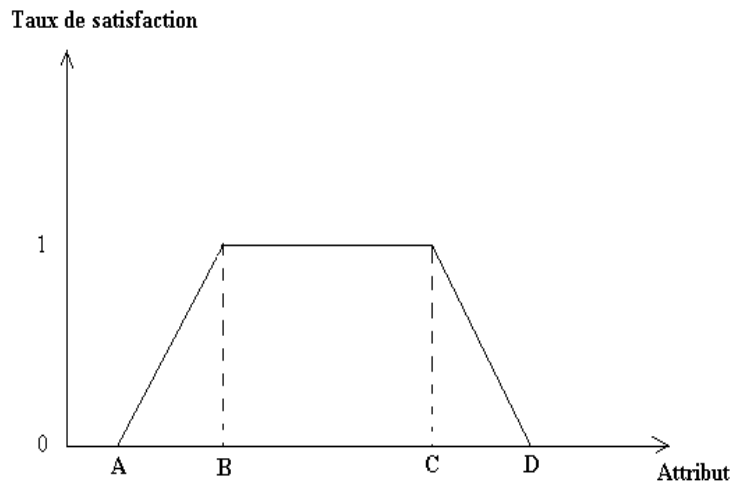


FIG. 1 Fonction trapézoïdale de définition des termes flous

Exemple: Prenons l'exemple du terme flou “jeune” appliqué à un employé. Une définition de ce terme peut être qu'un employé est jeune si son age est inférieur à 24 ans, puis de plus en plus âgé entre 24 et 30 ans et âgé au delà. Ce qui nous donne pour valeur : $A=0$, $B=0$, $C=24$ et $D=30$ car 24 et 30 sont les points pour lesquels la fonction change d'équation. La fonction trapézoïdale résultante est la suivante :

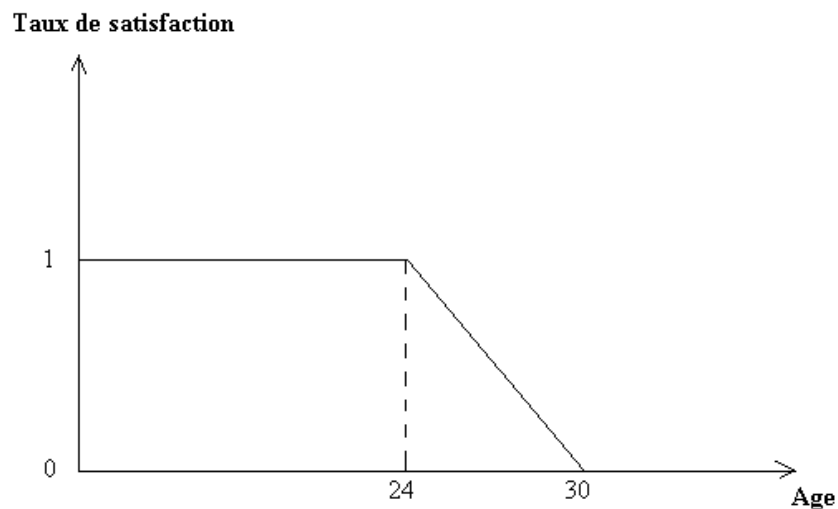


FIG. 2 Fonction de définition du terme flou jeune

1.2.3 Requêtes SQLf

Le langage SQL a été défini durant les années 1970 afin de permettre l'interrogation de bases de données relationnelles de manière naturelle. Ce langage est rapidement devenu un standard et a été étendu de différentes manières depuis. Il est composé d'un bloc de base de la forme :

```
SELECT <liste d'attributs>
FROM <liste de tables>
```

WHERE <conditions>

Afin d'utiliser les travaux de recherche sur l'interrogation floue menés par le L.L.I., les chercheurs du projet Badins ont mis au point le langage SQLf qui étend le bloc d'une requête SQL de la manière suivante :

```
SELECT [n|λ] <liste d'attributs>
FROM <liste de tables>
WHERE <conditions pouvant être floues>
```

La valeur **n** est le calibre, c'est à dire le nombre maximal de réponses désirées. La valeur **λ** est un seuil qualitatif permettant de ne sélectionner que les enregistrements de la base répondant avec une valeur suffisamment grande à la condition. Cette condition peut maintenant contenir des éléments permettant une gradualité dans les réponses.

Nous avons décrit dans un premier temps les termes flous. S'il est vrai que ce sont les principaux éléments abordés pour comprendre le fonctionnement de telles requêtes, le langage SQLf permet l'utilisation d'autres composants pour l'élaboration d'une interrogation flexible. Ils sont décrits dans la liste suivante.

Modificateurs flous

Ils augmentent ou atténuent l'effet d'un terme flou en modifiant la fonction trapézoïdale. Ils sont de type exponentiel et sont définis par la valeur de leur exposant. Par exemple, le modificateur "très" pourra être décrit par une puissance égale à 2.

Comparateurs flous

Ils sont définis par une fonction trapézoïdale et une fonction définissant le lien entre les opérandes. Par exemple, le comparateur "largement supérieur" (>>) effectue une soustraction entre les deux opérandes et le résultat de cette soustraction permet de trouver le degré de satisfaction de la comparaison.

Connecteurs flous

Ils permettent la combinaison de plusieurs éléments. Il existe les classiques opérateurs d'union et d'intersection, ainsi que des connecteurs plus propres au monde des requêtes floues, comme "la plupart des conditions {c1,...,cn} ".

1.2.4Présentation du logiciel iSQLf

Pour mettre en pratique l'interrogation floue sur des bases de données conventionnelles, il a été mis au point un logiciel nommé iSQLf. Développé depuis 1997, il permet à un utilisateur d'interroger une base de données en utilisant des requêtes floues. Le logiciel agit comme une interface entre un utilisateur particulier et le système de bases de données interrogé, qui ne comprend pas directement la logique floue.

Pour l'utilisateur, iSQLf se présente sous la forme d'un serveur web. Après avoir validé son entrée sur la page d'accueil par un mot de passe, l'utilisateur peut interroger une ou plusieurs bases de données sur lesquelles il possède les droits d'utilisation.

S'il connaît la structure de la base interrogée, il peut poser directement des requêtes en langage SQL classique ou bien en SQL flou. Sinon il peut utiliser l'interface d'interrogation pour les non-experts (*non-expert query*) afin d'être aidé dans l'élaboration de sa requête. iSQLf interroge ensuite la base concernée et retourne le résultat de la requête (figure 3).

NOM	Satisf
SALLE	1.00
DUBREUIL	1.00
DOUSSAU	1.00
BILLI	1.00
BOB	1.00
BOBO	0.67
BON	0.50
DELAMOTTE	0.33
CHARLES	0.00
ABDOUL	0.00
DUPONT	0.00
RENOIR	0.00
TIM	0.00

[Sauvegarder cette requête dans mon panier de requêtes](#)

For any further question or comment: [contact us](#)

Terminé

FIG. 3 Exemple de résultat d'une requête sous iSQLf

Pour comprendre comment fonctionne ce logiciel lors d'une requête, on peut séparer ses fonctionnalités en deux groupes :

- l'interface Web, sur un poste classique, qui permet à l'utilisateur de lancer des requêtes et de visualiser les résultats,
- le moteur SQLf, dans le serveur, qui traduit la requête en une requête SQL classique, prépare et lance un exécutable qui va interroger la base de données concernée.

L'architecture du logiciel peut être vue comme la rencontre de plusieurs technologies client/serveur :

- le navigateur Web utilisé est un client se connectant via Internet au logiciel iSQLf, qui assure le rôle de serveur HTTP,
- le logiciel iSQLf est lui-même un client se connectant via le réseau local (ou sur la même machine) au serveur de bases de données.

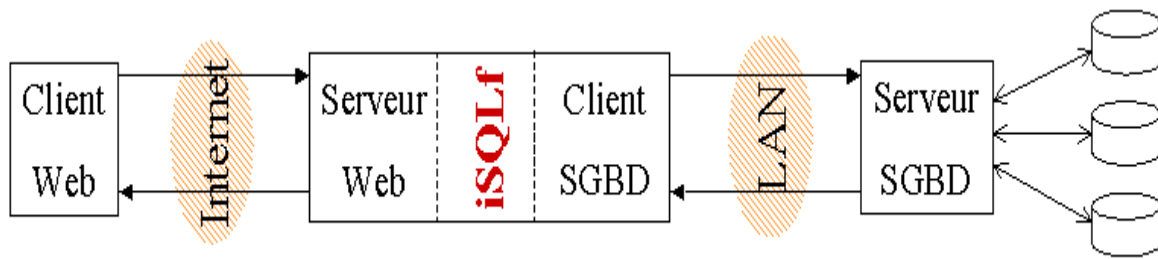


FIG. 4 Architecture globale du système

En explorant plus en détail l'architecture, on voit que celle est coconstituée plusieurs modules:

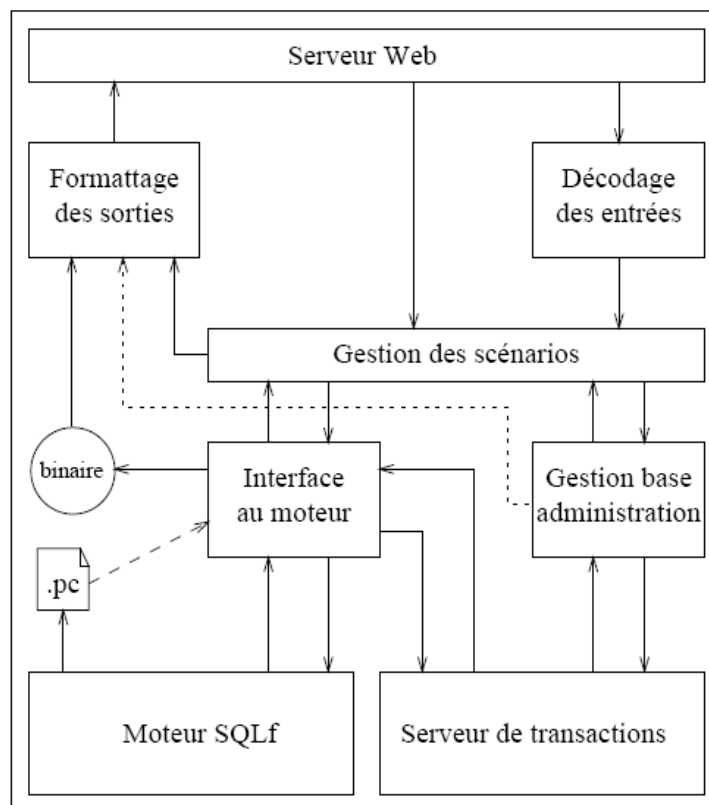


FIG. 5 Architecture détaillée du système

- le serveur web : il permet à l'utilisateur d'interagir avec le logiciel.
- le décodage des entrées : il permet l'interprétation des données saisies par l'utilisateur.
- la gestion des scénarios : elle permet au logiciel de définir dans quelle situation on se situe (connexion en mode administrateur, phase de login, utilisation en mode expert, utilisation en mode non expert, etc).
- la gestion base administration : elle permet à l'administrateur de définir des nouveaux utilisateurs et leur donner des droits d'accès aux différentes bases du système.
- le serveur de transaction : il réalise les échanges entre le logiciel et la BDD.
- le moteur SQLf : c'est le module principale du logiciel iSQLf.

Il réalise l'interprétation et la traduction des requêtes SQLf en requête booléenne à l'aide du traducteur syntaxique Lex et Yacc. Il génère ensuite un fichier Pro*C qui est envoyé au serveur de transactions par l'intermédiaire de l'interface au moteur.

Lorsqu'il reçoit le résultat d'une requête, il réalise les opérations de calcul des degrés de satisfaction des différents n-uplets du résultat.

- L'interface au moteur: elle permet la communication entre l'interface web et le moteur SQLf

- le formatage des sorties: ce module récupère les résultats des requêtes et produit un document HTML permettant leur affichage sur un navigateur.

Au cours de ce projet les parties sur lesquelles mon travail s'est porté sont principalement le serveur web et le moteur SQLf

2. Objectif

2.1 Etat des lieux et Objectif initial

le cadre de ce projet s'inscrit dans la reprise d'un logiciel ayant été réalisé par plusieurs étudiants depuis 1997.

Les deux premières versions permettaient l'entrée de requêtes floues par ligne de commande. C'est à cette époque qu'a été conçue la première version du moteur iSQLf. Depuis mars 2000, un utilisateur a la possibilité d'interroger des bases de données par l'intermédiaire d'un navigateur Internet. En 2001 et 2002, les principaux ajouts ont été la fonction d'interrogation floue par un non expert via l'interface web, ainsi qu'un travail de documentation et de fiabilisation. En 2004 une analyse détaillée du fonctionnement du logiciel a été réalisée afin de s'assurer du bon fonctionnement de l'existant et une documentation visant à aider la prise en main du logiciel a été rédigée.

En Septembre 2005, iSQLf est un logiciel en phase de finalisation pour ce qui est des requêtes simples. Au delà des fonctions essentielles, il est important de pouvoir l'améliorer en proposant aux utilisateurs des services qui sont présents dans les SGBD classiques. Par exemple, il n'est possible de formuler que des requêtes simples (en un bloc) avec le logiciel . Il serait intéressant de rendre possible l'utilisation de requêtes imbriquées.

Exemple:

```
select employé
  from emp
 where salaire=élevé and noDep in (select noDep
                                   from Dep
                                   where budget=faible);
```

cette requête permet de sélectionner les employés jeunes travaillant dans un département dont le budget est faible.

Cette forme de requêtes est très souvent utilisée dans les SGBD, notamment pour réaliser les opérations de divisions.

2.2 Problèmes rencontrés

2.2.1 Méthode de travail prévue initialement

Au début du projet l'organisation du travail était prévue en 3 étapes. Dans un premier temps, la prise en main du logiciel et la bonne compréhension de son architecture. Puis une phase de

réalisation du module permettant le traitement des requêtes imbriquées à partir de travaux théoriques réalisés par l'équipe de recherche du projet Badins. Et enfin une phase de test afin de s'assurer du bon fonctionnement du module réalisé.

Cependant lors de la première phase de mon travail, de nombreux problèmes liés aux interfaces pour utilisateurs experts et pour utilisateurs non experts ont été découverts. Par ailleurs de nombreux bugs persistaient dans les différentes couches du logiciel rendant difficile la poursuite du développement.

2.2.2 Interface pour utilisateurs experts

L'interface pour utilisateurs experts s'est avéré être inadaptée aux requêtes imbriquées.

FIG. 6 Ancienne interface pour experts

En effet le champ de saisie where n'est pas assez grand pour que l'utilisateur puisse saisir une condition comme "where age = jeune and noDep in select noDep from Dep where budget = faible". Dans l'optique de l'implémentation des requêtes imbriquées au logiciel iSQLf, il est nécessaire de modifier cette page.

2.2.3 Interface pour utilisateurs non experts

Pour l'interface pour utilisateurs non experts, de gros problèmes d'ergonomie ont été détectés lors de la phase de prise en main du logiciel.

Dans un premier temps l'utilisateur sélectionne la base sur laquelle il veut travailler.

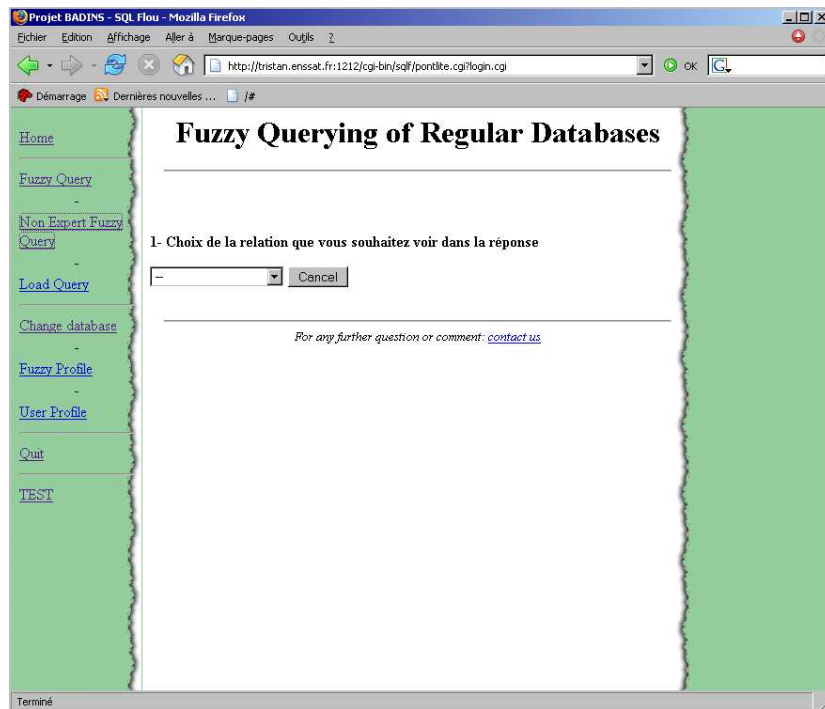


FIG. 7 Selection de la table

Puis il selectionne les attributs qu'il souhaite voir afficher dans la réponse ainsi que le calibre et le seuil.

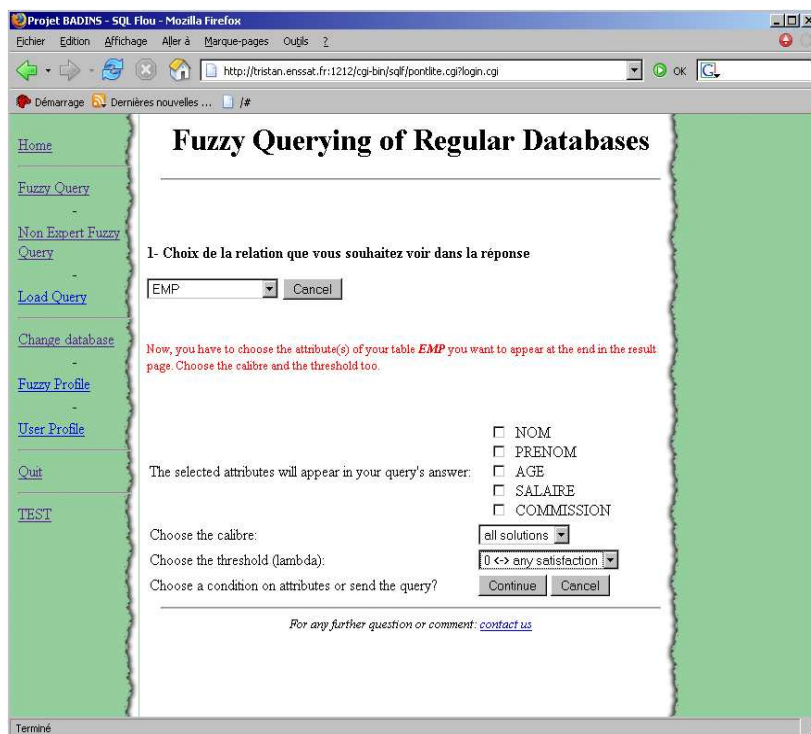


FIG. 8 Selection des attributs

FIG. 9 Ecriture de la clause where

Il rédige enfin la condition where de sa requête.

C'est cette dernière étape qui présente des défauts importants. En effet le fait de cliquer sur un attribut l'ajoute directement au champ de saisie where. De même si l'utilisateur choisit un terme flou, celui ci sera ajouté au champ where et c'est à l'utilisateur d'ajouter le comparateur souhaité. Cette interface n'est pas intuitive et est inadaptée à un utilisateur non expert n'ayant a priori que des connaissances limitées sur la syntaxe SQL.

Il est effet possible de valider la requête sans avoir saisi un comparateur. On obtiendrais alors une requête du type "where age jeune" qui est syntaxiquement fausse. Par ailleurs aucun contrôle n'est effectué pour s'assurer que l'utilisateur utilise un terme flou avec l'attribut pour lequel il a été défini.Ce qui permet l'écriture de requête du type "where age=correcte" ce qui est sémantiquement faux.

Il est donc nécessaire de réaliser une refonte de cette page afin de la rendre la plus intuitive possible et réduire au maximum les erreurs de syntaxes ou les erreurs sémantiques

2.2.4 Bugs persistants

Au cours de la phase de découverte du logiciel de nombreux bugs et dysfonctionnements ont été détectés:

- il était impossible de recharger une requête sauvegardée par l'utilisateur.
- le seuil n'était pas pris en compte pour l'affichage des réponses. En effet si l'utilisateur choisissait un seuil de 0,50, le logiciel affichait tout de même les résultats dont le degré de satisfaction était inférieur au seuil.
- lorsque l'utilisateur choisissait un seuil nul, il n'obtenait que les réponses dont les valeurs étaient comprises dans le trapézoïde

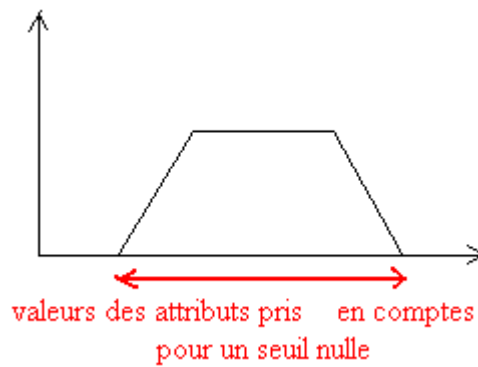


FIG. 10 Valeurs prise en compte pour un seuil de nul

Après discussion avec M. Pivert, il a été décidé que lorsque l'utilisateur choisissait un seuil nul, c'est qu'il désirait obtenir toutes les réponses quelque soit leur degré de satisfaction. Les résultats des requêtes étaient donc faux car ils ne renvoyaient pas les réponses dont la valeur de l'attribut était extérieure aux trapézoïdes.

- l'opérateur not ne fonctionnait qu'utilisé dans une requête floue
- l'opérateur "=" ne fonctionnait qu'associé à un terme flou
- il était obligatoire d'inclure dans la clause select les attributs intervenant dans la clause where

Ces bugs empêchant le fonctionnement correct des modules réalisés les années précédentes il a été décidé de les corriger avant de poursuivre le développement.

2.2.5 Problèmes de compilation

Un gros problème est apparu et m'a empêché de suivre le planning prévu. Le temps passé à trouver l'origine du problème peut être estimé à une dizaine de jours ouvrés de travail et provient entre autre du fait qu'il est extrêmement difficile de corriger des erreurs dans iSQLf de part son architecture multicouche comme dit précédemment, mais aussi de son développement réalisé par de nombreux étudiants avec autant de façons de travailler.

Ce problème n'est pas apparu tout de suite car lors des premiers jours de projet, consacrés à l'exploration et la prise en main du logiciel, il ne fût pas nécessaire d'effectuer des modifications du programme et de le recompiler pour découvrir le fonctionnement de l'application. Cependant lorsque j'ai essayé de corriger les premiers bugs, les modifications effectuées n'étaient pas prises en compte par le logiciel.

Pour cerner le problème, il fût nécessaire de connaître parfaitement l'arborescence et l'architecture du projet. Le problème ne provenant pas du makefile présent à la racine du projet, il m'a fallu étudier en détail les makefiles des différents modules. Le problème provenait d'une option du makefile du module des CGI dont une option avait été supprimée. Les fichiers étaient bien compilés mais n'étaient pas copiés sur le serveur, ce qui rendait toute modification du code sans aucune incidence sur l'application.

Par ailleurs, il s'est avéré indispensable de recompiler tout le projet afin que les changements du code soit pris en compte par le serveur. Une recompilation du module modifié ne suffisait pas.

Cela m'a fait perdre beaucoup de temps au cours du projet car à chaque modification, même minime, du code j'ai dû attendre les 2 minutes de compilation pour pouvoir effectuer des tests.

2.3 Refonte des objectifs

Ces différents problèmes nous ont obligés à revoir nos objectifs à la baisse fin décembre. L'objectif de cette année est finalement de:

- modifier l'interface pour utilisateurs experts afin de la préparer à l'implémentation d'un module de gestion des requête imbriquées
- réaliser une refonte de l'interface pour utilisateurs non experts afin de la rendre plus intuitive et limiter au maximum les risques d'erreurs de syntaxe ou d'erreurs de sémantique
- corriger les bugs persistants de l'application qui rendent difficile la poursuite du développement.

3. Refonte de l'interface Web

3.1 Interface pour utilisateurs experts

L'objectif de la modification de l'interface pour utilisateurs experts avait pour but de la préparer à l'implémentation d'un module de gestion des requêtes imbriquées. Le problème de l'interface précédente était que le champ de saisie *where* n'était pas assez grand pour que l'on puisse saisir une condition comportant une requête imbriquée.

La nouvelle interface pour utilisateurs expert est générée par le fichier `requete.cgi`. Les champs de saisie *select*, *from* et *where* ont été remplacés par un champ de saisie unique requête. L'utilisateur rédige donc sa requête comme il le ferait en ligne de commande sur un SGBD comme Oracle ou mySql. Une fois la requête saisie elle est envoyée au fichier `perfreq.cgi` qui l'interprète avant de la transmettre au moteur. L'avantage d'avoir un champ de saisie unique est de permettre une plus grande liberté à l'utilisateur pour la rédaction de sa requête. Il n'est en effet plus limité par la taille des différents champs pour choisir le nombre d'attributs à afficher, le nombre de tables sur lesquelles sa requête va porter ou alors pour ses conditions.

Cette nouvelle interface prépare donc correctement le logiciel à l'implémentation des requêtes imbriquées.



FIG. 11 Nouvelle interface pour experts

3.2 Interface pour utilisateurs non experts

L'objectif de la refonte de l'interface pour utilisateurs non experts était de la rendre la plus intuitive et de minimiser les risques d'erreurs de syntaxe ou de sémantique. Par ailleurs, après

discussion avec M. Pivert il a été décidé de ne pas préparer cette interface pour les requêtes imbriquées. En effet un utilisateur non expert est supposé ne pas connaître suffisamment la syntaxe SQL pour vouloir rédiger ce type de requêtes.

Cette nouvelle interface est générée par les fichiers chReq1.cgi, chReq2.cgi, chReq3.cgi, chReq4.cgi et chReq5.cgi.

L'utilisateur commence par choisir la table sur laquelle il veut travailler. La page de sélection de la page est générée par le fichier chReq1.cgi.



FIG. 12 Nouvelle interface pour non experts

Une fois la table choisie, son nom est transmis au fichier chReq2.cgi. Cette page recherche dans la base l'ensemble des attributs définis pour cette table et les affiche. L'utilisateur peut ensuite choisir les attributs qu'il veut afficher dans la réponse de sa requête.

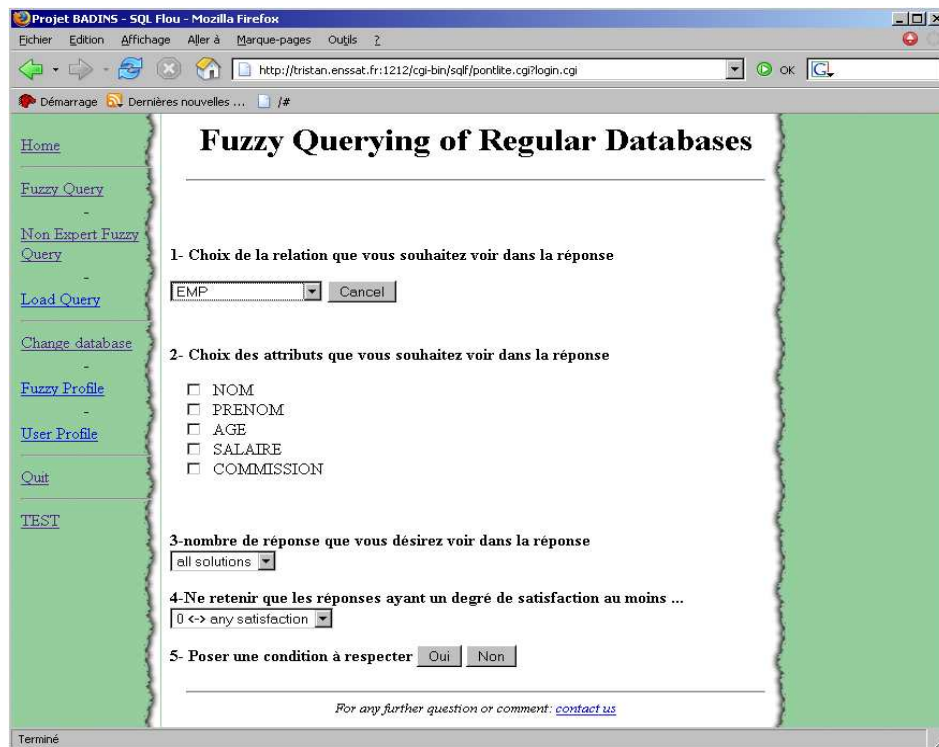


FIG. 13 Nouvelle interface pour non experts

Ces informations sont envoyées au fichier `chReq3.cgi`. Si l'utilisateur choisit de ne pas saisir de conditions ce fichier génère un page affichant la requête telle qu'elle sera envoyée au moteur et donne la possibilité à l'utilisateur de confirmer l'envoi ou de modifier la requête.

Si il choisit de poser une condition, il choisit ensuite le format de conditions de sa clause where. Il a été décidé de laisser le choix à l'utilisateur entre 4 types de format:

- une condition simple, c'est à dire ne faisant intervenir qu'un seul attribut.
- une condition conjonctive, c'est à dire du type *condition 1 AND...AND condition n*
- une condition disjonctive, c'est à dire du type *condition 1 OR...OR condition n*
- une condition utilisant un quantificateur comme *laplupart*

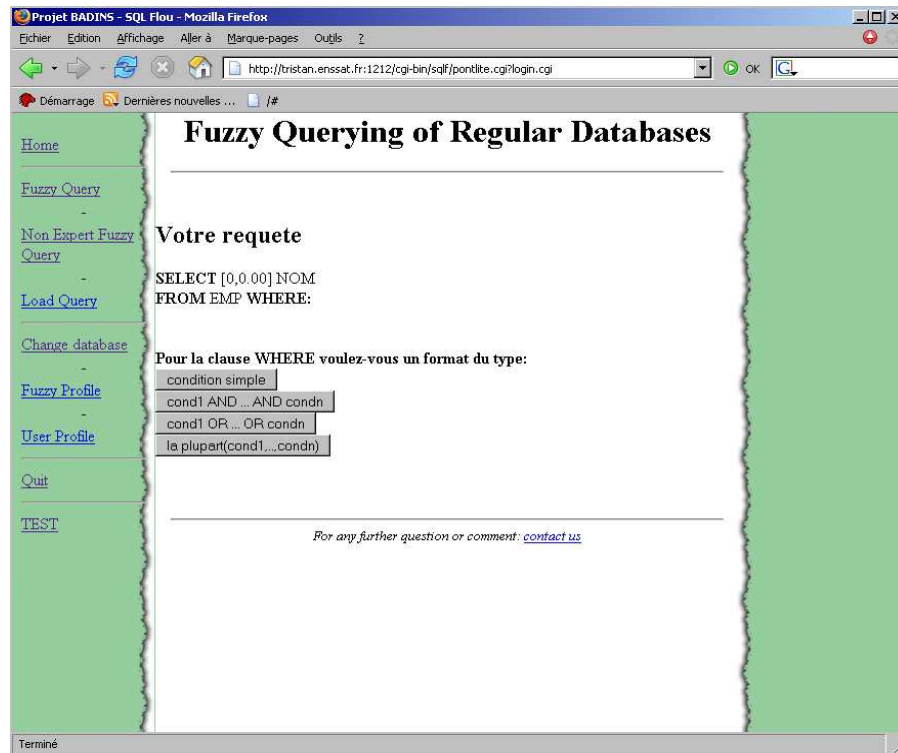


FIG. 14 Nouvelle interface pour non experts

Lorsque le type de la clause where a été choisi, la requête actuelle ainsi que le choix de l'utilisateur sont transmis au fichier chReq4.cgi

L'utilisateur peut ensuite choisir le 1er attribut intervenant dans la condition et une page différente est affichée en fonction de l'attribut sélectionné. Cette opération a été relativement difficile à implémenter.

Premièrement car lorsqu'un attribut flou est sélectionné, il est nécessaire de rafraichir la page. A chaque chargement de la page la fonction nbTF calcul le nombre de termes flous associés par l'utilisateur à l'attribut sélectionné. Par ailleurs si au moins un terme flou est défini, le CGI recupère la liste des termes flous et modificateur flou associé au terme.

Il est obligatoire de rafraichir la page pour réaliser cette opération car il n'est pas possible avec les technologies CGI en C/html de gérer de façon dynamique le contenu des listes déroulantes. En effet les pages HTML sont générées par des calculs des CGI. Cependant lorsque la page est affichée sur le navigateur, il est impossible de modifier son code. Il est donc nécessaire de la rafraichir.

La deuxième difficulté a été de rafraichir la page. Il n'existe pas de fonction pour rafraichir une page en HTML. Par ailleurs, un formulaire ne peut être associé qu'à une action effectuée.

j'ai donc dû créer le fichier chReq5.cgi. Ce fichier recupère un paramètre lui indiquant 3 situations différentes:

- rafraichir la page: il relance automatiquement le fichier chReq4.cgi en lui transmettant la requête ainsi que l'attribut préalablement sélectionné.
- confirmer l'envoyé de la requête au serveur : il génère une page d'affichage de la requête et de demande de confirmation d'envoie.
- compléter la requête: il relance automatiquement le fichier chReq4.cgi en lui transmettant la requête en cours.

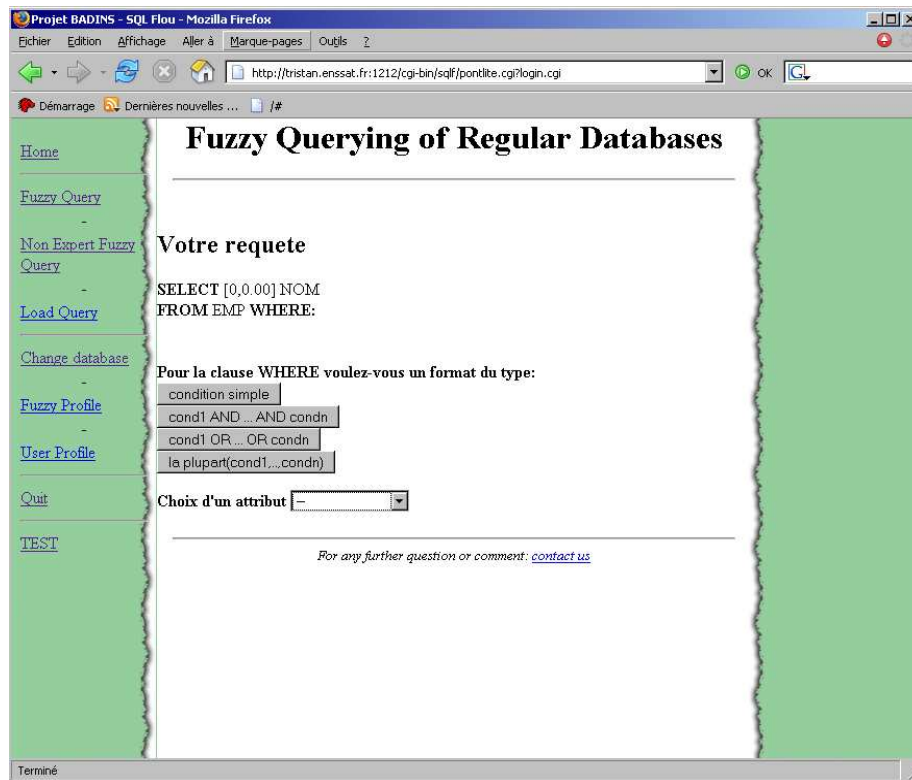


FIG. 15 Selection du 1er attribut de la clause where

Si l'attribut choisi est un attribut pour lequel aucun terme flou n'est défini dans le profil flou de l'utilisateur (c'est à dire si le resultat de la fonction nbTF est égal à 0) , celui ci n'aura la possibilité que de choisir un comparateur parmi les comparateurs usuels ($=$, $\neq$, $<$, $\leq$, $>$, $\geq$). L'utilisateur pourra saisir la valeur à comparer à l'attribut dans le champs *Valeur*. Ce champ n'est actif qu'une fois un comparateur choisi. Ceci permet d'éviter tout risque d'erreur de syntaxe.

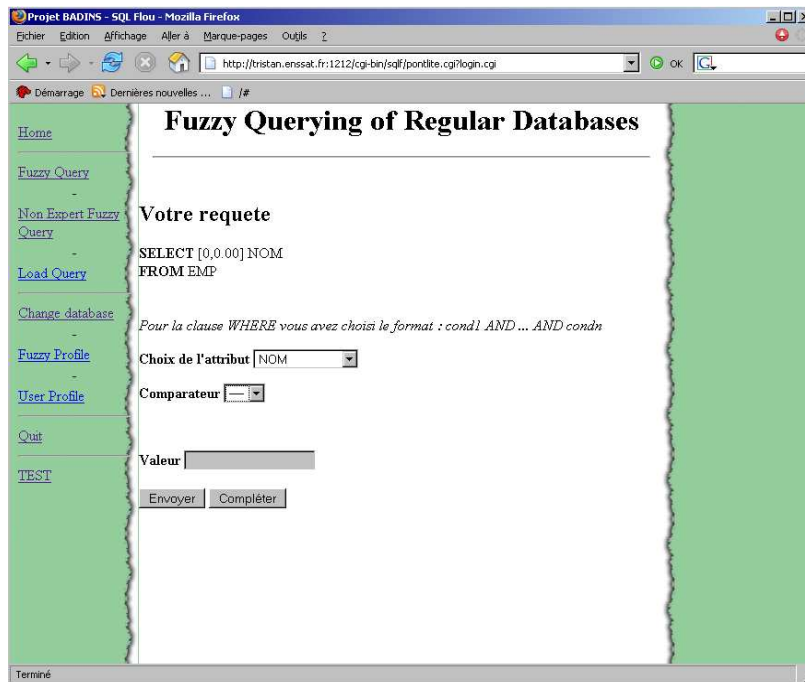


FIG. 16 Cas attribut sans terme flou

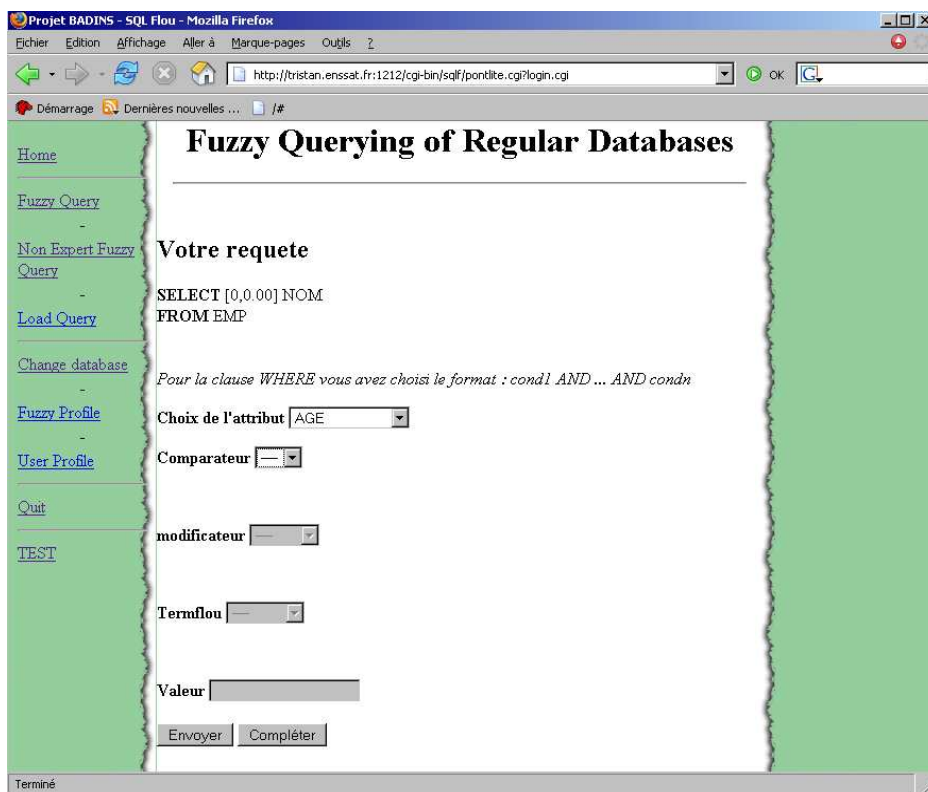


FIG. 17 Cas attribut avec terme flou

Si des termes flous ont été définis dans le profil flou de l'utilisateur, celui ci pourra choisir un comparateur usuel, mais également le comparateur spécial "is" qui est le comparateur de l'interface associé aux termes flous.

Tout comme les attributs sans terme flou, le champ valeur n'est actif que lorsque l'utilisateur a choisi un comparateur usuel. Si l'utilisateur choisi le comparateur "is" les champs de selection d'une modificateur flou et d'un terme flou seront rendus actifs. L'utilisateur ne pourra choisir un terme flou que parmi ceux qu'il a défini pour l'attribut selectionné. Ceci empêche l'utilisateur de faire une faute sémantique.

Avec cette nouvelle interface l'utilisateur ne peut pas saisir un terme flou et une valeur a comparer simultanément. Cela limite donc les risques d'erreurs syntaxiques.

La requête peut être ensuite envoyée au moteur pour être exécutée ou alors complétée.

Si l'utilisateur choisi de compléter la requête, il aura alors à choisir l'attribut suivant sur lequel il désire travailler.

Projet BADINS - SQL Flou - Mozilla Firefox

Fichier Edition Affichage Aller à Marque-pages Outils ?

http://tristan.enssat.fr:1212/cgi-bin/sqlf/pontlike.cgi?login.cgi

Démarrage Dernières nouvelles ... /#

Fuzzy Querying of Regular Databases

Votre requete

```
SELECT [0,0.00] NOM
FROM EMP
WHERE AGE=%JEUNE AND
```

Pour la clause WHERE vous avez choisi le format : cond1 AND ... AND condn

Choix de l'attribut

Comparateur

Valeur

Terminé

FIG. 18 2Eme condition de la clause where

Si l'utilisateur décide d'envoyer sa requête au moteur, celle ci est affichée telle qu'elle sera envoyée. L'utilisateur a alors le choix de soumettre sa requête ou alors de la modifier.

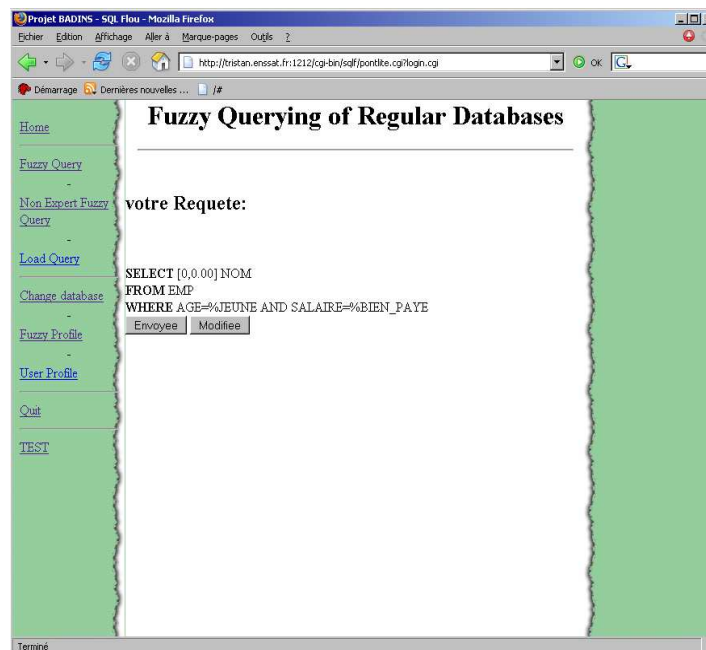


FIG. 19 Demande de validation d'envoi

Lors de la rédaction d'une requête dans le module pour utilisateurs non experts, la requête est en permanence visible sur la page. Cela permet à l'utilisateur de suivre le progression de la création de sa requête et de savoir à tout moment où il en est.

Cette nouvelle interface limite donc les risques d'erreurs de la part de l'utilisateur. Lors de son développement, j'ai pris conscience des limites de l'architecture actuelle de l'interface web. Les technologies utilisées (CGI en C et html) ont vieilli et n'offrent pas des fonctionnalités proposées par des technologies plus récentes comme JAVA. Cela a pour principale conséquence de ne pas avoir réellement de pages dynamiques. Effet une fois les pages HTML générée par les CGI, leur code source est statique et il est impossible de les modifier sans régénérer la page. Ceci entraîne un rafraichissement des pages lors de la selection d'un attribut. Par ailleurs l'obligation de devoir passer par des CGI intermédiaires entraîne un temps d'exécution important lors de l'évaluation d'une requête. L'architecture du module de l'interface WEB semble donc être à revoir pour permettre une plus grande souplesse dans le développement ainsi que pour améliorer les performances.

4. Correction des bugs persistants

4.1 Problèmes liés à la gestion du seuil

Plusieurs problèmes liés aux seuils ont été découverts lors de la phase de prise en main du logiciel.

Premièrement, le seuil choisi par l'utilisateur lors de la définition de la requête n'était pas pris en compte par le logiciel iSQLf. Celui-ci générait une page résultat comportant tous els n-uplets de degré supérieur à 0 et ce même si il désirait un seuil minimum de 0,9.

Le problème provenait de la selection des n-uplets à afficher dans le résultat. La page résultat est générée par le fichier perfreq.cgi. Celui-ci récupérait le calibre, la requête mais pas le seuil. Aucun traitement n'était donc réaliser pour filtrer les n-uplets en fonction de leur degré. J'ai utilisé la fonction trouverChamps précédemment utilisé pour accéder au calibre et à la requête, afin de récupérer le seuil. Un filtre a été ensuite codé afin de ne garder pour l'affichage que les réponses dont le degré était satisfaisant.

Le second problème était que lorsque l'utilisateur sélectionnait un seuil nul, il n'obtenait pas tous les attributs comme souhaité. Il était même possible d'obtenir une réponse vide avec un seuil égal à 0. Les réponses n'ont prises en compte étaient celle dont les valeurs étaient extérieurs aux bornes du ou des trapézoïdes. Cela provient donc d'un problème d'interprétation. Les étudiants ayant travaillé sur ce projet les années précédente avaient jugé que ces valeurs n'étaient pas significatives et n'avaient donc pas à être ajoutées. Ils traduisaient donc les clauses where floues en clauses booléenne. Après discussion avec M. Pivert, il a été décidé que cela ne correspondait pas au concept de seuil nul. En effet si l'utilisteur choisi un seuil nul, c'est qu'il souhaite voir afficher toutes les réponses quelque soit leur degré de satisfaction.

Il a donc fallu recoder la quasi intégralité du fichier effectuant les dérivations lors de l'analyse syntaxique, afin qu'il ne traduise plus les clauses floues et qu'il n'effectue plus qu'un traitement de détermination du degré.

4.2 Chargement des requêtes sauvegardés

Lors du chargement d'une requête préalablement sauvegardée par l'utilisateur, le logiciel n'effectuait aucun traitement et affichait une page blanche. Après avoir identifié les fonctions intervenant lors du chargement d'une requête sauvegardée j'ai pu déterminer l'origine de ce problème. L'utilisateur commence par choisir une requête à relancer parmi celle présente dans son profil. Celle ci est ensuite envoyé au fichier perfreq.cgi qui récupère le calibre, le seuil et identifie les différentes clauses de la requête avant de les transmettre au moteur. Le problème était que la fonction réalisant se traitement dans le cas d'une requête sauvegardée était présente dans le CGI mais n'était pas implémentée.

Il a donc fallu modifier le CGI pour implémenter cette fonction dans le cas du rechargement d'une requête (les traitements à effectuer lors de la création d'une fonction en mode expert ou non expert sont différents). Une fois cette modification effectuée une autre erreur est apparue. Il semblait que la fonction implémentée ne parvenait pas à identifier les différentes clauses de la requête. Il s'agissait d'un problème de casse. En effet les requêtes sont sauvegardées et envoyées au moteur sous forme majuscule pour des raisons de compatibilité avec la grammaire. Afin d'identifier les différentes clauses de la requête, la fonction parcourait celle ci à la recherche des termes "select", "from" et "where". Aucun résultat n'était renvoyé car les requêtes sont sauvegardé en majuscule. Cette erreur corrigée, le module de chargement d'une requête sauvegardée par

l'utilisateur fonctionne parfaitement.

4.3 Opérateur NOT pour des requêtes classiques

L'opérateur NOT ne fonctionnait pas avec des requêtes classiques. En effet associé à une comparaison booléenne, il renvoyait systématiquement un résultat nul. Le problème provenait de la dérivation et des traitements effectués lors de la traduction de la requête. En effet la fonction dérivationNot effectuait le même traitement pour un terme flou et pour une valeur numérique. C'est à dire qu'il faisait appel à la même fonction de calcul du degré avant d'en déterminer le complément. Or pour une comparaison avec une valeur numérique, le calcul du degré n'est pas identique à celui d'un terme flou. Il a donc fallu déterminer le type de condition afin de réaliser un traitement différent en fonction des cas.

4.4 Interdire le flou sur des attributs de type chaîne de caractères

Lors de la rédaction d'une requête, l'utilisateur pouvait utiliser un terme flou avec un attribut de type chaîne de caractère. Il obtenait alors des clauses where comme where nom=%jeune ce qui est incompréhensible par le SGBD. En effet lors de l'évaluation du degré de satisfaction d'un attribut, la valeur de celui-ci est comparée aux valeurs A, B, C, D du trapézoïdes définissant le terme flou. Ces valeurs étant des valeurs numériques, les comparer à une chaîne de caractères (leur code ASCII après conversion implicite) n'a pas vraiment de sens. Il est donc souhaitable d'interdire l'utilisation de termes flous avec les attributs de type chaînes de caractères.

Ce problème a été partiellement résolu avec la nouvelle interface pour utilisateurs non experts. Il est impossible d'associer un terme flou avec un autre attribut que celui pour lequel il a été défini. Cependant l'utilisateur peut toujours définir et associer un terme flou à un attribut de type chaîne de caractères.

Il faudrait dans le profil flou de l'utilisateur, lors de la création d'un terme, récupérer le type des attributs des différentes tables et les filtrer en fonction de leur type. Il ne serait alors possible d'associer à un terme flou qu'à un attribut numérique (entier ou flottant).

Cependant cette fonctionnalité n'a pas pu être développée. J'ai rédigé la requête SQL permettant de récupérer le type d'un attribut dans la table user_tab_columns qui contient la description des tables et attributs accessibles par l'utilisateur. La requête est :

```
select data_type from user_tab_columns where table_name=$nom_de_table and  
column_name=$nom_de_l'attribut;
```

Cependant, bien que renvoyant un résultat correcte lorsque exécutée en ligne de commande sous Oracle, cette requête ne fonctionne pas lorsque appelée depuis l'interface web de iSQLf. Après de nombreuses recherches, je n'ai pas réussi à identifier l'origine de cette erreur.

4.5 Problème lié aux attributs présents dans la clause WHERE

Si les attributs de la clause where n'étaient pas inclus à la clause select, le traitement du fichier Pro*C par le SGBD entraînait une erreur. Ce problème était lié à la différence entre les requêtes floues et les requêtes booléennes. En effet dans le cas d'une requête booléenne classique, un n-uplet est sélectionné par le SGBD si sa valeur satisfait les conditions de la clause where. Dans le cadre d'une requête floue, les n-uplets sont renvoyés par le SGBD et ensuite la surcouche du moteur de iSQLf réalise un traitement afin de déterminer le degré de satisfaction de chacun d'entre eux. Le

problème est donc que si les attributs intervenant dans la clause where, ne sont pas présent dans le select, le moteur n'aura pas accès à leur valeur qui lui sont nécessaires au calcul du degré.

Afin de résoudre ce problème lors de la dérivation de chaque condition de la clause where, les attributs y intervenant sont sauvegardés dans une variable att. Les attributs de la clause select sont sauvegardés puis l'ensemble des attributs de att sont ajoutés à la clause select; le moteur disposera ainsi des valeurs nécessaires au calcul des degrés. Lors de l'affichage des résultats, on ne conservera que les attributs du select initialement sauvegardés.

4.6 Opérateur “=” associé à des termes non flous

Lorsque on utilise l'opérateur “=” avec un terme non flou, le programme génère une page d'erreur indiquant la détection d'une erreur de syntaxe dans l'écriture de la clause where. Les clauses where ayant une syntaxe correcte (elle fonctionne sous Oracle). Le problème provient donc de la grammaire lex and yacc utilisée lors de l'analyse syntaxique. L'analyse du fichier de log du programme nous informe que l'erreur est survenue au moment de la traduction de la requête floue en arbre de dérivation. Cette étape est réalisée par la traduction sql1.y. Cependant, par manque de temps je n'ai pas pu corriger cette erreur. Il semblerait qu'il y ait un problème d'interprétation de l'opérateur “=”. Celle-ci n'étant pas la même dans le cas d'une valeur numérique et dans le cas d'un terme flou. En effet l'opérateur = fonctionne parfaitement associé à un terme flou.

Pour résoudre ce problème il pourrait être intéressant d'étendre l'opérateur “is” utilisé pour les termes flous à la grammaire. Il n'y aurait donc ainsi plus d'ambiguïté sur l'opérateur utilisé.

5. Perspectives

5.1 Amélioration de l'interface

L'interface actuelle utilise les technologies HTML et CGI en C. Créer au début des années 90 les CGI en C offrent moins de fonctionnalités que des langages de développement plus récents et moins de souplesse. L'utilisation des langages HTML et C pour les CGI permet de générer des pages en fonction des paramètres reçus. Cependant ce ne sont pas véritablement des pages dynamiques car il est impossible de modifier les sources d'une page (pour mettre à jour une liste déroulante par exemple) une fois la page générée. Cela oblige à rafraîchir la page pour la recalculer.

Il pourrait donc être intéressant de réaliser une refonte totale de l'interface, afin d'améliorer encore son ergonomie tout en utilisant des technologies plus récentes et plus adaptées à des applications webs telle que iSQLf.

Une solution serait d'utiliser des interfaces en client riche comme XUL. XUL, est un langage basé sur XML pour décrire une interface graphique, utilisé dans Firefox et les autres logiciels Mozilla. Il possède ainsi toute une série de balises correspondantes à des boutons, des listes, des menus, des listes, treeviews, zones d'éditations etc... Il propose donc toutes les fonctionnalités pour faire une véritable interface utilisateur.

En résumé, HTML est fait pour afficher de l'information et naviguer dans cette information (grâce aux liens hypertextes). XUL est fait pour réaliser des interfaces utilisateurs, pour réaliser une tâche précise. HTML est donc plus adapté pour réaliser un site web alors que XUL est plus adapté pour réaliser des applications web comme c'est le cas d'iSQLf.

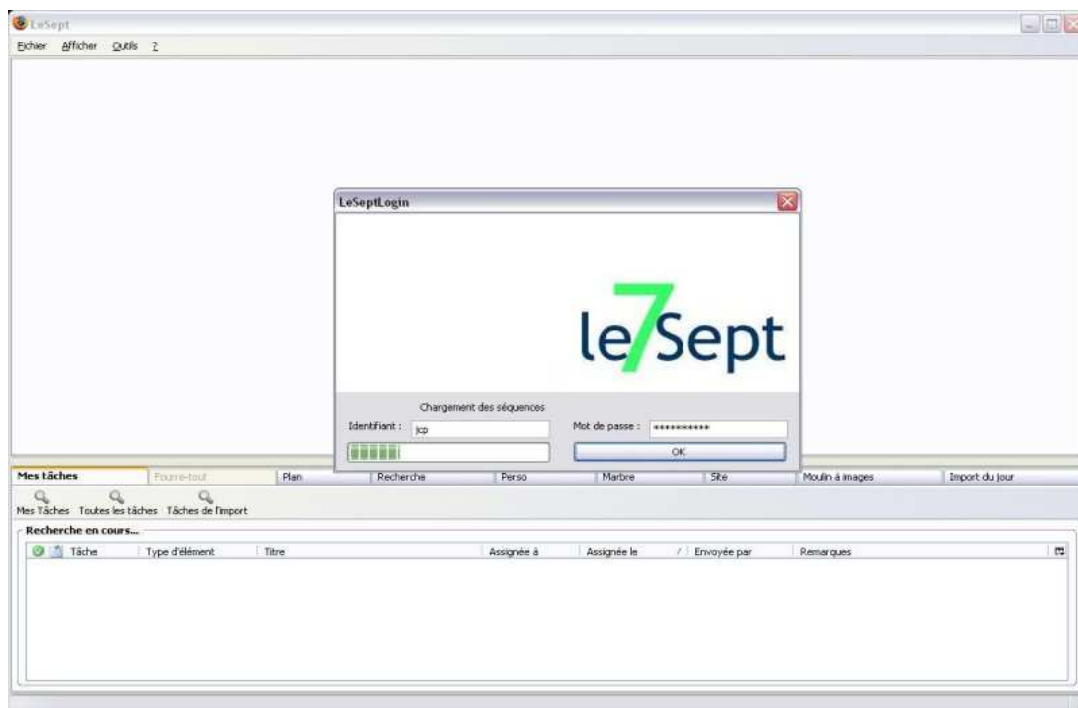


FIG. 20 Exemple d'interface XUL

Choisir XUL pour réaliser la refonte de l'interface WEB aurait donc plusieurs avantages:

- Le code XUL correspond à du code XML enseigné à l'ENSSAT. L'étudiant reprenant le projet n'aurait donc pas à réserver du temps dans son planning pour l'apprentissage du langage.

- Cela permettrait d'inclure iSQLf comme extension de firefox ou mozilla et d'avoir une interface beaucoup plus souple.
- Les langages utilisés dans les applications Mozilla sont pour la plupart des standards recommandés par le W3C (CSS, RDF, XBL, XML etc..) ou par d'autres comme Ecma (Javascript 1.5 repose sur EcmaScript). Ce sont des langages utilisés un peu partout. Ils ne sont pas spécifiques à Mozilla. Si on est habitué à les utiliser, apprendre à créer des applications Mozilla peut être assez rapide.
- Il n'y a pas besoin d'outils lourds et coûteux pour développer une application Mozilla. Dans l'absolu, un simple éditeur de texte suffit.
- Une application XUL est automatiquement multiplateforme : son code fonctionnera sur les différentes versions de Mozilla : Windows, Linux, [MacOs](#)... Il n'y a pas à écrire de code spécifique à une plateforme, même lorsque l'on programme des composants XPCOM en C++ . (Dans la mesure où l'on respecte bien sûr les règles de portabilités préconisées par la fondation Mozilla).
- Présence de documentation complète en français accessible sur internet.

XUL permettrait par ailleurs de réaliser la refonte de l'interface sans être obligé de modifier la configuration du serveur WEB

5.2 Amélioration des performances

Au cours de la phase de prise en main du logiciel et des différents tests réalisés est apparu un problème au niveau du temps d'exécution des requêtes. On peut estimer à une dizaine de secondes le temps nécessaire au calcul, génération et affichage d'une réponse après la validation d'une requête. Par ailleurs lors de la phase de correction des erreurs, il s'est avéré très difficile de déterminer l'origine d'un bug. Ceci est dû au grand nombre de fichiers et de modules intervenant dans les différents traitements réalisés par iSQLf et au nombre de langages utilisés. Il est difficile de récupérer les messages d'erreur générés par les scripts Pro*C ou les javascripts.

Une solution pour résoudre ce problème serait de passer d'une interface web comme c'est le cas actuellement à une application web. Le langage de programmation idéal pour réaliser cette refonte est Java. Il a remplacé les CGI en C dans de nombreuses applications web ces dernières années. Il offre de nombreux avantages :

- il permettrait l'utilisation d'un langage unique pour l'interface.
- il est portable sur plusieurs systèmes d'exploitation et est indépendant de l'architecture de la machine hôte.
- il permet une approche objet. Cela permettrait donc de ne plus avoir à utiliser de pointeurs. Ceux ci sont non testés par défaut et peuvent entraîner des erreurs difficiles à détecter.
- il offre de nombreux composants graphiques et de gestion des données fournis en standard qui simplifieraient le développement.
- enseigné à l'enssat en 2eme année l'étudiant reprenant ce projet n'aurait pas à se former à ce langage.
- contrairement au langage C, ce langage a été conçu pour le développement d'applications internet. Il est donc plus adapté pour la réalisation d'un logiciel tel que iSQLf.
- il offre la possibilité de concevoir une interface personnalisée et autonome(il ne serait plus obligatoire d'utiliser un navigateur web).

Il faudrait ensuite traduire la grammaire. En effet les outils lex et yacc ne sont pas compatibles avec le langage JAVA. Les analyseurs syntaxiques utilisés pour les applications Java sont :

-Jflex et CUP. Le premier est un analyseur lexical et le second un analyseur grammatical. Equivalent de lex et yacc ils sont utilisés pour générer un analyseur syntaxique. Ils permettraient une traduction rapide de la grammaire.

-ANTLR (another tool for language recognition) c'est un utilitaire permettant de réaliser un analyseur syntaxique à partir d'une grammaire spécifique. Il offre l'avantage de gérer aussi bien le code Java que C. Il serait donc possible de traduire la grammaire et de la tester avec l'ancienne interface. ANTLR offre par ailleurs une très bonne gestion des erreurs

5. Planning récapitulatif du travail effectué

Voici un résumé du planning de travail de l'année entière :

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
A																											
B																											
C																											
D																											
E																											
F																											
G																											
H																											
I																											
J																											
K																											

Légende des tâches

A= Prise en main du logiciel
 B=Etat des lieux et identification des différents bugs
 C= résolution du problème de compilation
 D=Définition de la nouvelle interface
 E=conception et implémentation de la nouvelle interface
 F=Correction des différentes erreurs persistantes
 G=Tests
 H=Recherche et définition des perspectives du projet
 I=Fiche de projet
 J=Préparation Oral
 K=Rapport final

Conclusion

L'interrogation flexible de bases de données fait partie des travaux de l'équipe de recherche Badins du L.L.I. Il y a 7 ans, à débiter un projet visant à mettre en oeuvre le langage SQLf. Un prototype a été développé à cette époque, mais il présentait l'inconvénient de n'être accessible que via une connexion au serveur Tristan et de ne pouvoir interroger qu'une seule base de données. Depuis 2000, une architecture web existe et permet d'accéder au logiciel iSQLf depuis n'importe quel système.

Néanmoins, même si cette architecture fonctionne globalement, plusieurs disfonctionnements persistent. Ces erreurs ont été considérées suffisamment critiques pour qu'il soit nécessaire de les corriger avant de poursuivre le développement. Par ailleurs l'interface n'était pas adaptée à l'ajout de nouveaux modules comme celui chargé de la gestion des requêtes imbriquées et l'interface pour utilisateurs non experts n'était pas suffisamment intuitive et facilitait les erreurs de syntaxe.

Afin de résoudre ses problèmes, de nouvelles interfaces expert et non expert ont été réalisées. Le code du moteur a également été modifié afin de corriger les bugs persistants.

iSQLf est désormais un logiciel pleinement fonctionnel pour l'exécution de requêtes simples. Cependant avant de poursuivre le développement et d'ajouter de nouveau module il serait préférable de changer les technologies utilisés pour le développement du logiciel afin d'améliorer son ergonomie ainsi que ses performances.

Pour ce qui est du bilan personnel sur l'année, il est fortement mitigé. La phase de découverte du logiciel a été grandement facilitée par la documentation rédigée par mon prédécesseur. Cependant, elle a été relativement longue car l'architecture en couches et en modules du logiciel rend sa prise en main difficile. Ceci, ainsi que le manque de clarté des messages d'erreurs, a par ailleurs fortement ralenti la correction des différents problèmes. Par ailleurs un problème au niveau de la compilation en début de projet m'a fait prendre du retard et m'a obligé à revoir mes objectifs à la baisse. Je n'ai donc pas pu réaliser le module de gestion des requêtes imbriquées, objectif qui m'avait motivé à choisir ce projet.

Ce projet m'a tout de même beaucoup apporté. En effet, le fait de d'avoir eu à plonger dans la totalité des sources du projet m'a permis de manipuler des langages et thèmes variés : javascript, lex & yacc, c cgi, Pro*c, Sql, la logique floue, etc.

Par ailleurs il m'a fait prendre conscience des difficultés inhérentes à la maintenance et à la reprise du développement d'un logiciel et de l'importance de respecter des normes de programmation.

Références

1. Julien Boulet *Interrogation de Bases de données : extension de iSQLf*. Rapport de projet, 2004
2. Johann L'HOURL *Interrogation floue de base de données : extension de iSQLf*. Rapport de projet, 2002.
3. Fabien VANNIER *Interrogation floue de base de données*. Rapport de projet, 2001
4. Didier DAUL *Serveur Web pour interrogation flexible de bases de données*. Rapport de projet, 2000.
5. Mathias Salle. *Couche logicielle pour la dérivation de requêtes floues. Mise en oeuvre du langage SQLf*. Rapport de projet ENSSAT, 1998.
6. Iman El Aouni. *Couche logicielle pour la dérivation de requêtes floues. Extension de la mise en oeuvre du langage SQLf*. Rapport de projet ENSSAT, 1999.
7. Patrick BOSC, Olivier PIVERT, Daniel ROCACHER, Ludovic LIETARD. *Gradualité et imprécision dans les bases de données*. Ellipse, 2004, ISBN: 2729820361
8. ENSSAT. Le laboratoire lannionnais d'informatique.
<http://li.enssat.fr/>
9. Présentation de XUL et tutorial
<http://xulfr.org/>
10. Documentation et présentation de Jflex
<http://jflex.de/>
11. Documentation et présentation de CUP
<http://www.cs.princeton.edu/%7Eappel/modern/java/CUP/>
12. Documentation et présentation de ANTLR
<http://www.antlr.org/>

Annexes

A)requete.c

```
/* *****  
* Nom      : requete.c  
* Chemin   : /Cgi  
* Objet    : Construction d'une nouvelle requete  
* Date     : 12 nov 99  
* Auteur   : Didier Daul  
* *****  
  
* Historique des modifications :  
* -----  
*  
* ***** */  
  
#ifndef __C_DEF_H__  
#define __C_DEF_H__  
#include "../Config/c_def.h"  
#endif  
  
void main ()  
{  
    s_champ      *cookies ;          /* tableau des cookies decodes */  
    s_champ      cook ;             /* structures ccookie temporaires */  
    int          nbCookie ;         /* nombre de cookies pour l'utilisateur */  
  
    /* Initialisation des pointeurs */  
    cook.nom = (char *)malloc (MAXGETNOM) ;  
    cook.val = (char *)malloc (MAXGETVAL) ;  
  
    /* Decodage de la liste des cookies */  
    decodeCookie (&nbCookie, &cookies) ;  
    if ( ! trouveCookie ( "pseudo", cookies, nbCookie, &cook ) )  
        erreur ("Impossible de determiner l'identite", 604) ;  
  
    entetePage () ;  
  
    /* controle en javascript pour eviter l'envoi d'un champs requête vide */  
    printf ("<script language=javascript>\n\  
function fnOk( )\n\  
{\n\  
var req = document.newReq.i_requete.value ;\n\  
    if (req == \"\")\n\  
        alert (\"il faut remplir le champ REQUETE\");\n\  
    else\n\  
    {\n\  
        document.newReq.submit();\n\  
    }\n\  
}\n\  
</script>\n\n") ;  
  
    /* la page elle-meme */  
    printf ("<form name=\"newReq\" method=post  
action=\"pont.cgi?perfreq.cgi_3\">\n") ;  
    printf ("<table width=90%% border=1 cellspacing=-4 cellpadding=4>\n") ;  
    printf ("<tr align=center valign=center>\n<td width=30%%  
bgcolor=\"#afd5fa\"><b>Calibre</b></td>\n") ;  
    printf ("<td width=75%%><input type=text size=3
```

```

name="\i_calibre\"></td>\n</tr>\n") ;

printf("<tr align=center valign=center>\n<td width=30%%
bgcolor=\"#afd5fa\"><b>Seuil</b></td>\n") ;
printf("<td width=75%%><input type=text size=3
name=\"i_seuil\"></td>\n</tr>\n") ;

printf("<tr align=center valign=center>\n<td width=30%%
bgcolor=\"#afd5fa\"><b>Requete</b></td>\n") ;
printf("<td width=75%%><textarea name=\"i_requete\" cols=30 rows=10>
</textarea> &nbsp;  </td>\n</tr>\n") ;

printf("<tr align=center valign=center>\n<td width=30%%
bgcolor=\"#afd5fa\"><b>Action</b></td>\n") ;
printf("<td width=75%%><table border=0>\n<tr>\n<td><input type=button
name=\"b_ok\" value=\"Lancer\" \">
printf("<td><input type=button name=\"b_quit\"
value=\"Annuler\" \">
printf
("<td>\n</tr>\n</table>\n") ;
printf("</td>\n</tr></table>\n\n<br>\n</form>\n") ;

piedDePage () ;
}

```

B)chReq1.c

```

/* *****
* Name : chReq1.c
* Path : /Cgi
* Object : Non Expert Fuzzy Query 1
* Date : 10 nov 2000
* Author : Fabien Vannier
*****

*****
* Modifications :
* -----
*
***** */

#ifndef __C_DEF_H__
#define __C_DEF_H__
#include "../Config/c_def.h"
#endif

int main (void)
{
    /* Declarations */
    char login[MAXGETNOM] ;
    char base[MAXGETNOM] ;
    char passwd[MAXGETNOM] ;
    s_tables *listTab ;
    int nbTables = 0 ;
    int cptr ;

    /* Pointer's init */
    bzero (login, MAXGETNOM) ;
    bzero (base, MAXGETNOM) ;
    bzero (passwd, MAXGETNOM) ;

```

```

openCnxAdmin();

/* Analyse of the cookies */
strcpy (login, trouveEltDsCookie ( "pseudo" )) ;
strcpy (base, trouveEltDsCookie ( "base" )) ;
strcpy (passwd, trouveEltDsCookie ( "passwd" )) ;

/* Search of all the tables on the user's database */
nbTables = listTables (base, passwd, &listTab) ;

if ( nbTables )
{
    printf("<SCRIPT language=\"Javascript\">\n");

    printf("function lancer()\n {\n");
    printf("if (document.f_table.i_from.selectedIndex == 0)\n {\n");
    printf("alert (\\"Aucune table sélectionnée!\");\n");
    printf("document.f_pastable.submit();\n}\n");
    printf("else\ndocument.f_table.submit();\n}\n");

    printf("</SCRIPT>\n\n");

    printf("</SCRIPT>\n\n");
    printf("<P> <B> 1- Choix de la relation que vous souhaitez voir dans
la réponse ") ;
    printf("</B></P>\n");

    printf("<FORM name=\"f_table\" method=\"post\"
action=\"pontliteBp.cgi?chReq2.cgi\" target=\"bas\">\n");

    printf("<SELECT name=\"i_from\" onChange=\"lancer()\">\n");
    printf("<OPTION value=\"null\">--\n");

    for ( cptr=0 ; cptr < nbTables ; cptr++ )
        printf("<OPTION value=\"%s\">%s\n", listTab[cptr].table,
listTab[cptr].table);
    printf("</SELECT>\n");
    printf("<INPUT type=\"button\" value=\"Cancel\"
onClick=\"parent.history.go(-1)\">") ;

    printf("</FORM>\n\n");

    printf("<FORM name=\"f_pastable\" action=\"/sqlf/modacc_r2.htm\"
target=\"bas\">\n");
    printf("</FORM>\n\n");
}
else
{
    printf("There is no table in this database...<BR>\n");
    printf("Sorry, but you can't make a query.\n<BR>");
}

closeCnxAdmin();
}

```

C)chReq2.c

```

/* *****
* Name      : chReq2.c
* Path      : /Cgi
* Object    : Non Expert Fuzzy Query 2
* Date      : 10 nov 2000
* Author    : Fabien Vannier
*****

```

```

*****
* Modifications :
* -----
* 2005 Samuel Bartel
* refonte de l'interface non expert
* correction des valeurs possibles pour le seuil (le 0 correspondait à 0.5
***** */

#ifndef __C_DEF_H__
#define __C_DEF_H__
#include "../Config/c_def.h"
#endif

int main (void)
{
    /* Declarations */
    s_champ champs[MAXCHAMPS] ;
    s_champ unChp ;
    int nbChamps ;
    int cptr ;
    int nbAttrib=0 ;
    char login[MAXGETNOM] ;
    char base[MAXGETNOM] ;
    char passwd[MAXGETNOM] ;
    char table[MAXLONGCOOKIE] ;
    s_att *listAtt;

    /* Pointer's init */
    unChp.nom = (char *)malloc (MAXGETNOM) ;
    unChp.val = (char *)malloc (MAXGETVAL) ;
    bzero (login, MAXGETNOM) ;
    bzero (base, MAXGETNOM) ;
    bzero (passwd, MAXGETNOM) ;

    openCnxAdmin () ;

    /* Analyse des cookies */
    strcpy (login, trouveEltDsCookie ( "pseudo" )) ;
    strcpy (base, trouveEltDsCookie ( "base" )) ;
    strcpy (passwd, trouveEltDsCookie ( "passwd" )) ;

    decodeFormulaire (&nbChamps, champs) ;
    trouveChamp ("i_from", champs, nbChamps, &unChp) ;
    strcpy (table, unChp.val) ;

    /* recherche de tous les attributs de la page*/
    nbAttrib = listAttributs(base, passwd, table, &listAtt) ;

    /* Fonction d'envoi du formulaire*/
    printf("<SCRIPT language=\"javascript\">\n\
        function fnOk2( )\n\
        {\n\
            document.f_choix.submit();\n\
        }\n\
    </SCRIPT>");

    /* Fonction javascript verifiant que tous les champs sont remplis */
    printf ("<SCRIPT language=\"javascript\">\n\
        function fnOk(type)\n\
        {\n\
            var nbr = 0 ;\n\
            var fro = document.f_choix.i_from.value ;\n\
            var cal = document.f_choix.i_calibre.value ;\n\
            var seu = document.f_choix.i_seuil.value ;\n\
        }");

```

```

for ( cptr=0 ; cptr < nbAttrib ; cptr++ )
    printf("\n\
        if ( document.f_choix.%s.checked )\n\
            nbr++ ;\n", listAtt[cptr].att) ;
printf("\n\
    if ( nbr == 0 )\n\
        alert (\"You have to choose at least one attribute!\");\n\
    else\n\
    {\n\
        if ( fro == \"\" )\n\
            alert (\"Problem detected with FROM. Sorry! Go back!\");\n\
\n\
        else\n\
        {\n\
            if ( cal == \"\" )\n\
                alert (\"Problem detected with CALIBRE. Sorry! Go\n\
back!\");\n\
\n\
            if ( seu == \"\" || seu > 1)\n\
                alert (\"Problem detected with THRESHOLD. Sorry!\n\
Go back!\");\n\
\n\
            document.f_choix.i_from.value = fro.toUpperCase();\n\
            if (type == \"1\") {\n\
                document.f_choix.i_cond.value=\"1\";\n\
                document.f_choix.submit();\n\
            }\n\
            else {\n\
                document.f_choix.i_cond.value=\"2\";\n\
                document.f_choix.submit();\n\
            }\n\
        }\n\
    }\n\
}");

printf ("function champSelect()\n\
{\n\
    var chaine;\n\
    for ( cptr=0 ; cptr < nbAttrib ; cptr++ )\n\
        printf("\t if (document.f_choix.%s.checked) chaine += %s ;\n\
\n", listAtt[cptr].att, listAtt[cptr].att);

    printf ("alert(\"The string length is \" + chaine.length)\n\");
    printf ("alert(\"The string last car is \" + chaine[chaine.length])\n\");

    printf ("\t return chaine\n\");
    printf ("\n</SCRIPT>\n\n");

printf("<P><B>") ;
printf("2- Choix des attributs que vous souhaitez voir dans la réponse");
printf("</B>\n");

/* Nom des attributs de la table choisie */

printf("<FORM name=\"f_choix\" method=\"POST\"
action=\"pont.cgi?chReq3.cgi\" target=\"work\">") ;

/* nom de la talbe choisie */
printf("<INPUT type=\"hidden\" name=\"i_from\" value=\"%s\">", table);

/* Choix des attributs de la clause select*/

```


D)chReq3.c

```
/* *****
 * Name      : chReq3.c
 * Path      : /Cgi
 * Object    : Non Expert Fuzzy Query 3 ; prepare the request before
               starting with 'perfreq'
 * Date     : janv 2006
 * Author    : Samuel Bartel
 * *****

 * Modifications :
 * -----
 *
 * ***** */

#ifndef __C_DEF_H__
#define __C_DEF_H__
#include "../Config/c_def.h"
#endif

int main (void)
{
    /* Declarations */
    s_champ champs[MAXCHAMPS] ;
    s_champ unChp ;
    int nbChamps ;
    int nbAttrib=0 ;
    int cptr, cptr2, cptr3 ;
    int no, ft, fc, fm, fq ;
    int nbTF, nbMF, nbCF, nbQF ;
    char login[MAXGETNOM] ;
    char base[MAXGETNOM] ;
    char passwd[MAXGETNOM] ;
    char i_from[MAXLONGCOOKIE] ;
    char i_calibre[MAXLONGCOOKIE] ;
    char i_seuil[MAXLONGCOOKIE] ;
    char i_select[MAXLONGCOOKIE] ;
    char att[MAXLONGCOOKIE] ;
    char i_cond[MAXLONGCOOKIE] ;
    char i_att[MAXLONGCOOKIE] ;
    s_att *listAtt ;
    s_tf *listTermes ;
    s_mf *listModifs ;
    s_cf *listCompas ;
    s_qf *listQuants ;

    /* Pointer's init */
    unChp.nom = (char *)malloc (MAXGETNOM) ;
    unChp.val = (char *)malloc (MAXGETVAL) ;
    bzero (login, MAXGETNOM) ;
    bzero (base, MAXGETNOM) ;
    bzero (passwd, MAXGETNOM) ;

    openCnxAdmin () ;

    /* Analyse des cookies */
    strcpy (login, trouveEltDsCookie ( "pseudo" )) ;
    strcpy (base, trouveEltDsCookie ( "base" )) ;
    strcpy (passwd, trouveEltDsCookie ( "passwd" )) ;

    /* Récupération des données du précédent formulaire */
    decodeFormulaire (&nbChamps, champs) ;
}
```

```

trouveChamp ("i_from", champs, nbChamps, &unChp) ;
strcpy (i_from, unChp.val) ;
trouveChamp ("i_calibre", champs, nbChamps, &unChp) ;
strcpy (i_calibre, unChp.val) ;
trouveChamp ("i_seuil", champs, nbChamps, &unChp) ;
strcpy (i_seuil, unChp.val) ;
trouveChamp ("i_cond",champs,nbChamps,&unChp);
strcpy(i_cond, unChp.val);

nbAttrib = listAttributs(base, passwd, i_from, &listAtt) ;

if (trouveChamp ("i_select", champs, nbChamps, &unChp))
    strcpy (i_select, unChp.val) ;
else
{
    /* We are finding all the attributes of this table i_from in order
to transmit them to the next CGI -> i_select */

    no = 0 ;
    strcpy(i_select, "") ;
    for ( cptr=0 ; cptr < nbAttrib ; cptr++ )
    {
        if (trouveChamp (listAtt[cptr].att, champs, nbChamps, &unChp))
        {
            no++ ;
            if (no > 1)
                strcat(i_select, ", ") ;
            strcat(i_select, unChp.val) ;
        }
    }
}

printf("<STYLE>\n");
printf(".cache{visibility :hidden}");
printf(".visible{visibility :visible}");
printf("</STYLE>\n");

/* JavaScript function which verify each answer of the query */
printf ("<SCRIPT language=\"javascript\">\n\
function fnOk( )\n\
{\n\
    var sel = document.f_choix.i_select.value ;\n\
    var fro = document.f_choix.i_from.value ;\n\
    var whe = document.f_choix.i_where.value.split(\"'\"); //pour
reperer les attributs chaines\n\
    var cal = document.f_choix.i_calibre.value ;\n\
    var seu = document.f_choix.i_seuil.value ;\n\
    var wherefinal =\"\";\n\
    var i;\n\
    if ( sel == \"\" )\n\
        alert (\"Problem detected with SELECT. Sorry ! Go back !\");
\n\
    else\n\
    {\n\
        if ( fro == \"\" )\n\
        {\n\
            alert (\"Problem detected with FROM. Sorry ! Go back !
\n) ;\n\
            return false ;\n\
        }\n\
        else\n\
        {\n\
            if ( cal == \"\" )\n\
            {\n\
                alert (\"Problem detected with CALIBRE. Sorry ! Go

```

```

back !\");\n\
                                return false ;\n\
                                }\n\
                                if ( seu == \"\" || seu > 1)\n\
                                {\n\
                                alert (\"Problem detected with THRESHOLD. Sorry !
Go back !\");\n\
                                return false ;\n\
                                }\n\
                                document.f_choix.i_select.value = sel.toUpperCase();\n\
                                document.f_choix.i_from.value = fro.toUpperCase();\n\
                                document.f_choix.i_where.value=wherefinal;\n\
                                return true;\n\
                                }\n\
                                }\n\
                                }\n\
                                </SCRIPT>");
                                printf ("<SCRIPT language=\"javascript\">\n\
                                function fn0k2(type)\n\
                                {\n\
                                fnActive(\"choixAtt\");\n\
                                switch (type) {\n\
                                case 1:\n\
                                document.f_choix.i_type.value=\"1\";\n\
                                fnActive(\"choixAtt\");\n\
                                break;\n\
                                case 2:\n\
                                document.f_choix.i_type.value=\"2\";\n\
                                fnActive(\"choixAtt\");\n\
                                break;\n\
                                case 3:\n\
                                document.f_choix.i_type.value=\"3\";\n\
                                fnActive(\"choixAtt\");\n\
                                break;\n\
                                case 4:\n\
                                document.f_choix.i_type.value=\"4\";\n\
                                fnActive(\"choixAtt\");\n\
                                break;\n\
                                }\n\
                                }\n\
                                }\n\
                                </SCRIPT>");

                                printf("<SCRIPT language=\"javascript\">\n\
                                function fnActive(idbloc)\n\
                                {\n\
                                var bloc=document.getElementById(idbloc);\n\
                                bloc.style.visibility='visible';\n\
                                }\n\
                                </SCRIPT>");

                                printf("<SCRIPT language=\"javascript\">\n\
                                function lancer()\n\
                                {\n\
                                document.f_choix.submit();\n\
                                }\n\
                                </SCRIPT>");

                                openCnxAdmin();
                                if (strcmp(i_cond,"2")){
                                printf("<FONT size=\"+2\"><B>Votre requete</B></FONT><BR>");

                                printf("<BR><B>SELECT</B> [%s,%s] %s<BR>\n",i_calibre,i_seuil,
                                i_select) ;
                                printf("<B>FROM</B> %s <B>WHERE:</B> <BR>\n", i_from) ;

```

```

        printf("<FORM name=\"f_choix\" method=\"post\"
action=\"pontliteHp.cgi?chReq4.cgi\" target=\"work\">\n");
        printf("<INPUT type=\"hidden\" name=\"i_calibre\" value=\"%s\">\n",
i_calibre) ;
        printf("<INPUT type=\"hidden\" name=\"i_seuil\" value=\"%s\">\n",
i_seuil) ;
        printf("<INPUT type=\"hidden\" name=\"i_select\" value=\"%s\">\n",
i_select) ;
        printf("<INPUT type=\"hidden\" name=\"i_from\" value=\"%s\">\n",
i_from) ;
        printf("<INPUT type=\"hidden\" name=\"i_where\" value=\"\">\n");

        printf("<INPUT type=\"hidden\" name=\"i_type\" >\n");
        printf("<BR><BR><B>Pour la clause WHERE voulez-vous un format du
type:</B><BR>\n");
        printf("<INPUT type=\"button\" value=\"condition
simple\"onClick=\"fn0k2(4)\"><BR>\n");
        printf("<INPUT type=\"button\" value=\"cond1 AND ... AND
condn\"onClick=\"fn0k2(2)\"><BR>\n");
        printf("<INPUT type=\"button\" value=\"cond1 OR ... OR
condn\"onClick=\"fn0k2(3)\"></BR>\n");
        printf("<INPUT type=\"button\" value=\"la plupart(cond1,...,condn)
\"onClick=\"fn0k2(1)\"><BR>\n");

        printf("<DIV id=\"choixAtt\" class=\"cache\">\n");
        printf("<P> <B> Choix d'un attribut ");
        printf("</B>\n");
        printf("<SELECT name=\"i_att\" onChange=\"lancer()\">\n");
        printf("<OPTION value=\"null\">--\n");

        for ( cptr=0 ; cptr < nbAttrib ; cptr++ )
            printf("<OPTION value=\"%s\">%s\n", listAtt[cptr].att,
listAtt[cptr].att);
        printf("</SELECT>\n");
        printf("</BR>\n");

        printf("</DIV>\n");
        printf("</FORM>\n\n\n");

    }else{
        printf("<FONT size=\"+2\"><B>votre Requête:</B></FONT><BR><BR>");
        printf("<BR><BR><B>SELECT</B> [%s,%s] %s<BR>\n",i_calibre, i_seuil,
i_select) ;
        printf("<B>FROM</B> %s;<BR>\n", i_from) ;
        printf("<FORM name=\"f_choix2\" method=\"POST\"
action=\"pont.cgi?perfreq.cgi_1\" onSubmit=\"return fn0k()\">\n");
        printf("<INPUT type=\"hidden\" name=\"i_calibre\" value=\"%s\">\n",
i_calibre) ;
        printf("<INPUT type=\"hidden\" name=\"i_seuil\" value=\"%s\">\n",
i_seuil) ;
        printf("<INPUT type=\"hidden\" name=\"i_select\" value=\"%s\">\n",
i_select) ;
        printf("<INPUT type=\"hidden\" name=\"i_from\" value=\"%s\">\n",
i_from) ;
        printf("<INPUT type=\"hidden\" name=\"i_cond\"value=\"%s\">\n",
i_cond);

        printf("<INPUT type=\"submit\" value=\"Envoyée\">\n");
        printf("<INPUT type=\"button\" value=\"Modifiée\"

```

```

onClick="history.go(-1)\">>\n") ;
        printf("</FORM>\n\n\n") ;
    }
    closeCnxAdmin () ;
}

```

E)chReq4.c

```

/* *****
 * Name      : chReq4.c
 * Path      : /Cgi
 * Object    : Non Expert Fuzzy Query 4 ; ecriture d'une condition de la clause
where
 * Date      : fev 2006
 * Author    : Samuel Bartel
*****

*****
 * Modifications :
 * -----
 *
***** */

#ifndef __C_DEF_H__
#define __C_DEF_H__
#include "../Config/c_def.h"
#endif

int main (void)
{
    /* Declarations */
    s_champ champs[MAXCHAMPS] ;
    s_champ unChp ;
    int nbChamps ;
    int nbAttrib=0 ;
    int cptr, cptr2, cptr3 ;
    int no, ft, fc, fm, fq ;
    int nbTF, nbMF, nbCF, nbQF ;
    char login[MAXGETNOM] ;
    char base[MAXGETNOM] ;
    char passwd[MAXGETNOM] ;
    char i_from[MAXLONGCOOKIE] ;
    char i_calibre[MAXLONGCOOKIE] ;
    char i_seuil[MAXLONGCOOKIE] ;
    char i_select[MAXLONGCOOKIE] ;
    char att[MAXLONGCOOKIE] ;
    char i_cond[MAXLONGCOOKIE] ;
    char i_type[MAXLONGCOOKIE] ;
    char i_att[MAXLONGCOOKIE] ;
    char i_where[MAXLONGCOOKIE] ;
    char typeAtt[20] ;
    s_att *listAtt ;
    s_tf *listTermes ;
    s_mf *listModifs ;
    s_cf *listCompas ;
    s_qf *listQuants ;

    int deb;

    /* Pointer's init */
    unChp.nom = (char *)malloc (MAXGETNOM) ;
    unChp.val = (char *)malloc (MAXGETVAL) ;
    bzero (login, MAXGETNOM) ;
    bzero (base, MAXGETNOM) ;

```

```

bzero (passwd, MAXGETNOM) ;

openCnxAdmin () ;

/* Analyse of the cookies */
strcpy (login, trouveEltDsCookie ( "pseudo" )) ;
strcpy (base, trouveEltDsCookie ( "base" )) ;
strcpy (passwd, trouveEltDsCookie ( "passwd" )) ;

/* Find the last form datas */
decodeFormulaire (&nbChamps, champs) ;
trouveChamp ("i_from", champs, nbChamps, &unChp) ;
strcpy (i_from, unChp.val) ;
trouveChamp ("i_calibre", champs, nbChamps, &unChp) ;
strcpy (i_calibre, unChp.val) ;
trouveChamp ("i_seuil", champs, nbChamps, &unChp) ;
strcpy (i_seuil, unChp.val) ;
trouveChamp ("i_cond", champs, nbChamps, &unChp) ;
strcpy (i_cond, unChp.val) ;
trouveChamp ("i_type", champs, nbChamps, &unChp) ;
strcpy (i_type, unChp.val) ;
trouveChamp ("i_att", champs, nbChamps, &unChp) ;
strcpy (i_att, unChp.val) ;
trouveChamp ("i_where", champs, nbChamps, &unChp) ;
strcpy (i_where, unChp.val) ;

nbAttrib = listAttributs(base, passwd, i_from, &listAtt) ;

if (trouveChamp ("i_select", champs, nbChamps, &unChp))
    strcpy (i_select, unChp.val) ;
else
{
    /* We are finding all the attributes of this table i_from in order
to transmit them to the next CGI -> i_select */

    no = 0 ;
    strcpy(i_select, "") ;
    for ( cptr=0 ; cptr < nbAttrib ; cptr++ )
    {
        if (trouveChamp (listAtt[cptr].att, champs, nbChamps, &unChp))
        {
            no++ ;
            if (no > 1)
                strcat(i_select, ", ") ;
            strcat(i_select, unChp.val) ;
        }
    }
}

printf("<STYLE>\n");
printf(".cache{visibility :hidden}");
printf(".visible{visibility :visible}");
printf("</STYLE>\n");

/* JavaScript function which verify each answer of the query */
printf ("<SCRIPT language=\"javascript\">\n\
function fnOk(val,nbTF)\n\
{\n\
    var sel = document.f_choix.i_select.value ;\n\
    var fro = document.f_choix.i_from.value ;\n\
    var att = document.f_choix.i_att.value;\n\
    var comp = document.f_choix.i_comp.value;\n\
    var valeur = document.f_choix.i_val.value;\n\
    var where = document.f_choix.i_where.value;\n\
    var cal = document.f_choix.i_calibre.value ;\n\

```



```

        }else{\n\
            where+=modif;\n\
            where+=\"(\";\n\
            where+=att;\n\
            if (comp==\"is\"){\n\
                where+=\"=\";\n\
                where+=\"%%\";\n\
                where+=termFlou;\n\
            }else{\n\
                where+=comp;\n\
                for (i=0;i<valeur.length;i++)\n\
                {\n\
                    where+=valeur[i].
toUpperCase();\n\
                }\n\
            }\n\
            where+=\")\";\n\
        }\n\
    }else{\n\
        where+=att;\n\
        where+=comp;\n\
        for (i=0;i<valeur.length;i++)\n\
        {\n\
            where+=valeur[i].toUpperCase();\n\
        }\n\
    }\n\
    if ((val==2)&&(type==1)){\n\
        where+=\"]\";\n\
    }\n\
    document.f_choix.i_where.value=where;\n\n\
    lancer(val);\n\
    return true;\n\
}\n\
}\n\
</SCRIPT>");
printf("<SCRIPT language=\"javascript\">\n\
function fn0k2(type)\n\
{\n\
    switch (type) {\n\
        case 1:\n\
            document.f_choix.i_type.value=\"1\";\n\
            fnActive(\"choixAtt\");\n\
        break;\n\
        case 2:\n\
            document.f_choix.i_type.value=\"2\";\n\
            fnActive(\"choixAtt\");\n\
        break;\n\
        case 3:\n\
            document.f_choix.i_type.value=\"3\";\n\
            fnActive(\"choixAtt\");\n\
        break;\n\
    }\n\
}\n\
</SCRIPT>");
printf("<SCRIPT language=\"javascript\">\n\
function fn0k3(comp,nbTF)\n\
{\n\
    if(nbTF==0){\n\
        if(comp== \"null\")\n\
        {\n\
            var bloc3=document.getElementById(\"valeurComp\");
\n\
            bloc3.disabled=true;\n }else{\n\
            var bloc3=document.getElementById(\"valeurComp\");
\n\

```

```

        bloc3.disabled=false;\n }\n\n
    }else{\n\n
        if (comp == \"is\") {\n\n
            var bloc1=document.getElementById(\"modifFlou\");\n\n
            var bloc2=document.getElementById(\"termFlou\");\n\n
            var bloc3=document.getElementById(\"valeurComp\");\n\n
            bloc1.disabled=false;\n\n
            bloc2.disabled=false;\n\n
            bloc3.disabled=true;\n\n
        }else{\n\n
            if(comp== \"null\")\n\n
            {\n\n
                var bloc1=document.getElementById(\"modifFlou\");\n\n
\n\n
                var bloc2=document.getElementById(\"termFlou\");\n\n
\n\n
                var bloc3=document.getElementById(\"valeurComp\");\n\n
\n\n
                bloc1.disabled=true;\n\n
                bloc2.disabled=true;\n\n
                bloc3.disabled=true;\n }else{\n\n
\n\n
                var bloc1=document.getElementById(\"modifFlou\");\n\n
\n\n
                var bloc2=document.getElementById(\"termFlou\");\n\n
\n\n
                var bloc3=document.getElementById(\"valeurComp\");\n\n
\n\n
                bloc1.disabled=true;\n\n
                bloc2.disabled=true;\n\n
                bloc3.disabled=false;\n\n
            }\n\n
        }\n\n
    }\n\n
</SCRIPT>");

printf("<SCRIPT language=\"javascript\">\n
function fnActive(idbloc)\n
{\n\n
    var bloc=document.getElementById(idbloc);\n\n
    bloc.style.visibility='visible';\n\n
}\n\n
function lancer(type)\n
{\n\n
    switch (type){\n\n
        case 1 :\n\n
document.f_choix.i_att.value=document.f_choix.i_att2.value;\n\n
        document.f_choix.i_typeVal.value=\"1\";\n\n
        document.f_choix.submit();\n\n
        break;\n\n
        case 2 :\n\n
        document.f_choix.i_typeVal.value=\"2\";\n\n
        document.f_choix.submit();\n\n
        break;\n\n
        case 3 :\n\n
        document.f_choix.i_typeVal.value=\"3\";\n\n
        document.f_choix.submit();\n\n
        break;\n\n
    }\n\n
}\n\n
</SCRIPT>");

openCnxAdmin();
if (strcmp(i_cond,"2")){

```

```

printf("<FONT size=\"+2\"><B>Votre requete</B></FONT><BR>") ;

i_select) ;
printf("<BR><B>SELECT</B> [%s,%s] %s<BR>\n",i_calibre,i_seuil,
printf("<B>FROM</B> %s <BR>\n", i_from) ;
if (strcmp(i_where,"")!=0){
    printf("<B> WHERE <B>\n");
    switch(i_type[0]){
        case '1':
            if (strcmp(i_where,"")==0){
                printf(" la plupart de\n", i_where);
            }else{
                printf("%s , \n", i_where);
            }
            break;
        case '2':
            printf(" %s AND \n",i_where);
            break;
        case '3':
            printf("%s OR \n",i_where);
            break;
    }
}
printf("</BR>\n");
}

printf("<FORM name=\"f_choix\" method=\"post\"
action=\"pontliteHp.cgi?chReq5.cgi\" target=\"work\">\n") ;
printf("<INPUT type=\"hidden\" name=\"i_calibre\" value=\"%s\">\n",
i_calibre) ;
printf("<INPUT type=\"hidden\" name=\"i_seuil\" value=\"%s\">\n",
i_seuil) ;
printf("<INPUT type=\"hidden\" name=\"i_select\" value=\"%s\">\n",
i_select) ;
printf("<INPUT type=\"hidden\" name=\"i_from\" value=\"%s\">\n",
i_from) ;
printf("<INPUT type=\"hidden\" name=\"i_type\" value=\"%s\">\n",
i_type) ;
printf("<INPUT type=\"hidden\" name=\"i_att\" value=\"%s\">\n",
i_att) ;
printf("<INPUT type=\"hidden\" name=\"i_where\" value=\"%s\">\n",
i_where) ;
printf("<INPUT type=\"hidden\" name=\"i_typeVal\">\n") ;

/*rappel du choix du type de la clause where choisi*/
printf("<BR><BR><i>Pour la clause WHERE vous avez choisi le format :
\n");
switch(i_type[0]){
    case '1':
        printf("la plupart(cond1,..,condn)</i>\n");
        break;
    case '2':
        printf("cond1 AND ... AND condn</i>\n");
        break;
    case '3':
        printf("cond1 OR ... OR condn</i>\n");
        break;
    case '4':
        printf("condition simple</i>\n");
        break;
}

```

```

printf("<P> <B> Choix de l'attribut ") ;
printf("</B>\n");
printf("<SELECT value=\"%s\" name=\"i_att2\" onChange=\"lancer
(1)\">>\n",i_att);
    printf("<OPTION value=\"null\">--\n");

    for ( cptr=0 ; cptr < nbAttrib ; cptr++ ){
        if (strcmp(i_att,listAtt[cptr].att)==0){
            printf("<OPTION SELECTED value=\"%s\">%s\n",
listAtt[cptr].att, listAtt[cptr].att);
        }else{
            printf("<OPTION value=\"%s\">%s\n", listAtt
[cptr].att, listAtt[cptr].att);
        }
    }
    printf("</SELECT>\n") ;
    printf("</BR>\n");

    nbTF = listTermesFlous4NEFQ(login, base, i_from, i_att,
&listTermes) ;

    printf("<P> <B> Comparateur ") ;

    printf("</B>\n");
    printf("<SELECT name=\"i_comp\" Onchange=\"fn0k3(this.value,%
i)\">>\n",nbTF) ;
        printf("<option value=\"null\">----</option>\n");
        printf("    <OPTION value=\"=\">=</OPTION>\n") ;
        if (strcmp(typeAtt,"CHAR")!=0){
            printf("        <OPTION value=\">\"></OPTION>\n") ;
            printf("        <OPTION value=\"<\"><</OPTION>\n") ;
            printf("        <OPTION value=\">=\">=</OPTION>\n") ;
            printf("        <OPTION value=\"<=\"><=</OPTION>\n") ;
        }
        if (nbTF>0){
            printf("    <OPTION
value=\"is\">is</OPTION>\n") ;
        }
        printf("</SELECT></P>\n") ;
        printf("</BR>\n");

        if(nbTF>0){
            nbMF = listModifFlous4NEFQ(login, base, &listModifs) ;
            if ( nbMF > 0 )
            {
                /*liste des modificateurs flous*/
                printf("<P> <B> modificateur ") ;
                printf("</B>\n");
                printf("<SELECT id=\"modifFlou\"
name=\"i_modificateur\" DISABLED> \n") ;
                printf("<option value=\"null\">----</option>\n");
                for ( cptr=0 ; cptr < nbMF ; cptr++ )
                    printf("<option value=\"%s\">%s</option>\n",
listModifs[cptr].modif, listModifs[cptr].modif) ;
                printf("</SELECT></P>\n") ;
                printf("</BR>\n");
            }

            /*liste des termes flous*/

```

```

        printf("<P> <B> Termflou ") ;
        printf("</B>\n");
        printf("<SELECT id=\"termFlou\" name=\"i_termFlou\"
DISABLED>\n") ;
        printf("<option value=\"null\">----</option>\n");
        for ( cptr2=0 ; cptr2 < nbTF ; cptr2++ )
            printf("<option value=\"%s\">%s</option>\n",
listTermes[cptr2].terme, listTermes[cptr2].terme) ;
        printf("</SELECT></P>\n") ;

        printf("</BR>\n");
    }
    /*champs de saisie d'une valeur*/
    printf("<P> <B> Valeur ") ;
    printf("</B>\n");
    printf("<INPUT id=\"valeurComp\" TYPE=\"text\" NAME=\"i_val\"
DISABLED>\n");
    printf("</BR></BR>\n");

    printf("<INPUT type=\"Button\" value=\"Envoyer\"
onclick=\"fnOk(2,%i)\"> ", nbTF) ;
    if (i_type[0]!='4'){
        printf("<INPUT type=\"Button\" value=\"Compléter\"
onclick=\"fnOk(3,%i)\"> ", nbTF) ;
    }

    printf("</FORM>\n");

}
}else{
    printf("<FONT size=\"+2\"><B>votre Requête:</B></FONT><BR><BR>") ;
    printf("<BR><BR><B>SELECT</B> [%s,%s] %s<BR>\n",i_calibre, i_seuil,
i_select) ;
    printf("<B>FROM</B> %s;<BR>\n", i_from) ;
    printf("<FORM name=\"f_choix2\" method=\"POST\"
action=\"pont.cgi?perfreq.cgi_1\" onSubmit=\"return fnOk()\">\n") ;
    printf("<INPUT type=\"hidden\" name=\"i_calibre\" value=\"%s\">\n",
i_calibre) ;
    printf("<INPUT type=\"hidden\" name=\"i_seuil\" value=\"%s\">\n",
i_seuil) ;
    printf("<INPUT type=\"hidden\" name=\"i_select\" value=\"%s\">\n",
i_select) ;
    printf("<INPUT type=\"hidden\" name=\"i_from\" value=\"%s\">\n",
i_from) ;
    printf("<INPUT type=\"hidden\" name=\"i_cond\" value=\"%s\">\n",
i_cond);

    printf("<INPUT type=\"submit\" value=\"Envoyée\">\n") ;
    printf("<INPUT type=\"button\" value=\"Modifiée\"
onClick=\"history.go(-1)\">\n") ;
    printf("</FORM>\n\n\n") ;
}
closeCnxAdmin () ;
}

```

F)chReq5.c

```

/* *****
* Name   : chReq5.c
* Path   : /Cgi
* Object : Non Expert Fuzzy Query 5 ; CGI intermédiaire pour rafraichissement

```

```

*          ou affichage de la requete avant validation
* Date      : fev 2006
* Author    : Samuel Bartem
*****

*****

* Modifications :
* -----
*
***** */

```

```

#ifndef __C_DEF_H__
#define __C_DEF_H__
#include "../Config/c_def.h"
#endif

```

```

int main (void)
{
    /* Declarations */
    s_champ champs[MAXCHAMPS] ;
    s_champ unChp ;
    int nbChamps ;
    int nbAttrib=0 ;
    int cptr, cptr2, cptr3 ;
    int no, ft, fc, fm, fq ;
    int nbTF, nbMF, nbCF, nbQF ;
    char login[MAXGETNOM] ;
    char base[MAXGETNOM] ;
    char passwd[MAXGETNOM] ;
    char i_from[MAXLONGCOOKIE] ;
    char i_calibre[MAXLONGCOOKIE] ;
    char i_seuil[MAXLONGCOOKIE] ;
    char i_select[MAXLONGCOOKIE] ;
    char att[MAXLONGCOOKIE] ;
    char i_cond[MAXLONGCOOKIE] ;
    char i_type[MAXLONGCOOKIE] ;
    char i_att[MAXLONGCOOKIE] ;
    char i_typeVal[MAXLONGCOOKIE] ;
    char i_where[MAXLONGCOOKIE] ;
    s_att *listAtt ;
    s_tf *listTermes ;
    s_mf *listModifs ;
    s_cf *listCompas ;
    s_qf *listQuants ;

    /* Pointer's init */
    unChp.nom = (char *)malloc (MAXGETNOM) ;
    unChp.val = (char *)malloc (MAXGETVAL) ;
    bzero (login, MAXGETNOM) ;
    bzero (base, MAXGETNOM) ;
    bzero (passwd, MAXGETNOM) ;

    openCnxAdmin () ;

    /* Analyse of the cookies */
    strcpy (login, trouveEltDsCookie ( "pseudo" )) ;
    strcpy (base, trouveEltDsCookie ( "base" )) ;
    strcpy (passwd, trouveEltDsCookie ( "passwd" )) ;

    /* Find the last form datas */
    decodeFormulaire (&nbChamps, champs) ;
    trouveChamp ("i_from", champs, nbChamps, &unChp) ;
    strcpy (i_from, unChp.val) ;
    trouveChamp ("i_calibre", champs, nbChamps, &unChp) ;
    strcpy (i_calibre, unChp.val) ;

```

```

trouveChamp ("i_seuil", champs, nbChamps, &unChp) ;
strcpy (i_seuil, unChp.val) ;
trouveChamp ("i_cond",champs,nbChamps,&unChp);
strcpy(i_cond, unChp.val);
trouveChamp ("i_type",champs,nbChamps,&unChp);
strcpy(i_type, unChp.val);
trouveChamp ("i_att",champs,nbChamps,&unChp);
strcpy(i_att, unChp.val);
trouveChamp ("i_typeVal",champs,nbChamps,&unChp);
strcpy(i_typeVal, unChp.val);
trouveChamp ("i_where",champs,nbChamps,&unChp);
strcpy(i_where, unChp.val);

    if (trouveChamp ("i_select", champs, nbChamps, &unChp))
        strcpy (i_select, unChp.val) ;
else
{

    no = 0 ;
    strcpy(i_select, "") ;
    for ( cptr=0 ; cptr < nbAttrib ; cptr++ )
    {
        if (trouveChamp (listAtt[cptr].att, champs, nbChamps, &unChp))
        {
            no++ ;
            if (no > 1)
                strcat(i_select, ", ") ;
            strcat(i_select, unChp.val) ;
        }
    }
}

printf("<SCRIPT language=\"javascript\">\n\
function lancer()\n\
{\n\
    document.f_choix.submit();\n\
}\n\
</SCRIPT>\n\n");

printf("<BODY ONLOAD=\"'lancer()'\>\n");

switch (i_typeVal[0]){
case '1': /*rafraichissement de la page*/
    printf("<BODY ONLOAD=\"'lancer()'\>\n");
    printf("<FORM name=\"f_choix\" method=\"post\"
action=\"pontliteHp.cgi?chReq4.cgi\" target=\"work\">\n");
    printf("<INPUT type=\"hidden\" name=\"i_calibre\" value=\"%s\">\n",
i_calibre) ;
    printf("<INPUT type=\"hidden\" name=\"i_seuil\" value=\"%s\">\n",
i_seuil) ;
    printf("<INPUT type=\"hidden\" name=\"i_select\" value=\"%s\">\n",
i_select) ;
    printf("<INPUT type=\"hidden\" name=\"i_from\" value=\"%s\">\n",
i_from) ;
    printf("<INPUT type=\"hidden\" name=\"i_type\" value=\"%s\">\n",
i_type) ;
    printf("<INPUT type=\"hidden\" name=\"i_att\" value=\"%s\">\n",
i_att) ;
    printf("<INPUT type=\"hidden\" name=\"i_where\" value=\"%s\">\n",
i_where) ;
    printf("</FORM>\n");
    printf("</BODY>\n");
break;
case '2': /*affichage avant validation*/

```

```

        printf("<FONT size=\"+2\"><B>votre Requete:</B></FONT><BR><BR>");
        printf("<BR><BR><B>SELECT</B> [%s,%s] %s<BR>\n",i_calibre, i_seuil,
i_select) ;
        printf("<B>FROM</B> %s<BR>\n", i_from) ;
        printf("<B>WHERE</B> %s\n", i_where);
        printf("<FORM name=\"f_choix2\" method=\"POST\"
action=\"pont.cgi?perfreq.cgi_1\" onSubmit=\"return fnOk()\">\n");
        printf("<INPUT type=\"hidden\" name=\"i_calibre\" value=\"%s\">\n",
i_calibre) ;
        printf("<INPUT type=\"hidden\" name=\"i_seuil\" value=\"%s\">\n",
i_seuil) ;
        printf("<INPUT type=\"hidden\" name=\"i_select\" value=\"%s\">\n",
i_select) ;
        printf("<INPUT type=\"hidden\" name=\"i_from\" value=\"%s\">\n",
i_from) ;
        printf("<INPUT type=\"hidden\" name=\"i_cond\" value=\"%s\">\n",
i_cond);
        printf("<INPUT type=\"hidden\" name=\"i_where\" value=\"%s\">\n",
i_where);
        printf("<INPUT type=\"submit\" value=\"Envoyee\">\n");
        printf("<INPUT type=\"button\" value=\"Modifiee\"
onClick=\"history.go(-1)\">\n");
        printf("</FORM>\n\n\n");
        break;
        case '3': /*choix de completer la requête*/
        printf("<BODY ONLOAD=\"lancer()\">\n");
        printf("<FORM name=\"f_choix\" method=\"post\"
action=\"pontliteHp.cgi?chReq4.cgi\" target=\"work\">\n");
        printf("<INPUT type=\"hidden\" name=\"i_calibre\" value=\"%s\">\n",
i_calibre) ;
        printf("<INPUT type=\"hidden\" name=\"i_seuil\" value=\"%s\">\n",
i_seuil) ;
        printf("<INPUT type=\"hidden\" name=\"i_select\" value=\"%s\">\n",
i_select) ;
        printf("<INPUT type=\"hidden\" name=\"i_from\" value=\"%s\">\n",
i_from) ;
        printf("<INPUT type=\"hidden\" name=\"i_type\" value=\"%s\">\n",
i_type) ;
        printf("<INPUT type=\"hidden\" name=\"i_where\" value=\"%s\">\n",
i_where) ;
        printf("</FORM>\n");
        printf("</BODY>\n");
        break;
        break;
    }

    closeCnxAdmin () ;

}

```

H)perfreq.c

```

/* *****
* Nom      : perfreq.c
* Chemin   : /Cgi
* Objet    : executer la requete floue puis afficher le resultat
* Date     : 15 nov 99
* Auteur   : Didier Daul
*****

```

```

*****
* Historique des modifications :
* -----
* Fabien Vannier (26/10/2000) : permettre sauvegarde d'une requete
  meme qd celle-ci ne renvoie aucun resultat
  Fabien Vannier (2001) : travail de redirection des erreurs afin
  d'eviter la presence de 'core dumped' a chaque requete posant probleme...
  Samuel Bartel (2006) modification des traitements pour création des
requetes
  experts et non experts pour les adaptés aux nouvelles interfaces
  Samuel Bartel (2006) correction de la chargement de requêtes sauvegardées
***** */

#ifndef __C_DEF_H__
#define __C_DEF_H__
#include "../Config/c_def.h"
#include "/usr/include/ctype.h"

/* ##### PROTOTYPES
## */

void affReqSQLfErronee();
void affCompilationPb();
void affExecutionPb();
void affAnalysePb();

/* ##### FONCTIONS
## */

/* Fonction d'affichage pour le cas ou la requete SQLf comporte des erreurs */
void affReqSQLfErronee(void)
{
    printf("<BR><BR>Your request is not valid.<BR><BR>\n") ;
    printf("The field <EM>WHERE</EM> may be wrong written: <BR>\n") ;
    printf("<UL>\n") ;
    printf("<LI>a chosen attribute doesn't exist\n") ;
    printf("<LI>a chosen fuzzy term doesn't exist\n") ;
    printf("<LI>a chosen fuzzy term can't be used the associated attribute\n")
;
    printf("<LI>the sentence is not a valid SQLf one\n") ;
    printf("<LI>other cases... :o(\n") ;
    printf("</UL>\n") ;
    printf("<BR>If the problem appears several times, contact the webmaster ")
;
    printf("(<EM>problem during the traduction of the fuzzy query</EM>).
<BR><BR><BR>\n") ;
}

/* Fonction d'affichage pour le cas ou la chaine de developpement ait plante */
void affCompilationPb(void)
{
    printf("<BR><BR>The system has detected a problem during the
compilation.<BR>\n") ;
    printf("<BR>If the problem appears several times, contact the webmaster ")
;
    printf("(<EM>problem during the compilation</EM>).<BR><BR><BR>\n") ;
}

/* Fonction d'affichage pour le cas ou l'execution n'ait pas marche */
void affExecutionPb(void)
{

```

```

        printf("<BR><BR>The system has detected a problem during the execution of
the ") ;
        printf("file generated by the request.<BR>\n") ;
        printf("<BR>If the problem appears several times, contact the webmaster ")
;
        printf("(<EM>problem during the execution</EM>).<BR><BR><BR>\n") ;
}

/* Fonction d'affichage pour le cas ou l'analyse du fichier resultat ait plante
*/
void affAnalysePb(void)
{
    printf("<BR><BR>The system has detected a problem during the analyse of
the ") ;
    printf("result file.<BR>\n") ;
    printf("<BR>If the problem appears several times, contact the webmaster ")
;
    printf("(<EM>problem with the analyse of the result file</EM>).
<BR><BR>\n") ;
}

/* argv[1] est censé contenir le code de la page d'accès :
1 -> chReq3.cgi (non expert) ou requete.cgi (expert)
3 -> chReq2.cgi (non expert sans WHERE non fonctionnel)
2 -> listreq.cgi (bibliotheque de requete) */
int main (int argc, char **argv)
{
    FILE *fd ;          /* Descripteur de fichier pour la requete floue */

    t_sat      seuil;

    int         fdsm ;          /* Descripteur de fichier pour la requete
floue */
    int         cptr ;          /* compteur de boucle a usage libre */
    int         cptr2 ;         /* compteur de boucle a usage libre */
    int         calibre ;       /* nombre de n-uplets resultats souhaites par
l'utilisateur */
    int         ligne ;         /* nombre de lignes du fichier contenant le
resultat de l'interrogation */
    int         nbCol ;         /* nombre de colonnes du fichier contenant
le resultat de l'interrogation */
    int         cptrCol ;       /* compteur de boucle pour travailler sur chaque
colonne separemment */
    int         cptrCar ;       /* compteur pour travailler caractere par
caractere sur le n-uplet courant */
    int         cptrTmpCh ;     /* compteur d'avancement dans tmpCh */
    char        tmpCh[MAXGETVAL] ; /* chaine temporaire qui contient une
colonne du n-uplet courant */
    char        nomfd[100] ;
    int         nbChamps ;       /* nombre de champs dans formulaire */
    s_champ     champs[MAXCHAMPS] ; /* tableau des champs decodes */
    s_champ     unChp ;          /* structures champ temporaires
*/

    char        traduc[MAXGETVAL] ;
    char        nomLog[MAXGETVAL] ;
    char        mainCde[MAXGETVAL] ; /* Nom du traducteur requete floue -->
fichier .pc */
    char        tmpTest[2] ;      /* buffer temporaire pour tester l'etat
d'ouverture des fichiers */
    char        **tabRes ;        /* tableau de tous les n-uplets resultat */
    char        champSelect[MAXGETVAL] ; /* liste des colonnes voulues dans la
requete */
    t_sat       annexe[MAXGETVAL] ; /* tableau des satisfaction des n-uplets
resultats */

```

```

char tmpAnnexe[4] ;
char tmpTabRes[MAXGETVAL] ;
s_champ *cookies ; /* tableau des cookies decodes */
s_champ cook ; /* structures ccookie temporaires */
int nbCookie ; /* nombre de cookies pour l'utilisateur */
char login[MAXGETNOM] ;
char tmp[MAXGETNOM] ;
int pos ;
int i;

/*****
/* INITIALISATION */
*****/

/* Initialisation des pointeurs */
unChp.nom = (char *)malloc (MAXGETNOM) ;
unChp.val = (char *)malloc (MAXGETVAL) ;
cook.nom = (char *)malloc (MAXGETNOM) ;
cook.val = (char *)malloc (MAXGETVAL) ;
bzero (login, MAXGETNOM) ;
for (cptr=0; cptr<MAXGETVAL; cptr++)
    bzero (annexe[cptr], 5) ;
bzero (tmp, MAXGETNOM) ;
bzero (champSelect, MAXGETVAL) ;

/* Placement de variables d'environnement utiles (CGI) */
placeEnv () ;

/* Decodage de la liste des cookies */
decodeCookie (&nbCookie, &cookies) ;
if ( ! trouveCookie ( "pseudo", cookies, nbCookie, &cook ) )
{
    erreur ("Impossible de determiner l'identite", 505) ;
}
else
{
    strcpy (login, cook.val);
}
/*tracerbis("Debut de requete pour l'utilisateur",login);*/

/* construction du nom du fichier qui servira de LOG */
sprintf ( nomLog, "%s/%s.%s", PATH_ERROR_LOG, LOG_TRADUCTEUR, login ) ;

/* Nettoyage des repertoires temporaires de travail */
if ( faireMenageAvantTraduc ( login, nomLog ) != 1 ){
    erreur ("Une erreur s'est produite pendant le nettoyage
preliminaire", 608) ;
}

/* Decodage du formulaire et ecriture dans le fichier de requete floue */
decodeFormulaire (&nbChamps, champs) ;

/*****
/* CREATION DU FICHIER CONTENANT LA REQUETE */
*****/

/* Creation du fichier qui va contenir la requete saisie par l'utilisateur
*/
sprintf (nomfd, "%s/%s.%s", PATH_FUZZY_PARAM, REQUETE_FLOUE_EXT, login) ;
if ( atoi(argv[1]) == 2 )
{
    /* On relance une requete pre-enregistree */
    fdsm = open ( nomfd , O_RDWR | O_TRUNC | O_CREAT , S_DEFFILEMODE ) ;

```

```

        if ( ! fdsm )
            erreur ("Impossible de creer le fichier de requete floue
temporaire", 661) ;

        trouveChamp ("requete", champs, nbChamps, &unChp) ;
        write (fdsm, unChp.val, strlen (unChp.val)) ;
        /* Requete transcrite dans le fichier --> on peut le fermer */
        close (fdsm) ;

        /* Trouvons quels sont les champs demandes */
        cptr = 0 ; pos = 0 ;
        while ( unChp.val[cptr] != ']' ) cptr++ ;
        cptr++ ;
        while ( unChp.val[cptr] == ' ' ) cptr++ ;
        pos = cptr ;
        while ( strncmp (unChp.val+pos, " FROM", 5)) pos++ ;
        strncpy (champSelect, unChp.val+cptr, pos-cptr) ;

        /* Determinons le calibre */
        trouveChamp ("calibre", champs, nbChamps, &unChp) ;
        calibre = atoi (unChp.val) ;

    }
else if ( atoi(argv[1]) == 3 )
{
    /* On a construit une nouvelle requete expert*/
    if ( (fd = fopen (nomfd, "w+")) == NULL )
        erreur ("Impossible de creer le fichier de requete floue
temporaire", 61) ;

    trouveChamp ("i_calibre", champs, nbChamps, &unChp) ;
    fprintf (fd, "SELECT [%s,", unChp.val) ;
    calibre = atoi (unChp.val) ;
    trouveChamp ("i_seuil", champs, nbChamps, &unChp) ;
    fprintf (fd, "[%s] ", unChp.val) ;

    trouveChamp ("i_requete", champs, nbChamps, &unChp) ;
    i=0;
    while (unChp.val[i]){
        unChp.val[i]=toupper(unChp.val[i]);
        i++;
    }

    if (unChp.val[0]=' '){
        fprintf (fd, "%s ;", unChp.val+7) ;
    }else{
        fprintf (fd, "%s ;", unChp.val+6) ;
    }
    }

    /* Requete transcrite dans le fichier --> on peut le fermer */
    fclose (fd) ;

}

}

/* On a construit une nouvelle requete non expert*/
if ( (fd = fopen (nomfd, "w+")) == NULL )
    erreur ("Impossible de creer le fichier de requete floue
temporaire", 61) ;

trouveChamp ("i_calibre", champs, nbChamps, &unChp) ;

```

```

fprintf (fd, "SELECT [%s,", unChp.val) ;
calibre = atoi (unChp.val) ;
trouveChamp ("i_seuil", champs, nbChamps, &unChp) ;

strcpy(seuil,unChp.val);

fprintf (fd, "%s] ", unChp.val) ;

trouveChamp ("i_requete", champs, nbChamps, &unChp) ;
i=0;
while (unChp.val[i]){
    unChp.val[i]=toupper(unChp.val[i]);
    i++;
}

trouveChamp ("i_select", champs, nbChamps, &unChp) ;
fprintf (fd, "%s ", unChp.val) ;
strcpy (champSelect, unChp.val) ;
trouveChamp ("i_from", champs, nbChamps, &unChp) ;
fprintf (fd, "FROM %s ", unChp.val) ;
trouveChamp("i_cond", champs, nbChamps, &unChp);
if (strcmp(unChp.val,"2")){

if ( atoi(argv[1]) == 1)
{
    trouveChamp ("i_where", champs, nbChamps, &unChp) ;
    fprintf (fd, "WHERE %s ;", unChp.val) ;
}

}else{
    fprintf(fd,";");
}

/* Requete transcrite dans le fichier --> on peut le fermer */
fclose (fd) ;

}

/*****
/* APPEL DU MOTEUR SQLf -> creation d'un fichier genere .pc */
*****/

if ( traduireRequeteFloue (login, nomLog ) != 0 )
{
    affReqSQLfErronee () ;
    return 2 ;
}
/* Dans la fct 'traduireRequeteFloue', un fichier de log est cree */

/*****
/* COMPILATION DU FICHER GENERE -> executable .bin */
*****/

tracer ("\n+++++", nomLog) ;
tracer ("+ COMPILATION .pc => .bin +", nomLog) ;
tracer ("+++++\n", nomLog) ;
fflush (stdout) ;

if ( compilation_ProC ( nomLog, login ) != 0 )
{
    /* Pb durant la chaine de compilation */
    affCompilationPb () ;
    return 2 ;
}

```

```

/*****
/* EXECUTION DU FICHIER GENERE -> creation d'un fichier 'resultats' */
*****/

tracer ("\n+++++", nomLog) ;
tracer ("+ EXECUTION .bin => fichier resultat          +", nomLog) ;
tracer ("+++++\n", nomLog) ;
fflush (stdout) ;

bzero (mainCde, MAXGETVAL) ;
sprintf (mainCde, "%s/%s.bin", PATH_FUZZY_RESULT, login) ;
if (system (mainCde) != 0)
{
    /* Pb durant l'execution */
    affExecutionPb () ;
    return 2 ;
}

/*****
/* ANALYSE DU RESULTAT ET AFFICHAGE FORMATE HTML */
*****/

tracer ("\n+++++", nomLog) ;
fflush (stdout) ;
tracer ("+ ANALYSE ET FORMATAGE HTML          +", nomLog) ;
fflush (stdout) ;
tracer ("+++++\n", nomLog) ;
fflush (stdout) ;
/* Il faut maintenant analyser et decortiquer le fichier resultat ... */
ligne = analyseFichierResultat ( login, &tabRes, nomLog ) ;
if (ligne < 1)
{
    /* pb durant l'analyse du fichier resultat */
    return 2 ;
}

/* a ce niveau, ligne contient le nombre de n-uplets reponse */
if ( (ligne == 1) && (atoi(argv[1]) != 1) )
{
    /* Cas ou :
    - il s'agit d'une requete deja memorisee
    - il n'y a pas de resultat correspondant a la requete soumise */
    printf ("<b><br><br><center>Il n'y a pas de resultat correspondant \
a la requete soumise</b><br><br><br>\n") ;
}
else if ( (ligne == 1) && (atoi(argv[1]) == 1) )
{
    /* Cas ou :
    - il s'agit d'une requete non memorisee
    - il n'y a pas de resultat correspondant a la requete soumise
    => on propose la sauvegarde de la requete
    rq : la variable calibre a bien ete calculee ds ce cas */
    printf ("<center><b><br><br>Il n'y a pas de resultat correspondant \
a la requete soumise</b><br><br><br>\n");
    printf ("<a href=\"pont.cgi?savereq.cgi_%d\"
target=_window><b>Sauvegarder \
cette requ&ecirc;te dans mon panier de
requ&ecirc;tes</b></a></center><br>\n", calibre) ;
}
else
{
    /* Il y a au moins un n-uplet resultat */

    /* Il faut savoir combien de colonnes il y a : une de plus que de
separateurs de colonnes ! */

```

```

        for (cptr=0, nbCol=0; cptr<strlen(tabRes[1])-strlen(CH_SEP_COL);
cptr++)
            if ( !strcmp (CH_SEP_COL, tabRes[1]+cptr, strlen
(CH_SEP_COL))) nbCol++ ;
            if (nbCol) nbCol++ ;

        /* Il faut maintenant trier le fichier dans le sens de la
satisfaction croissante */
        /* Pour cela, on construit un tableau annexe contenant la
satisfaction de chaque
            ligne du tableau principal */
        for ( cptr=1; cptr < ligne; cptr++ )
            strncpy (annexe[cptr], tabRes[cptr]+(strlen (tabRes[cptr]) -
4), 4) ;

        /* On fait maintenant un tri a bulles sur les deux tableaux
simultanement */
        cptr = 1 ;
        while (cptr < ligne)
        {
            cptr2 = 1 ;
            while (cptr2 < ligne)
            {
                if ( strcmp (annexe[cptr2], annexe[cptr2+1]) < 0 )
                {
                    strcpy (tmpAnnexe, annexe[cptr2]) ;
                    strcpy (tmpTabRes, tabRes[cptr2]) ;

                    strcpy (annexe[cptr2], annexe[cptr2+1]) ;
                    strcpy (tabRes[cptr2], tabRes[cptr2+1]) ;

                    strcpy (annexe[cptr2+1], tmpAnnexe) ;
                    strcpy (tabRes[cptr2+1], tmpTabRes) ;
                }
                cptr2++ ;
            }
            cptr++ ;
        }

        cptr=1;
        while ((cptr < ligne)&&(strcmp(annexe[cptr],seuil)>=0)){
            cptr++;
        }
        ligne=cptr;

        /* Il faut a present calibrer le tableau resultat, ie ne retenir que
le nombre de n-uplets desire par l'utilisateur */
        if ( calibre && calibre < (ligne-1) )
            ligne = calibre + 1 ;

        tracer ("Calibrage termine !", nomLog) ; fflush (stdout) ;

        /* il faut maintenant traiter le tableau pour l'afficher en page
html */

        /* on peut maintenant construire une table */
        printf ("<table width=100%% border=1 cellpadding=4 cellspacing=-
4>\n") ;

        /* Etape 1 : la ligne avec le nom des colonnes */
        printf ("<tr align=center valign=center bgcolor=\"#afd5fa\">\n") ;
        for (cptrCol=0, cptrCar=0; cptrCol<nbCol-1; cptrCol++)
        {
            bzero (tmpCh, MAXGETVAL) ;
            cptrTmpCh = 0 ;
            while ( cptrCar < strlen (champSelect) && champSelect[cptrCar]

```

```

!= ' , ' )
    {
        tmpCh[cptrTmpCh] = champSelect[cptrCar] ;
        cptrCar++ ;
        cptrTmpCh++ ;
    }
    cptrCar++ ;
    printf ("<td width=%d%%>%s</td>\n", 100/nbCol, tmpCh) ;
}
bzero (tmpCh, MAXGETVAL) ;
printf ("<td width=%d%%>Satisf.</td>\n", 100/nbCol) ;
printf ("</tr>\n") ;

/* Etape 2 : les donnees dans les colonnes */
for (cptr=1; cptr<ligne; cptr++)
{
    printf ("<tr align=center valign=center>\n") ;
    for (cptrCol=0, cptrCar=0; cptrCol<nbCol-1; cptrCol++)
    {
        bzero (tmpCh, MAXGETVAL) ;
        cptrTmpCh = 0 ;
        while ( strncmp (CH_SEP_COL, tabRes[cptr]+cptrCar,
strlen (CH_SEP_COL)) )
        {
            tmpCh[cptrTmpCh] = tabRes[cptr][cptrCar] ;
            cptrCar++ ;
            cptrTmpCh++ ;
        }
        cptrCar += strlen (CH_SEP_COL) ;
        printf ("<td width=%d%%>%s</td>\n", 100/nbCol, tmpCh) ;
    }
    bzero (tmpCh, MAXGETVAL) ;
    printf ("<td width=%d%%>%s</td>\n", 100/nbCol, tabRes[cptr]
+cptrCar) ;
    printf ("</tr>\n") ;
}
printf ("</table>\n") ;

if ( atoi(argv[1]) ==1 )
    printf ("<center><br><br><a href=\"pont.cgi?savereq.cgi_%d\"
target=_window><b>Sauvegarder cette requ&ecirc;te dans mon panier de
requ&ecirc;tes</b></a><br>\n", calibre) ;
else
    printf ("<center><br><br><a href=\"javascript:window.back()
\"><b>Retour</b></a><br>\n") ;
}

for (cptr=0; cptr<ligne; cptr++)
    free ( tabRes[cptr] ) ;

tracer ("\n+++++ FIN +++", nomLog) ;
fflush (stdout) ;

return 0 ;
}

```

I)derivation.c

```

/*****
*
**

```

MOTEUR SQLf POUR iSQLf -> DERIVATION

**

*

Mathias SALLE, Imad EL AOUNI, Didier DAUL, Fabien VANNIER
ENSSAT -- L.L.I.

File: 'derivation.c'
Last modifications: 20/04/2001 by MR

* */

/* ##### HISTORIQUE DES
MODIFICATIONS ## */

/*-----
Module de derivation pour SQLf

Nom: SALLE Mathias Creer le: 28/11/97

Historique
Date: le 17/12/98 Nom:EL AOUNI
Modifications: Parametrage: Enlever tous les appels systemes avec des chemins
absolus et les remplacer par des variables d'environnements

Date: le 01/03/99 Nom:EL AOUNI
Modifications: Bogue: les derivation de <, >, <=, >= n'ont pas ete prevue
rajout de fonction de derivation pour ces comprateurs

----- */

/*-----
Fonctions de derivation pour SQLf

Nom: SALLE Mathias Creer le: 18/12/97

Historique
Date: le 17/12/98 Nom:EL AOUNI

Modifications: Portage de l'IUT a L'ENSSAT :
rajout de variables d'environnement pour les
appels d'executables et de fichiers.

but: Calcul de l'expression derivee et de la fonction de calcul
du degre associee

----- */

/*-----
Historique

Date: le 01/02/99 Nom:EL AOUNI
Modifications: Bogue: les derivation EGAL OR AND...(ne considerait que le cas flou)
ne marchait pas pour des cas simple (exemple:age=10 AND salaire=200)

si condition boolean <cond_boole> alors
 derivee(<cond_boole>)=<cond_boole>
 fction_calcul_degre(<cond_boole>)=null;

Date: le 01/03/99 Nom:EL AOUNI
Modifications: Bogue: les derivation de <,>,<=,>= n'ont pas ete prevue
rajout de fonction de derivation pour ces compateurs

Date: le 19/03/1998 Nom:EL AOUNI
==> prise en compte de la jointure ce qui a necessite un grand changement
la clause where peut etre du type
==> <att>[comparateur]<att> : jointure EMP.NOM=COMMANDe.NOM
==> <att>[compreteur] "%termeflou" AGE="%jeune"
==> <att>[compreteur] "chaine" NOM="SALLE"
==> <att>[comparateur] flottant AGE=23
toutes ses derivations ont ete rajoutees

Date: le 01/04/1998 Nom:EL AOUNI
==>il a ete decide que les termes flou n'auront plus la forme "%terme_flou"
mais %terme_flou donc
la clause where peut contenir
==> <att> = %termeflou exemple AGE=%jeune
changement effectue dans DerivationEgal (...)

Date: le 26/03/2001 Nom:Maria RUEDA
Localitation: derivation
Modifications: Derniere branche, le maillon garde un quantificateur

Date: le 20/04/2001 Nom:Maria RUEDA
Localitation:
Modifications: Reorganisation du fichier

Date: le 31/01/2002 Nom:Johann L'HOURL
Localitation: derivationEgal
Modifications: Chaine resultat mal formee

Date: le 01/03/2002 Nom:Johann L'HOURL
Localitation: derivationEgal
Modifications: modification du cas ATT=ATT pour supporter les jointures

Date: le 08/03/2002 Nom:Johann L'HOURL
Localitation: derivationEgal
Modifications: Chaine resultat mal formee

Date: le 01/03/2002 Nom:Johann L'HOURL

Localitation: DerivationIN

Modifications: creation de la fonction pour supporter le 'IN (...)'

Date: 03/2006

Nom: Samuel Bartel

localisation: derivationEgal

Modifications:correction problème du seuil=0

localisation: derivationInf

Modifications:correction problème du seuil=0

localisation: derivationInfEgal

Modifications:correction problème du seuil=0

localisation: derivationSup

Modifications:correction problème du seuil=0

localisation: derivationSupEgal

Modifications:correction problème du seuil=0

localisation: derivationDiffer

Modifications:correction problème du seuil=0

localisation: derivationOr

Modifications:correction problème du seuil=0

localisation: derivationAnd

Modifications:correction problème du seuil=0

-----*/

/* #####
*/

/* Bibliotheque de configuration commune (moteur SQLf) */

#include "Includes/commun.h"

/* ##### FONCTIONS

*/

/* -----

DerivationEgal

----- */

void DerivationEgal(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char
**att, char *fichier)

{

char *u;

char *t;

char *v;

```

char *w;
STRING temp, deg;
int pent;
int i,j;
float a,b;

/* Ajout de l'attribut a la gauche du egal a la liste des attributs internes*/
u=(char *) malloc(sizeof(char)*1024);
/*strcpy(u,"");
strcat(u,pointeur->fg->Type.type);*/
strcpy(u,"");
strcat(u,pointeur->fg->Type.type);
strcat(u,"");

printf("[DerivationEgal]>> pointeur->Type.type : %s\n",pointeur->Type.type);
printf("[DerivationEgal]>> pointeur->Genre : %d\n",pointeur->Genre);
printf("[DerivationEgal]>> pointeur->Degre : %f\n",pointeur->Degre);
printf("[DerivationEgal]>> pointeur->fg->Type.type: %s\n",u);
printf("[DerivationEgal]>> pointeur->fg->Genre %d\n",pointeur->fg->Genre);
printf("[DerivationEgal]>> pointeur->fg->Degre: %f\n",pointeur->fg->Degre);
printf("[DerivationEgal]>> pointeur->fd->Type.type: %s\n",pointeur->fd->Type.type);fflush
(stdout);
printf("[DerivationEgal]>> pointeur->fd->Genre %d\n",pointeur->fd->Genre);
printf("[DerivationEgal]>> pointeur->fd->Degre: %f\n",pointeur->fd->Degre);
*att=u;

printf("liste attributs %s\n",*att);
v=(char *) malloc(sizeof(char)*1024);
strcpy(v,"1");
if (pointeur->fd->Genre==2) /* cas d'un flottant comme valeur */
{
    /*t=(char *) malloc(sizeof(char)*1024);
    strcpy(t,pointeur->fg->Type.type);
    strcat(t," < ");
    sprintf(deg,"%f",pointeur->fd->Type.numf);
    strcat(t,deg);
    *cd = t; */
    t=(char *) malloc(sizeof(char)*1024);
    strcpy(t,"CalculDegreFloat");
    strcat(t,"1");
    strcat(t,"");
    strcat(t,pointeur->fg->Type.type);
    strcat(t,"");
    sprintf(deg,"%f",pointeur->fd->Type.numf);
    strcat(t,deg);
    strcat(t,"");
    *fa=t;
    /**fa=v*/

return;
}

```

```

t=(char *) malloc(sizeof(char)*1024);
strcpy(t,"CalculValAttributToLambda(");
strcat(t,Rendre_attribut(pointeur->fd->Type.type));
strcat(t,",");
strcat(t,"");
strcat(t,Rendre_attribut(pointeur->fg->Type.type));
strcat(t,"");
strcat(t,",");

if (strcmp (Rendre_relation(pointeur->fg->Type.type), "")!=0)
{
    strcat(t,Rendre_relation(pointeur->fg->Type.type));
    strcat(t," _");
}
strcat(t,Rendre_attribut(pointeur->fg->Type.type));
strcat(t,",");
strcat(t,"");
strcat(t,fichier);
strcat(t,"");
strcat(t,"");
*fa = t;
}
/* -----
DerivationDiffer
entree: pointeur : pointeur sur un noeud de l'arbre de derivation
      signe   : permet de connaitre le valeur que l'on desire recuperer
      lambda  : valeur du lambda
sorie: cd     : pointeur sur la chaine representant la condition derivee
      fa      : pointeur sur la chaine representant la fonction associee
attributs_internes:chaine de caracteres dans laquelle figure tous
                  les attributs utilise a l'interieur du where
----- */
void DerivationDiffer(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char
**attribut_internes)
{
    char *t;
    char *u;
    char deg[256];
    STRING temp;
    int i,j;
    char *v;

    v=(char *) malloc(sizeof(char)*1024);
    strcpy(v,"1");

    u=(char *) malloc(sizeof(char)*1024);
    strcpy(u,"");

```

```

    strcat(u,pointeur->fg->Type.type);
    strcat(u,"");
    *attribut_interne=u;
    printf("liste attributs %s\n",*attribut_interne);

/* ===== */
/* Derivation du cas==> ATT != Valeur */
/* ===== */
if (pointeur->fd->Genre==2) /* cas d'un flottant comme valeur */
{

    t=(char *) malloc(sizeof(char)*1024);
    strcpy(t,"CalculDegreFloat");
    strcat(t,"2");
    strcat(t,"");
    strcat(t,pointeur->fg->Type.type);
    strcat(t,"");
    sprintf(deg,"%f",pointeur->fd->Type.numf);
    strcat(t,deg);
    strcat(t,"");
    *fa=t;

    return;
}
/* ===== */
/* Derivation du cas==> ATT != "CHAINE" */
/* ===== */
else if (pointeur->fd->Type.type[0] == "")
{
    if(pointeur->fd->Type.type[1] != '%')
    {
        /* cas d'une chaine de caractere pas terme flou */
        /* on va nettoyer les " au tour de la chaine par exemple "JEUNE" ==>
JEUNE */
        j=0;
        strcpy(temp,pointeur->fd->Type.type);
        for(i=1;i<(strlen(pointeur->fd->Type.type)-1);i++)
        {
            temp[j] = pointeur->fd->Type.type[i];
            j++;
        }
        temp[j]='\0';

        t=(char *) malloc(sizeof(char)*1024);
        strcpy(t,pointeur->fg->Type.type);
        strcat(t," != ");
        strcat(t,"");
        strcat(t,temp);
        strcat(t,"");
    }
}

```



```

/* on va tester s'il ne s'agit pas d'un cas simple attribut < valeur */

if (pointeur->fd->Genre==2) /* cas d'un flottant comme valeur */
{

    t=(char *) malloc(sizeof(char)*1024);
    strcpy(t,"CalculDegreFloat(");
    strcat(t,"3");
    strcat(t,",");
    strcat(t,pointeur->fg->Type.type);
    strcat(t,",");
    sprintf(deg,"%f",pointeur->fd->Type.numf);
    strcat(t,deg);
    strcat(t,")");
    *fa=t;

    return;
}

/*Calcul de la fonction d'appartenance*/
t=(char *) malloc(sizeof(char)*1024);
strcpy(t,"CalculValAttributToLambda(");
strcat(t,pointeur->fd->Type.type);
strcat(t,",");
strcat(t,"\"");
strcat(t,pointeur->fg->Type.type);
strcat(t,"\"");
strcat(t,",");
strcat(t,pointeur->fg->Type.type);
strcat(t,",");
strcat(t,"\"");
strcat(t,fichier);
strcat(t,"\"");
strcat(t,")");
*fa = t;
}
/* -----
    DerivationInfEgal (<= inferieur ou egal)
entree: pointeur : pointeur sur un noeud de l'arbre de derivation
    signe   : permet de connaitre le valeur que l'on desire recuperer
           lambda : valeur du lambda
sorie: cd    : pointeur sur la chaine representant la condition derivee
    fa      : pointeur sur la chaine representant la fonction associee
attributs_internes:chaine de caracteres dans laquelle figure tous
                    les attributs utilise a l'interieur du where
----- */
void DerivationInfEgal(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char
**attribut_internes, char *fichier)
{
    char *t;

```

```

char *u;
char deg[256];
char *v;

v=(char *) malloc(sizeof(char)*1024);
strcpy(v,"1");

/* Ajout de l'attribut a la gauche du egal a la liste des attributs internes*/

u=(char *) malloc(sizeof(char)*1024);
strcpy(u,"");
strcat(u,pointeur->fg->Type.type);
strcat(u,"");
*attribut_internes=u;
printf("liste attributs %s\n",*attribut_internes);

/* on va tester s'il ne s'agit pas d'un cas simple attribut < valeur */

if (pointeur->fd->Genre==2) /* cas d'un flottant comme valeur */
{

    t=(char *) malloc(sizeof(char)*1024);
    strcpy(t,"CalculDegreFloat");
    strcat(t,"5");
    strcat(t,"");
    strcat(t,pointeur->fg->Type.type);
    strcat(t,"");
    sprintf(deg,"%f",pointeur->fd->Type.numf);
    strcat(t,deg);
    strcat(t,"");
    *fa=t;

return;

}

/*Calcul de la fonction d'appartenance*/
t=(char *) malloc(sizeof(char)*1024);
strcpy(t,"CalculValAttributToLambda");
strcat(t,pointeur->fd->Type.type);
strcat(t,"");
strcat(t,"");
strcat(t,pointeur->fg->Type.type);
strcat(t,"");
strcat(t,"");
strcat(t,"");
strcat(t,pointeur->fg->Type.type);
strcat(t,"");
strcat(t,"");

```

```

        strcat(t,fichier);
        strcat(t,"\\");
        strcat(t,"");
        *fa = t;
    }
    /* -----
        DerivationSup (strictement >)
entree: pointeur : pointeur sur un noeud de l'arbre de derivation
        signe   : permet de connaitre le valeur que l'on desire recuperer
        lambda   : valeur du lambda
sorie: cd       : pointeur sur la chaine representant la condition derivee
        fa       : pointeur sur la chaine representant la fonction associee
        attributs_internes:chaine de caracteres dans laquelle figure tous
                        les attributs utilise a l'interieur du where
    ----- */
void DerivationSup(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char
**attribut_internes, char *fichier)
{
    char *t;
    char *u;
    char deg[256];
    char *v;

    v=(char *) malloc(sizeof(char)*1024);
    strcpy(v,"1");

    /* Ajout de l'attribut a la gauche du egal a la liste des attributs internes*/
    /*u=(char *) malloc(sizeof(char)*1024);
    strcpy(u,"");
    strcat(u,pointeur->fg->Type.type);
    *attribut_internes=u;*/

    u=(char *) malloc(sizeof(char)*1024);
    strcpy(u,"");
    strcat(u,pointeur->fg->Type.type);
    strcat(u,"");
    *attribut_internes=u;
    printf("liste attributs %s\\n",*attribut_internes);

    /* on va tester s'il ne s'agit pas d'un cas simple attribut < valeur */

    if (pointeur->fd->Genre==2) /* cas d'un flottant comme valeur */
    {
        /*t=(char *) malloc(sizeof(char)*1024);
        strcpy(t,pointeur->fg->Type.type);
        strcat(t," > ");
        sprintf(deg,"%f",pointeur->fd->Type.numf);
        strcat(t,deg);
        *cd = t;
        *fa=v;*/
        t=(char *) malloc(sizeof(char)*1024);

```

```

        strcpy(t,"CalculDegreFloat(");
        strcat(t,"4");
        strcat(t,",");
        strcat(t,pointeur->fg->Type.type);
        strcat(t,",");
        sprintf(deg,"%f",pointeur->fd->Type.numf);
        strcat(t,deg);
        strcat(t,")");
        *fa=t;

    return;
}

/*Calcul de la fonction d'appartenance*/
t=(char *) malloc(sizeof(char)*1024);
strcpy(t,"CalculValAttributToLambda(");
strcat(t,pointeur->fd->Type.type);
strcat(t,",");
strcat(t,"\"");
strcat(t,pointeur->fg->Type.type);
strcat(t,"\"");
strcat(t,",");
strcat(t,pointeur->fg->Type.type);
strcat(t,",");
strcat(t,"\"");
strcat(t,fichier);
strcat(t,"\"");
strcat(t,")");
*fa = t;
}
/* -----
        DerivationSupEgal (>=)
entree: pointeur : pointeur sur un noeud de l'arbre de derivation
        signe   : permet de connaitre le valeur que l'on desire recuperer
                lambda : valeur du lambda
sorie: cd      : pointeur sur la chaine representant la condition derivee
        fa      : pointeur sur la chaine representant la fonction associee
attributs_interne:chaine de caracteres dans laquelle figure tous
                  les attributs utilise a l'interieur du where
----- */
void DerivationSupEgal(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char
**attribut_interne, char *fichier)
{
    char *t;
    char *u;
    char deg[256];
    char *v;

    v=(char *) malloc(sizeof(char)*1024);
    strcpy(v,"1");

```

```

/* Ajout de l'attribut a la gauche du egal a la liste des attributs internes*/
/*u=(char *) malloc(sizeof(char)*1024);
strcpy(u,"");
strcat(u,pointeur->fg->Type.type);
*attribut_internes=u;*/
u=(char *) malloc(sizeof(char)*1024);
strcpy(u,"");
strcat(u,pointeur->fg->Type.type);
strcat(u,"");
*attribut_internes=u;
printf("liste attributs %s\n",*attribut_internes);

```

/* on va tester s'il ne s'agit pas d'un cas simple attribut < valeur */

```

if (pointeur->fd->Genre==2) /* cas d'un flottant comme valeur */
{
    /*t=(char *) malloc(sizeof(char)*1024);
    strcpy(t,pointeur->fg->Type.type);
    strcat(t," >= ");
    sprintf(deg,"%f",pointeur->fd->Type.numf);
    strcat(t,deg);
    *cd = t;
    *fa=v;*/
    t=(char *) malloc(sizeof(char)*1024);
    strcpy(t,"CalculDegreFloat");
    strcat(t,"6");
    strcat(t,"");
    strcat(t,pointeur->fg->Type.type);
    strcat(t,"");
    sprintf(deg,"%f",pointeur->fd->Type.numf);
    strcat(t,deg);
    strcat(t,"");
    *fa=t;

    return;
}

```

```

/*Calcul de la fonction d'appartenance*/
t=(char *) malloc(sizeof(char)*1024);
strcpy(t,"CalculValAttributToLambda");
strcat(t,pointeur->fd->Type.type);
strcat(t,"");
strcat(t,"");
strcat(t,pointeur->fg->Type.type);
strcat(t,"");
strcat(t,"");
strcat(t,pointeur->fg->Type.type);

```

```

        strcat(t, ",");
        strcat(t, "\"");
        strcat(t, fichier);
        strcat(t, "\"");
        strcat(t, " ");
        *fa = t;
    }

/* -----
    DerivationAnd
entree: pointeur : pointeur sur un noeud de l'arbre de derivation
        signe   : permet de connaitre le valeur que l'on desire recuperer
        lambda   : valeur du lambda
sorie:  cd       : pointeur sur la chaine representant la condition derivee
        fa       : pointeur sur la chaine representant la fonction associee
        attributs_internes: chaine de caracteres dans laquelle figure tous
                           les attributs utilise a l'interieur du where
----- */
void DerivationAND(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char
**attribut_internes, char *fichierTF, char
*fichierMF, char *fichierCF, char *fichierQF)
{
    char *cd1;
    char *cd2;
    char *fa1;
    char *fa2;
    char *att1;
    char *att2;
    char *t;
    char *v;
    char *u;

    /* On regarde si le noeud suivant dans l'arbre est un exists, au quel cas il faut
        agir d'une maniere differente */
    if(strcmp(pointeur->fd->Type.type, "EXISTS")==0)
    {
        /*Calcul des conditions derivees*/
        Derivation(pointeur->fg, signe, lambda, &cd1, &fa1, &att1, fichierTF, fichierMF,
fichierCF, fichierQF);
        printf("[DerivationAND]>> condition derivee exterieure: %s\n", cd1);
        printf("[DerivationAND]>> fonction associee exterieure: %s\n", fa1);
        Derivation(pointeur->fd, signe, lambda, &cd2, &fa2, &att2, fichierTF, fichierMF,
fichierCF, fichierQF);
        printf("[DerivationAND]>> condition derivee interieure: %s\n", cd2);
        printf("[DerivationAND]>> fonction associee interieure: %s\n", fa2);

    }
    else
    {

        /*Calcul de la condition derivee*/

```

```
/*On derive les expressions de gauche et de droite */
```

```
Derivation(pointeur->fg,signe,lambda,&cd1,&fa1, &att1, fichierTF, fichierMF,
fichierCF, fichierQF);
```

```
Derivation(pointeur->fd,signe,lambda,&cd2,&fa2, &att2, fichierTF, fichierMF,
fichierCF, fichierQF);
```

```
/*Calcul de la fonction d'appartenance*/
```

```
v=(char *) malloc(sizeof(char)*1024);
strcpy(v,"Minimum(");
strcat(v,fa1);
strcat(v,",");
strcat(v,fa2);
strcat(v,")");
*fa = v;
```

```
/* Ajout des attributs retourner a la liste des attributs internes */
```

```
u=(char *) malloc(sizeof(char)*1024);
strcpy(u,att1);
strcat(u,att2);
*attribut_interne=u;
```

```
}
```

```
}/* -----
```

```
DerivationOR
```

```
entree: pointeur : pointeur sur un noeud de l'arbre de derivation
```

```
signe : permet de connaitre le valeur que l'on desire recuperer
```

```
lambda : valeur du lambda
```

```
sorie: cd : pointeur sur la chaine representant la condition derivee
```

```
fa : pointeur sur la chaine representant la fonction associee
```

```
attributs_interne : chaine de caracteres dans laquelle figure tous
les attributs utilise a l'interieur du where
```

```
----- */
```

```
void DerivationOR(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char
**attribut_interne
```

```
,char *fichierTF, char *fichierMF, char *fichierCF, char *fichierQF)
```

```
{
```

```
char *cd1,*cd2;
```

```
char *fa1,*fa2;
```

```
char *t,*v,*u;
```

```
char *att1, *att2;
```

```
/*Calcul de la condition derivee*/
```

```

/*On derive les expressions de gauche et de droite */
Derivation(pointeur->fg,signe,lambda,&cd1,&fa1, &att1, fichierTF, fichierMF, fichierCF,
fichierQF);
Derivation(pointeur->fd,signe,lambda,&cd2,&fa2, &att2, fichierTF, fichierMF, fichierCF,
fichierQF);

/* plusieurs cas sont a considerer */

/*Calcul de la fonction d'appartenance*/
v=(char *) malloc(sizeof(char)*1024);
strcpy(v,"Maximum(");
strcat(v,fa1);
strcat(v,",");
strcat(v,fa2);
strcat(v,")");
*fa = v;
/*}*/
/* Ajout des attributs retourner a la liste des attributs internes */
u=(char *) malloc(sizeof(char)*1024);
strcpy(u,att1);
strcat(u,att2);
*attribut_internes=u;

}
/* -----
DerivationNOT
entree: pointeur : pointeur sur un noeud de l'arbre de derivation
signe : permet de connaitre le valeur que l'on desire recuperer
lambda : valeur du lambda
sorie: cd : pointeur sur la chaine representant la condition derivee
fa : pointeur sur la chaine representant la fonction associee
attributs_internes:chaine de caracteres dans laquelle figure tous
les attributs utilise a l'interieur du where
----- */
void DerivationNOT(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char
**att,char *fichierTF, char *fichierMF,
char *fichierCF, char *fichierQF)
{
char *cd1;
char *cd2;
char *fa1;
char *att1;
char *t;
float nl;

/* Calcul du nouveau lambda */
nl = 1-lambda;
printf("[DerivationNOT]>> Nouveau degre calcule : %f\n",nl);
Derivation(pointeur->fd,(1-sign),nl,&cd1,&fa1, &att1, fichierTF, fichierMF, fichierCF,
fichierQF);

```

```

printf("valeur de FA1 :%s donc celle a comparer\n",fa1);
if (fa1[0]=='1'){
    printf("flou\n");
    cd2=(char *) malloc(sizeof(char)*1024);
    strcpy(cd2,"not(");
    strcat(cd2,cd1);
    strcat(cd2,")");
    *cd=cd2;
    *att=att1;
    t=(char *) malloc(sizeof(char)*1024);
    strcpy(t,"(");
    strcat(t,fa1);
    strcat(t,")");
    *fa = t;
}
else{
    printf("pas flou\n");
    *cd=cd1;
    *att=att1;

    /*Calcul de la fonction d'appartenance*/
    t=(char *) malloc(sizeof(char)*1024);
    strcpy(t,"(1-");
    strcat(t,fa1);
    strcat(t,")");
    *fa = t;
}
}
/* -----
    DerivationAM (Dérivation Moyenne Arithmétique)
entree: pointeur : pointeur sur un noeud de l'arbre de derivation
    signe   : permet de connaitre le valeur que l'on desire recuperer
    lambda  : valeur du lambda
sortie: cd    : pointeur sur la chaine representant la condition derivee
    fa      : pointeur sur la chaine representant la fonction associee
    att     : pointeur sur la chaine qui conserve les attributs utilises en internes
----- */
void DerivationAM(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char
**att,char *fichierTF, char *fichierMF, char *fichierCF, char *fichierQF)
{
    int cpt;
    struct Maillon *pt;
    int i;
    char *cd1,*fa1,*att1;
    char *cdi, *fai, *atti;
    float nl, n; /* nouveau lambda */
    STRING tmp;

    cdi=(char *) malloc(sizeof(char)*1024);
    fai=(char *) malloc(sizeof(char)*1024);
    atti=(char *) malloc(sizeof(char)*1024);

```

```

/* on recupere le nombre de termes existant dans la moyenne */
n = pointeur->Degre; /* ca sert pour calculer le nouveau lambda */
cpt = pointeur->Degre; /* ca sert pour parcourir la chaine d'elements */

pt = pointeur->fd; /* pointeur a la chaine de predicats*/

strcpy(fai,"");

/*le nouveau lambda depend du signe de derivation*/
nl = (signe == 1) ? max(0, n*(lambda-1)+1): min(1, n*lambda);
/*nl = max(0, n*(lambda-1)+1);*/
printf("[DerivationAM]>> Nouveau lambda : %f\n",nl);

while(pt != 0)
{

    /* Derivation de l'expression avec le nouveau lambda */
    Derivation(pt,signe,nl,&cd1,&fa1, &att1, fichierTF, fichierMF, fichierCF,fichierQF);

    /* Elaboration de la chaine des attributs internes */
    strcat(att1,att1);

    /* Elaboration de la fonction d'appartenance */
    strcat(fai,fa1);
    if(--cpt!=0) strcat(fai,"+");

    /* Elaboration de la condition derivee */
    strcat(cdi,cd1);
    strcat(cdi," AND ");

    /* On passe au suivante */
    pt = pt->succ;
}
/* Elaboration de la fonction d'appartenance */
strcat(fai,"/");
sprintf(tmp,"%f",n);
strcat(fai,tmp);

pt = pointeur->fd; /* pointeur encore a tete de la chaine de predicats*/
cpt = n; /* et on reinitialise aussi le compteur*/
strcat(cdi,"");
while(pt != 0)
{

    printf("[DerivationAM]>> terme: %s\n",pt->fd->Type.type);
    /* Derivation de l'expression avec le ancien lambda */
    Derivation(pt,signe,lambda,&cd1,&fa1, &att1, fichierTF, fichierMF,
fichierCF,fichierQF);

```

```

        strcat(cdi,cd1);
        if(--cpt!=0) strcat(cdi," OR ");

        /* On passe au suivante */
        pt = pt->succ;
    }
    strcat(cdi,"");
    /* Affectation des valeurs de retour */
    *att = atti;
    *fa = fai;
    *cd = cdi;
}

/* -----
    DerivationGM (Dérivation Moyenne Géométrique)
entree: pointeur : pointeur sur un noeud de l'arbre de derivation
    signe : permet de connaitre le valeur que l'on desire recuperer
    lambda : valeur du lambda
sortie: cd : pointeur sur la chaine representant la condition derivee
    fa : pointeur sur la chaine representant la fonction associee
    attr : pointeur sur la chaine qui conserve les attributs utilises en internes
----- */
void DerivationGM(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char
**att,char *fichierTF, char *fichierMF, char *fichierCF, char *fichierQF)
{
    int cpt;
    struct Maillon *pt;
    int i;
    char *cd1,*fa1,*att1;
    char *cdi, *fai, *atti;
    float nl, n; /* nouveau lambda */
    STRING tmp;

    cdi=(char *) malloc(sizeof(char)*1024);
    fai=(char *) malloc(sizeof(char)*1024);
    atti=(char *) malloc(sizeof(char)*1024);

    /* on recupere le nombre de termes existant dans la moyenne */
    n = pointeur->Degre; /* ca sert pour calculer le nouveau lambda */
    cpt = pointeur->Degre; /* ca sert pour parcourir la chaine d'elements */
    printf("[DerivationGM]>> nombre de termes existant dans la moyenne : %f\n",n);
    printf("[DerivationGM]>> nombre de termes existant dans la moyenne : %d\n",cpt);

    pt = pointeur->fd; /* pointeur a la chaine de predicats*/

    strcpy(fai,"pow(");

    /* Calcul du nouveau lambda */
    nl = pow(lambda, n);
    printf("[DerivationGM]>> Nouveau lambda : %f\n",nl);

```

```

while(pt != 0)
{
    /* Derivation de l'expression avec le nouveau lambda */
    Derivation(pt,signe,nl,&cd1,&fa1, &att1, fichierTF, fichierMF, fichierCF,fichierQF);

    /* Elaboration de la chaine des attributs internes */
    strcat(att1,att1);

    /* Elaboration de la fonction d'appartenance et de la condition derivee*/
    strcat(fai,fa1);
    strcat(cdi,cd1);
    if(--cpt!=0)
    {
        strcat(fai,"*");
        strcat(cdi," AND ");
    }
    /* On passe au suivante */
    pt = pt->succ;
}
/* fin de la fonction d'appartenance */
strcat(fai,", (1/");
sprintf(tmp,"%f",n);
strcat(fai,tmp);
strcat(fai,")");
/* Affectation des valeurs de retour */
*att = att1;
*fa = fai;
*cd = cdi;
}
/* -----

```

DerivationIN

entree: pointeur : pointeur sur un noeud de l'arbre de derivation
 signe : permet de connaitre le valeur que l'on desire recuperer
 lambda : valeur du lambda
 sortie: cd : pointeur sur la chaine representant la condition derivee
 fa : pointeur sur la chaine representant la fonction associee
 attr : pointeur sur la chaine qui conserve les attributs utilises en internes

```

----- */
void DerivationIN(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char
**att,char *fichierTF, char *fichierMF, char *fichierCF, char *fichierQF)
{
    int cpt;
    struct Maillon *pt;
    int i;
    char *cd1,*fa1,*att1;
    float nl, n; /* nouveau lambda */
    STRING tmp;

    cd1=(char *) malloc(sizeof(char)*1024);
    fa1=(char *) malloc(sizeof(char)*1024);
    att1=(char *) malloc(sizeof(char)*1024);

```

```

/*
printf("[DerivationIN]>> pointeur->Type.type : %s\n",pointeur->Type.type);
printf("[DerivationIN]>> pointeur->Genre : %d\n",pointeur->Genre);
printf("[DerivationIN]>> pointeur->Degre : %f\n",pointeur->Degre);fflush(stdout);
*/
if (strcmp("SELECT",pointeur->fd->Type.type)== 0) /* c'est une requete imbriquee dans le
IN */
{
    printf("[DerivationIN]>> requete imbriquee\n");fflush(stdout);
    Derivation(pointeur->fd, POSITIF, lambda, cd, fa, att, fichierTF,
fichierMF,fichierCF,fichierQF);
}
else /* c'est une liste de constantes dans le IN */
{
    /* on recupere le nombre de termes existant dans la moyenne */
    cpt = pointeur->Degre;      /* ca sert pour parcourir la chaine d'elements */
    printf("[DerivationIN]>> nombre de termes dans la liste : %d\n",cpt);fflush(stdout);
    strcpy(cd1,pointeur->fg->Type.type);
    strcat(cd1," ");
    strcat(cd1,pointeur->Type.type);
    strcat(cd1," (");

    /* parcour des elements */
    pt = pointeur->fd;      /* pointeur a la chaine des constantes */
    while(cpt != 0)
    {
        /* Elaboration de la chaine des attributs internes */
        strcat(cd1,pt->Type.type);
        if ((cpt-1)>0) strcat(cd1," ");

        /* On passe au suivant */
        pt = pt->succ;
        cpt--;
    }
    /* fin de la fonction d'appartenance */
    strcat(fa1,"1");
    strcat(cd1,")");
    /* Affectation des valeurs de retour */
    *cd = cd1;
    *fa = fa1;
}
}

/* -----
DerivationCompareurFlouAtt

```

Exemple: >>,...

Fonction de derivation d'un Compareur flou dans le cas ou le terme
situe a droite de l'operateur est un attribut

entree: pointeur : pointeur sur un noeud de l'arbre de derivation

 signe : permet de connaitre le valeur que l'on desire recuperer

 lambda : valeur du lambda

```

sorie: cd      : pointeur sur la chaine representant la condition derivee
fa      : pointeur sur la chaine representant la fonction associee
attributs_internes:chaine de caracteres dans laquelle figure tous
                  les attributs utilise a l'interieur du where
----- */
void DerivationCompareurFlouAtt(struct Maillon *pointeur, int signe, float lambda, char **cd,
char **fa, char **attribut_internes,char *fichierCF)
{
    char *t;
    char *u;
    char deg[256];
    char cderiv[256];
    char *condition_derivee;
    float *resultat;

    resultat=(float *) malloc(sizeof(float)*3);
    u=(char *) malloc(sizeof(char)*1024);
    strcpy(u,"");
    strcat(u,pointeur->fg->Type.type);
    strcat(u,"");
    strcat(u,pointeur->fd->Type.type);
    *attribut_internes = u;

    /* Calcul de l'intervalle dans lequel doivent se situer les
n-uplets afin de satisfaire le degre LAMBDA (intervalle de satisfaction) */
    resultat = CalculValeurCF(pointeur->Type.type, lambda, signe,fichierCF);
    /* Demande d'informations supplementaire concernant le compareur flou */
    printf("[DerivationCompareurFlouAtt] >> avant construit condition a derivier =%s %s %
s\n ",pointeur->fg->Type.type,pointeur->Type.type,pointeur->fd->Type.type);

    RetourneInfoCompareurFlou(pointeur->Type.type,&condition_derivee,fichierCF);

    /* Calcul de la condition derivee */

    strcpy(cderiv,pointeur->fg->Type.type);
    strcat(cderiv,condition_derivee);
    strcat(cderiv,pointeur->fd->Type.type);

    printf("[DerivationCompareurFlouAtt] >> apres construit condition a derivier = %s \n
",cderiv);

    if(resultat[2]==POSITIF)
    {
        t=(char *) malloc(sizeof(char)*1024);
        strcpy(t,"");
        strcat(t,cderiv);
        strcat(t,"");
        strcat(t," >= ");
    }
}

```

```

        sprintf(deg,"%0.1f",resultat[0]);
        strcat(t,deg);
        *cd = t;
    }
    else
    {
        if(resultat[2]==NEGATIF)
        {
            t=(char *) malloc(sizeof(char)*1024);
            strcpy(t,"");
            strcat(t,cderiv);
            strcat(t,"");
            strcat(t," <= ");
            sprintf(deg,"%0.1f",resultat[0]);
            strcat(t,deg);
            *cd = t;
        }
        else
        {
            t=(char *) malloc(sizeof(char)*1024);
            sprintf(deg,"%0.1f",resultat[0]);
            strcpy(t,deg);
            strcat(t," <= ");
            strcat(t,cderiv);
            strcat(t," AND ");
            strcat(t,cderiv);
            strcat(t," <= ");
            sprintf(deg,"%0.1f",resultat[1]);
            strcat(t,deg);
            *cd = t;
        }
    }
}

```

```

/*Calcul de la fonction d'appartenance*/
t=(char *) malloc(sizeof(char)*1024);
strcpy(t,"CalculValeurVLCF(");
strcat(t,"");
strcat(t,pointeur->Type.type);
strcat(t,"");
strcat(t,"");
strcat(t,"");
strcat(t,modifie_mask(cderiv));
strcat(t,"");
strcat(t,"");
strcat(t,"");
strcat(t,fichierCF);
strcat(t,"");
strcat(t,"");
*fa = t;

```

```

}
/* -----

```

DerivationCompareurFlouAtt

Exemple: >>,...

Fonction de derivation d'un Compareur flou dans le cas ou le terme situe a droite de l'operateur est un attribut

entree: pointeur : pointeur sur un noeud de l'arbre de derivation

 signe : permet de connaitre le valeur que l'on desire recuperer

 lambda : valeur du lambda

sortie: cd : pointeur sur la chaine representant la condition derivee

 fa : pointeur sur la chaine representant la fonction associee

 attributs_internes: chaine de caracteres dans laquelle figure tous les attributs utilise a l'interieur du where

```
----- */
void DerivationCompareurFlouFloat(struct Maillon *pointeur, int signe, float lambda, char **cd,
char **fa, char **attribut_internes, char *fichierCF)
{
```

```
    char *t;
    char *u;
    char deg[256];
    char valeur[256];
    char cderiv[256];
    char *condition_derivee;
    float *resultat;
```

```
    resultat=(float *) malloc(sizeof(float)*3);
    u=(char *) malloc(sizeof(char)*1024);
    strcpy(u,"");
    strcat(u,pointeur->fg->Type.type);
```

```
    *attribut_internes = u;
```

```
    /* Calcul de l'intervalle dans lequel doivent se situer les
n-uplets afin de satisfaire le degre LAMBDA (intervalle de satisfaction) */
    resultat = CalculValeurCF(pointeur->Type.type, lambda, signe,fichierCF);
```

```
    RetourneInfoCompareurFlou(pointeur->Type.type,&condition_derivee,fichierCF);
    /* Calcul de la condition derivee */
```

```
    strcpy(cderiv,pointeur->fg->Type.type);
    strcat(cderiv,condition_derivee);
```

```
    sprintf(valeur,"%f",pointeur->fd->Type.numf);
```

```
    strcat(cderiv,valeur);
    if(resultat[2]==POSITIF)
    {
```

```
        t=(char *) malloc(sizeof(char)*1024);
        strcpy(t,"");
        strcat(t,cderiv);
        strcat(t,"");
```

```

        strcat(t, " >= ");
        sprintf(deg, "%.1f", resultat[0]);
        strcat(t, deg);
        *cd = t;
    }
    else if(resultat[2]==NEGATIF)
    {
        t=(char *) malloc(sizeof(char)*1024);
        strcpy(t, "(");
        strcat(t, cderiv);
        strcat(t, ")");
        strcat(t, " <= ");
        sprintf(deg, "%.1f", resultat[0]);
        strcat(t, deg);
        *cd = t;
    }
    else
    {
        t=(char *) malloc(sizeof(char)*1024);
        sprintf(deg, "%.1f", resultat[0]);
        strcpy(t, deg);
        strcat(t, " <= ");
        strcat(t, cderiv);
        strcat(t, " AND ");
        strcat(t, cderiv);
        strcat(t, " <= ");
        sprintf(deg, "%.1f", resultat[1]);
        strcat(t, deg);
        *cd = t;
    }

    /*Calcul de la fonction d'appartenance*/
    t=(char *) malloc(sizeof(char)*1024);
    strcpy(t, "CalculValeurVLCF(");
    strcat(t, "\"");
    strcat(t, pointeur->Type.type);
    strcat(t, "\"");
    strcat(t, ", ");
    strcat(t, "(");
    strcat(t, cderiv);
    strcat(t, ")");
    strcat(t, ", ");
    strcat(t, "\"");
    strcat(t, fichierCF);
    strcat(t, "\"");
    strcat(t, ")");
    *fa = t;
}

/* -----
DerivationModificateurFlouAtt

```

Exemple: tres, extremement

Fonction de derivation d'un operateur flou dans le cas ou le terme
situe a droite de l'operateur est un attribut

entree: pointeur : pointeur sur un noeud de l'arbre de derivation

 signe : permet de connaitre le valeur que l'on desire recuperer

 lambda : valeur du lambda

sortie: cd : pointeur sur la chaine representant la condition derivee

 fa : pointeur sur la chaine representant la fonction associee

attributs_internes: chaine de caracteres dans laquelle figure tous
les attributs utilise a l'interieur du where

```
----- */
void DerivationModificateurFlou(struct Maillon *pointeur, int signe, float lambda, char **cd, char
**fa, char **attribut_internes, char *fichierTF, char *fichierMF, char *fichierCF, char *fichierQF)
{
    char *cd1;
    char *fa1;
    char deg[256];
    char *t;
    float val;
    float nl=0;          /*nouveau lambda*/

    /*Calcul de la condition derivee*/
    val = RetournePuissanceModificateurFlou(pointeur->Type.type, fichierMF);
    if (val != 0)
    {
        nl = powf(lambda, (1/val));
        Derivation(pointeur->fd, signe, nl, &cd1, &fa1, attribut_internes, fichierTF, fichierCF,
fichierMF, fichierQF);
        *cd=cd1;

        /*Calcul de la fonction d'appartenance*/
        t=(char *) malloc(sizeof(char)*1024);
        strcpy(t,"powf(");
        strcat(t,fa1);
        strcat(t,"");
        sprintf(deg,"%0.1f",RetournePuissanceModificateurFlou(pointeur-
>Type.type,fichierMF));
        strcat(t,deg);
        strcat(t,"");
        *fa = t;
    }
}
```

```
/* -----
DerivationQuantificateurFlou
----- */
```

```
void DerivationQuantificateurFlou
(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char **att, char
*fichierTF, char *fichierMF, char *fichierCF, char *fichierQF)
{
    int n;
```

```

struct Maillon *pt;
int i;
char *cd1,*fa1,*att1;
char *cdi, *fai, *atti;
float *liste; /* liste avec les degre du quantificateur*/
STRING tmp;

cdi=(char *) malloc(sizeof(char)*1024);
fai=(char *) malloc(sizeof(char)*1024);
atti=(char *) malloc(sizeof(char)*1024);

n = pointeur->Degre; /* nombre de predicats */
pt = pointeur->fd; /* pointeur a la chaine de predicats*/
DegreQuantificateur( pointeur->Type.type, n, &liste, fichierQF );

sprintf(cdi,"[");
sprintf(fai,"Quant(%d,Q[" , n);
sprintf(tmp, "%f",liste[0]);
strcat(fai,tmp);
for(i=1;i<n;i++) /*On a enleve le dernier valeur du quantificateur, parce que
c'es toujours 1*/
{
    strcat(fai, "; ");
    sprintf(tmp, "%f",liste[i]);
    strcat(fai,tmp);
}
strcat(fai, "], P[");

for(i=1;i<n;i++)
{
    Derivation(pt,signe,lambda,&cd1,&fa1, &att1, fichierTF, fichierMF,
fichierCF,fichierQF);
    /* Elaboration de la chaine des attributs internes */
    strcat(atti,att1);
    /* Elaboration de la fonction d'appartenance */
    strcat(fai,fa1);
    strcat(fai,"; ");
    /* Elaboration de la condition derivee */
    strcat(cdi,cd1);
    strcat(cdi,"; ");
    pt = pt->succ;
}
Derivation(pt,signe,lambda,&cd1,&fa1, &att1, fichierTF, fichierMF, fichierCF,fichierQF);
/* Elaboration de la chaine des attributs internes */
strcat(atti,att1);
/* Elaboration de la fonction d'appartenance */
strcat(fai,fa1);
strcat(fai,"]");
/* Elaboration de la condition derivee */
strcat(cdi,cd1);
strcat(cdi,"]");

```

```

*att = atti;
*fa = fai;
*cd = cdi;
}

```

```

/* -----

```

DERIVATION

Fonction d'aiguillage.

Cette fonction effectue la derivation des regles de l'arbre semantique.

La regle de derivation est appliquee sur le sommet de l'arbre et le mecanisme se repete recursivement.

parametre d'entree:

pointeur : pointeur sur le maillon a deriver
lambda : degre de satisfaction (seuil)

parametre de sortie:

cd : chaine de caracteres representant la condition derivee
fa : chaine de caracteres representant la fonction de calcul du degre
att : chaine de caracteres dans laquelle figure tous
les attributs utilise a l'interieur du where

```

----- */

```

```

void Derivation(struct Maillon *pointeur, int signe, float lambda, char **cd, char **fa, char **att,
char*fichierTF, char *fichierMF,
char *fichierCF, char *fichierQF)
{

```

```

    if (pointeur==NULL) {
        printf("[Derivation.c]>> WHERE VIDE\n");
    }

```

```

    else if(strcmp("=",pointeur->Type.type) == 0)
    {
        printf("[Derivation.c]>> DERIVATION EGAL\n");
        DerivationEgal(pointeur, signe, lambda, cd, fa, att, fichierTF);
    }

```

```

    else if( (strcmp("<",pointeur->Type.type) == 0) || (strcmp("!=" ,pointeur->Type.type) ==
0) )

```

```

    {
        printf("[Derivation.c]>> DERIVATION DIFFERENT\n");
        DerivationDiffer(pointeur,signe,lambda, cd, fa, att);
    }

```

```

    else if( strcmp("<=",pointeur->Type.type) == 0)
    {
        printf("[Derivation.c]>> DERIVATION INFERIEUR-EGAL\n");
        DerivationInfEgal(pointeur,signe,lambda, cd, fa, att, fichierTF);
    }

```

```

    else if( strcmp("<",pointeur->Type.type) == 0 )
    {
        printf("[Derivation.c]>> DERIVATION INFERIEUR\n");
        DerivationInf(pointeur,signe,lambda, cd, fa, att, fichierTF);
    }
}

```

```

else if( strcmp(">=",pointeur->Type.type) == 0 )
{
    printf("[Derivation.c]>> DERIVATION SUPERIEUR-EGAL\n");
    DerivationSupEgal(pointeur,signe,lambda, cd, fa, att, fichierTF);
}
else if( strcmp(">",pointeur->Type.type) == 0 )
{
    printf("[Derivation.c]>> DERIVATION SUPERIEUR\n");
    DerivationSup(pointeur,signe,lambda, cd, fa, att, fichierTF);
}
else if(strcmp("AND",pointeur->Type.type) == 0)
{
    printf("[Derivation.c]>> DERIVATION AND\n");
    DerivationAND(pointeur, signe, lambda, cd, fa, att, fichierTF, fichierMF, fichierCF,
fichierQF);
}
else if(strcmp("OR",pointeur->Type.type) == 0)
{
    printf("[Derivation.c]>> DERIVATION OR\n");
    DerivationOR(pointeur, signe, lambda, cd, fa, att, fichierTF, fichierMF, fichierCF,
fichierQF);
}
else if(strcmp("NOT",pointeur->Type.type) == 0)
{
    printf("[Derivation.c]>> DERIVATION NOT\n");
    DerivationNOT(pointeur, signe, lambda, cd, fa, att, fichierTF, fichierMF, fichierCF,
fichierQF);
}
else if(strcmp("AM",pointeur->Type.type) == 0)
{
    printf("[Derivation.c]>> DERIVATION MOYENNE ARITHMETIQUE\n");
    DerivationAM(pointeur, signe, lambda, cd, fa, att, fichierTF, fichierMF, fichierCF,
fichierQF);
}
else if(strcmp("GM",pointeur->Type.type) == 0)
{
    printf("[Derivation.c]>> DERIVATION MOYENNE GEOMETRIQUE\n");
    DerivationGM(pointeur, signe, lambda, cd, fa, att, fichierTF, fichierMF, fichierCF,
fichierQF);
}
else if(strcmp("IN",pointeur->Type.type) == 0)
{
    printf("[Derivation.c]>> DERIVATION IN\n");
    DerivationIN(pointeur, signe, lambda, cd, fa, att, fichierTF, fichierMF, fichierCF,
fichierQF);
}
else if((RechercheCompareteurFlou(pointeur->Type.type,fichierCF) == 0) && (pointeur-
>fd->Genre == 1) )/*comp: >>,..*/
{
    /*Si pointeur->fd->Genre == 1 alors on sait que la partie droite de l'operateur
est un attribut */

```

```

printf("[Derivation.c]>> DERIVATION COMPARATEUR\n");
DerivationCompareteurFlouAtt(pointeur, signe, lambda, cd, fa, att, fichierCF);

}
else if(RechercheCompareteurFlou(pointeur->Type.type,fichierCF) == 0 && (pointeur->fd-
>Genre==2) )/*comp: >>,..*/
{
    /*Si pointeur->fd->Genre == 1 alors on sait que la partie droite de l'operateur
    est un flottant */
    printf("[Derivation.c]>> DERIVATION COMPARATEUR (cas Flottant)\n");
    DerivationCompareteurFlouFloat(pointeur, signe, lambda, cd, fa, att, fichierCF);

}
else if(RechercheModificateurFlou(pointeur->Type.type,fichierMF) == 0)/*modif: tres,...*/
{
    printf("[Derivation.c]>> DERIVATION MODIFICATEUR\n");
    DerivationModificateurFlou(pointeur, signe, lambda, cd, fa, att, fichierTF,
fichierMF, fichierCF, fichierQF);

}
else if(ChercheQuantificateurFlou(pointeur->Type.type, fichierQF) == 1)/*quant:
la_plupart,...*/
{
    printf("[Derivation.c]>> DERIVATION QUANTIFICATEUR\n");
    DerivationQuantificateurFlou(pointeur, signe, lambda, cd, fa, att, fichierTF,
fichierMF, fichierCF, fichierQF);
}
else
{
    gestion_erreur("derivation","Derivation","impossible de deriver cette
expression\n",10);
}
}
/* ##### FIN ##
*/

```

K)moteur.c

```

/*****
*
** MOTEUR SQLf POUR iSQLf -> FCTS POUR LES EXECUTABLE GENERE POUR LE
MOTEUR **
*****/

File: 'moteur.c'
Last modifications: 20/04/2001 by MR

*****/

```

```

/* #####
*/

/* Bibliotheque de configuration commune */
#include "/users/stagelsi/isqlf/projet/Sources/SqlGram/Sources/Includes/commun.h"

/* ##### FONCTIONS
***/

float CalculValAttributToLambda(char *terme, char *attribut_ref, float valeur, char *fichier)
{
    /* Tampon de stockage */
    char termeflou[256], attribut[256];
    /* Fichier des termes flous */
    FILE *filetermefl;
    /* Variables definissant la fonction du terme linguistique */
    float A,B,C,D;
    /* Variables servant au calcul */
    float num, den, val=9;
    int status;

    filetermefl = fopen(fichier,"r");
    if (filetermefl == NULL)
        gestion_erreur("gestion_termeflou","CalculValAttributToLambda","ouverture du
fichier de terme flou",50);

    status=fscanf (filetermefl,"%s %s %f %f %f %f",termeflou,attribut,&A,&B,&C,&D);

    while( (status != EOF)
            && ( (strcmp(terme,termeflou, strlen(terme)) != 0)
                || (strcmp(attribut_ref,attribut, strlen(attribut_ref)) != 0)))
    {
        status=fscanf (filetermefl,"%s %s %f %f %f %f",termeflou,attribut,&A,&B,&C,&D);
    }

    if( (status!=EOF)
        && (strcmp(terme,termeflou, strlen(terme))==0 )
        && (strcmp(attribut_ref,attribut, strlen(attribut_ref))==0))

    /* cas ou on n'a pas atteint la fin du fichier (!EOF) et ou on a le bon terme flou avec le bon
attribut */
    {
        /*printf ("terme:%s attribut:%s A:%f B:%f C:%f
D:%f\n",termeflou,attribut,A,B,C,D);*/
        if (B>C)
        {
            C=INT_MAX;

```

```

        D=C;
    }
    if ((valeur<=A)|| (valeur>=D)) val=0.0;
    else if ((valeur<=C)&&(valeur>=B)) val= 1.0;
    else if ((valeur<B)&&(valeur>A))
    {
        num=valeur-A;
        den=B-A;
        val= num/den;
    }
    else if ((valeur<D)&&(valeur>C))
    {
        num=D-valeur;
        den=D-C;
        val= num/den;
    }
    }
    fclose(filetermefl);
    /*printf("valeur= %f\tval= %f\n",valeur, val);*/
    return val;
}
/*float CalculDegreFloat(float g, int comp, float d)*/
float CalculDegreFloat(int comp,float g, float d)
{
    float res=0;
    switch (comp)
    {
        case 1:/*EGAL*/
            res = (g==d) ? 1 :0;
            break;
        case 2:/*DIFFERENT*/
            res = (g!=d) ? 1 :0;
            break;
        case 3:/*INFERIEUR*/
            res = (g<d) ? 1 :0;
            break;
        case 4:/*SUPERIEUR*/
            res = (g>d) ? 1 :0;
            break;
        case 5:/*INFERIEUR-EGAL*/
            res = (g<=d) ? 1 :0;
            break;
        case 6:/*SUPERIEUR-EGAL*/
            res = (g>=d) ? 1 :0;
            break;
        default:
            printf("[CalculDegreFloat] erreur: Type du comparateur n'est pas
considere\n");
            res = -1;
            break;
    }
}

```

```

        return res;
    }
float CalculDegreString(char *g, int comp, char *d)
{
    float res=0;
    /* le type de gauche est char asciz[20] donc il faut comparer avec le longer de droite
    et s'assurer que d n'est pas le debut du g (Ex. Bob et Bobby*/;
    switch (comp)
    {
        case 1:/*EGAL*/
            res = ((strncmp(g,d, strlen(d))==0)&&(g[strlen(d)]==' ')) ? 1 : 0;
            break;
        case 2:/*DIFFERENT*/
            res = ((strncmp(g,d, strlen(d))==0)&&(g[strlen(d)]==' ')) ? 0 : 1;
            break;
        default:
            printf("[CalculDegreString] erreur: On ne considere pas une ordre sur
les chaines\n");
            res = -1;
            break;
    }
    return res;
}
void Ordonner(int n, float **liste)
{
    float *f;
    float *u;    /*liste auxilier*/
    int i=0, j, k;

    u=(float *)malloc (sizeof (float)*n);
    f=(float *)malloc (sizeof (float)*n);

    f=liste[i];
    while (i<n)
    {
        j=0;
        while((j<i)&&(f[i]<=u[j])) j++;    /*on parcourt la liste jusqu'a trouver le place
correspondant*/

        if (j<i)
        {
            /*insérer au milieu de la liste*/

            for (k=0; k<i;k++) u[i-k]=u[i-k-1];    /*deplacer a droite*/
            u[j]=f[i];
        }
        else u[i]=f[i];    /*insérer a la fin de la liste*/
        i++;
    }/*fin du while i<n*/
    *liste=u;
}

```

```

/* -----
    CALCULVALEURVLCF
    Cette fonction calcule la valeur Lambda d'un n-uplet a partir de la valeur
    de l'attribut concerne
    Parametres d'entree:
        terme    : terme flou a rechercher
        attribut  : attribut sur lequel est defini le terme flou
        valeur    : valeur de l'attribut

    Valeur de sortie:
        La fonction retourne la valeur du lambda calcule
    ----- */
float CalculValeurVLCF(char *terme, float valeur, char * fichier)
{
    /* Tampon de stockage */
    char termeflou[256],deriv[256];
    /* Fichier des termes flous */
    FILE *file;
    /* Variables definissant la fonction du terme linguistique */
    float A,B,C,D;
    /* Variables servant au calcul */
    float a,b,l;
    int status;

    file=fopen(fichier,"r");
    if(file == NULL)
        gestion_erreur("gestion_compflou","CalculValeurVLCF","ouverture du fichier de comparateur
        flou",30);

    status=fscanf (file,"%s %f %f %f %f %s",termeflou,&A,&B,&C,&D,deriv);

    while ((status!= EOF) && (strcmp(terme,termeflou)!=0))
    {
        status=fscanf (file,"%s %f %f %f %f %s",termeflou,&A,&B,&C,&D,deriv);
    }

    if((status!= EOF) && (strcmp(terme,termeflou)==0))
    {

        if(A!=0 && B!=0 && C!=0 && D!=0)/* Cas ou la trapezoidale est complete */
        {
            if(valeur >= B && valeur <= C) /*milieu de la trapezoidale*/
                l = 1;
            else
            {
                if(valeur >= A && valeur < B) /*pente gauche de la trapezoidale*/
                {
                    a = 1/(B-A);
                    b=(1-((A+B)/(B-A)))/2;
                    l = ((a*valeur)+b);
                }
            }
        }
    }
}

```

```

    }
    else
    if(valeur > C && valeur <= D) /*pente droite de la trapezoidale*/
    {
    a = - 1/(D-C);
    b = (1+((C+D)/(D-C)))/2;
    l = ((a*valeur)+b);
    }
    else
    l= 0;
    }
    }
    else
    {
    if(A==0 && B==0) /* Cas d'une pente negative */
    {
    if(valeur >= B && valeur <= C)
    l = 1;
    else
    if(valeur > C && valeur <= D)
    {
    a = - 1/(D-C);
    b = (1+((C+D)/(D-C)))/2;
    l = ((a*valeur)+b);
    }
    else
    l = 0;
    }
    else
    {
    if (C==0 && D==0) /* Cas d'une pente positive */
    {
    if(valeur >= B)
    l = 1;
    else
    if(valeur >= A && valeur < B)
    {
    a = 1/(B-A);
    b=(1-((A+B)/(B-A)))/2;
    l = ((a*valeur)+b);
    }
    else
    l=0;
    }
    else
    {
    if(A==0 && D==0)
    {
    if(valeur >= C)
    l = 1;
    else

```

```

        if (valeur < B)
            l = 0;
        else
            if(valeur >= B && valeur < C)
            {
                a = 1/(C-B);
                b=(1-((B+C)/(C-B)))/2;
                l = ((a*valeur)+b);
            }
            }
        }
    }
}
else
    l = -1;
fclose(file);
return l;
}

```

```

float Maximum(float a, float b)
{
    float res;
    res= (a>b) ? a: b;
    return res;
}
float Minimum(float a, float b)
{
    float res;
    res= (a<b) ? a: b;
    return res;
}

```

L)scn1.l

```

%{

/*****
**  MOTEUR SQLf POUR iSQLf -> LEXER **
*****/

Mathias SALLE, Imad EL AOUNI, Didier DAUL, Fabien VANNIER
ENSSAT -- L.L.I.

File: 'scn1.l'
Last modifications: 02/03/2001 by FV

*****/

/* ##### DIRECTIVES

```

```

## */

/* Bibliotheque de configuration commune (moteur SQLf) */
#include "Includes/commun.h"

/* 'y.tab.h' : bibliotheque generee par Yacc a partir de 'sql1.y'
*/
#include "y.tab.h"

/* ##### DECLARATIONS
## */

/*
-----
    DEFINE
-----
- */

/* Macros pour la lisibilite de la grammaire */
#define macroString      yylval.strval = malloc(sizeof(yytext)); strcpy
(yylval.strval, yytext)
#define macroFloat      yylval.floatval = atof(yytext)

/*
-----
    VARIABLES
-----
- */

int    lineno = 1;
char   linebuf[500];

/* ##### ANALYSEUR LEXICAL
## */

/* Rq1 : les commentaires, dans l'analyseur, doivent s'ecrire avec une
tabulation
    avant les symboles classiques (slash et etoile)
    Rq2 : bien respecter la presentation comme ci-dessous pour les '{', '}' et
'|' */

%}

    /* MKS needs the next line to increase the NFA table */

%e 1200

%%

    /* literal keyword tokens */

AVG      |
avg      |
MIN      |
min      |
MAX      |
max      |
SUM      |
sum      |

```

COUNT		
count		{ macroString; return AMMSC; }
ALL		
all		{ return ALL; }
AND		
and		{ return AND; }
ANT		
ant		{ return ANT; }
AM		
am		{ return AM; }
GM		
gm		{ return GM; }
HM		
hm		{ return HM; }
WM		
wm		{ return WM; }
OWAM		
owam		{ return OWAM; }
GLM		
glm		{ return GLM; }
ANY		
any		{ return ANY; }
ARE		
are		{ return ARE; }
BETWEEN		
between		{ return BETWEEN; }
BY		
by		{ return BY; }
CONTAINS		
contains		{ return CONTAINS; }
DISTINCT		
distinct		{ return DISTINCT; }
ESCAPE		
escape		{ return ESCAPE; }
EXISTS		
exists		{ return EXISTS; }
FROM		
from		{ return FROM; }
GROUP		
group		{ return GROUP; }
HAVING		
having		{ return HAVING; }
IN		
in		{ return IN; }
INDICATOR		
indicator		{ return INDICATOR; }
IS		
is		{ return IS; }
LIKE		
like		{ return LIKE; }
MEMBER		
member		{ return MEMBER; }
NOT		
not		{ return NOT; }
NULL		
null		{ return NULLX; }
OF		
of		{ return OF; }
OR		
or		{ return OR; }
OUT		
out		{ return OUT; }
SELECT		
select		{ return SELECT; }

```

SET          |
set          |          { return SET; }
SOME        |
some        |          { return SOME; }
UNION       |
union       |          { return UNION; }
USER        |
user        |          { return USER; }
WHERE       |
where       |          { return WHERE; }

/* names */

[A-Za-z][A-Za-z0-9_]*          { macroString; return NAME; }

/* punctuation */

"="          |
"<>"        |
"!="         |
"<"         |
">"         |
"<="        |
">="        |          { macroString; return COMPARISON; }
[<=>]{1,5}   { macroString; return COMP_FLOUE; }
[-+*/:\(\),,;{}] { return yytext[0]; }

/* numbers */

[0-9]+       |
[0-9]+ "."[0-9]* |
"."[0-9]*    { macroFloat; return INTNUM; }

[0-9]+[eE][+-]?[0-9]+ |
[0-9]+ "."[0-9]*[eE][+-]?[0-9]+ |
"."[0-9]*[eE][+-]?[0-9]+ { macroFloat; return APPROXNUM; }

/* strings \"%[A-Za-z][A-Za-z0-9_]{0,31}\" */
%[A-Za-z][A-Za-z0-9_]{0,31} { macroString; return TERMEFLOUE; }

'^\\n]*'          { macroString; return STRING; }

'([\\n]*'')'      { macroString; return STRING; }

'^\\n]*$          { yyerror("Lexical error"); printf("\tUnterminated
string: %s\\n\\n", yytext); }

\\n.*             { strcpy(linebuf, yytext+1); lineno++; yless(1); }

```

```

[ \t\r]+          ;      /* white space */

"--".*$           ;      /* comment */

.                  { yyerror("Lexical error"); printf("\tProblem (jeton no
encontrado ?!?): %s\n\n", yytext); }

%%

/* ##### FONCTIONS
## */

/*
-----
  FONCTION D'ERREUR
-----
- */

/*//////////////////////////////////////
/////*/
void yyerror (char * s)
/*//////////////////////////////////////
/////*/
{
    printf("\n[scn1.l] >>> Ligne %d : %s dans la ligne \n\t%s\n",
    lineno,s,linebuf);
}
/*//////////////////////////////////////
/////*/

/*
-----
  MAIN -> executable 'sql1'
-----
- */

/*//////////////////////////////////////
/////*/
main (int argc, char **argv)
/*//////////////////////////////////////
/////*/
{
    /* Rq : le 'main' doit etre situe apres l'analyseur pour qu'il puisse
    reconnaitre la variable 'yyin' (de type FILE*) */

    printf("\n[scn1.l] >>> Debut du main\n  ->\n");
    printf("Fichier de la requete floue: %s\n", argv[1]);
    printf("Fichier des Termes flous:%s\n", argv[2]);
    printf("Fichier des Compateurs flous:%s\n", argv[3]);
    printf("Fichier des Modificateurs flous:%s\n", argv[4]);
    printf("Fichier des Quantificateurs flous:%s\n", argv[5]);
    printf("Fichier de requete floue:%s\n", argv[6]);
    printf("Fichier de l'expression derive:%s\n", argv[7]);
    printf("Fichier des resultats:%s\n\n", argv[8]);
    if ((argc > 1) && ((yyin = fopen(argv[1], "r")) == NULL))
    {
        perror(argv[1]);
    }
}

```

```

        exit(-1) ;
    }

    /* Initialisation_fichier dans divers.c */
    Initialisation_fichier(argv[2], argv[3], argv[4], argv[5], argv[6], argv
[7], argv[8]);
    printf ("[scn1.l] >>> Retour de Initialisation_fichier\n\n") ;

    if (!yyparse())
    {
        printf("[scn1.l] >>> \t*** SQL parse made ***\n\n");
fflush(stdout);
        return 1;

    }
    else
    {
        printf("[scn1.l] >>> \t*** SQL parse finished with an error
***\n\n");
fflush(stdout);
        return 0;

    }
}
/*//////////////////////////////////////////////////////////////////////////
////////*/

/* ##### FIN
## */

```