UNIVERSITE DE TUNIS EL MANAR

المدرسة الوطنية للمهندسين بتونس
école nationale d'ingénieurs de Tunis

Unité de recherche en image, traitement de signal et reconnaissance de formes

---

Mémoire en vue de l'obtention du

Diplôme de Mastère

en

Automatique et Traitement du Signal

Intitulé

CONTRIBUTION A L'IMPLÉMENTATION D'UNE BASE DE DONNÉES FLOUE SOUS UN SYSTÈME DE GESTION DE BASE DE DONNÉES RELATIONNEL

Elaboré par :
Mohamed Ali BEN HASSINE

Soutenu le 19 juillet 2005 devant le jury composé de

Président : Monsieur Noureddine ELLOUZE

Membres : Monsieur Kamel HAMROUNI

Madame Amel GRISSA

Année universitaire : 2004-2005

Dédicaces

Je dédie ce travail à :

A ma mère,

A mon père,

A mes soeurs,

A toute ma famille,

A tous ceux que j'aime,

A tous mes enseignants.

Qu'ils trouvent ici l'expression de toute ma reconnaissance.

Remerciements

Je tiens à remercier ***Monsieur Noureddin****e ELLOUZE,* *Professeur à l'ENIT, pour avoir bien voulu accepter de présider ce jury.*

Je tiens également à remercier ***Monsieur Kamel HAMROUNI,*** *Maître assistant à l'ENIT, pour l'honneur qu'il nous a fait d'accepter d'être membre de ce jury.*

Mes remerciements s'adressent aussi à ***Madame Amel GRISSA****, Maître assistant à l'ENIT, pour le sujet proposé, ses directives et sa disponibilité.*

J'aimerai signifier, sincèrement, toute ma reconnaissance à ***Monsieur Habib OUNELLY****, Maître de conférence à la FST, pour ses conseils précieux.*

Je remercie également ***Monsieur José GALINDO,*** *Maître de conférence à l'université de Malaga (Espagne), pour son aide.*

Je ne devrais pas oublier de remercier mes collègues du laboratoire du traitement de signal de l'ENIT pour l'aide qu'ils m'ont apporté.

Table des figures

Liste des tableaux

Table des matières

Introduction générale

Les Systèmes de Gestion de Bases de Données relationnelles (SGBDR) sont devenus, sans conteste, le noyau de tout système informatique. Cependant, la diversification des applications des bases de données a montré les limites des SGBDR notamment sur le plan de modélisation des données imprécises et de l'interrogation flexible. Ainsi, plusieurs extensions du modèle relationnel et du langage SQL ont été proposées pour introduire une certaine incertitude dans la modélisation des données et une certaine flexibilité dans l'interrogation des BD.

En ce qui concerne les données, la convention qui est toujours utilisée dans les bases de données classiques est l'Hypothèse du Monde Fermé (CWA : Closed World Assumption). Son effet principal est de donner implicitement la valeur de vérité FAUX à toute information qui n'est pas dérivable de la base (c-à-d, à toute information où nous ne pouvons pas dire à partir de la base qu'elle est vraie) pourvu que cela ne rentre pas en contradiction avec les informations de la base.
En ce qui concerne l'interrogation classique d'une BDR, le premier problème est que cette interrogation est qualifiée par "interrogation booléenne" dans la mesure où l'utilisateur formule une requête, avec SQL par exemple, qui retourne un résultat ou rien du tout. Cette interrogation pose un problème pour certaines applications. En effet, l'utilisateur doit connaître tous les détails sur le schéma et sur les données de la BD.
Plusieurs travaux ont été menés pour palier ce problème. Le deuxième problème est que l'interrogation booléenne ne permette pas à l'utilisateur ni d'utiliser des termes linguistiques vagues et imprécis dans les critères de qualification des données recherchées ni d'exprimer des préférences entre ces critères, ce qui est souvent une demande légitime des utilisateurs finaux.
Pour illustrer ce problème, considérons un utilisateur qui consulte, via Internet, une BD d'offres de location de biens immobiliers. L'utilisateur souhaite, d'une part trouver un appartement de préférence dans le 16ème ou le 15ème Arrondissement de

Paris ayant une surface "moyenne" et un loyer "modérée" avec une place de parking si possible. Peu importe le mode de formulation de la requête, l'utilisateur souhaite que la qualification soumise à l'évaluation soit la suivante :

```
Select * From Annonce Where ville= 'Paris' and Arrondissement in
(15,16) and Loyer 'modéré' and Surface 'moyenne' and Garage='oui'
```

D'autre part, l'utilisateur souhaite que ses préférences soient considérées selon l'ordre décroissant : Ville, Arrondissement, Loyer, Surface et Garage. Ainsi, les données retournées par le SGBD doivent être ordonnées et présentées à l'utilisateur selon ces préférences. Sans cette flexibilité, l'utilisateur doit affiner ces critères de recherche jusqu'à obtenir éventuellement satisfaction puisqu'il n'a pas des connaissances à priori sur les données qu'il consulte. Ce problème est analogue à celui des moteurs de recherche sur Internet.

Plusieurs travaux ont été proposés dans la littérature pour introduire la flexibilité dans l'interrogation des BD. La majorité de ces travaux ont utilisé le formalisme des ensembles flous et de la logique floue pour modéliser les termes linguistiques tels que ("modéré", "moyen") et pour évaluer des prédicats comportant de tels termes. L'idée essentielle dans ces travaux consiste à étendre le langage SQL et à ajouter une couche supplémentaire d'un SGBD classique pour évaluer les prédicats flous.

En 1995 Bosc [BOS 95] a introduit la première version d'un langage manipulant les requêtes flexibles SQLf. D'autres propositions ont été présentées comme celle de Medina qui a introduit le modèle GEFRED [MED 94] en 1994 puis avec José Galindo [GAL 99a] pour un introduire le langage FSQL similaire à celui de Bosc mais présentant de nouvelles approches tels que les comparateurs, les attributs et les constantes flous. Nous nous intéressons dans ce travail aux travaux de ces derniers, qui ont aboutit à développer une architecture FIRST utilisant le modèle GEFRED en joutant à l'architecture actuelle d'une BDR, un serveur FSQL, une base de méta connaissances Floues (BMCF) et un logiciel FQ qui permet d'interroger une base de données relationnelle floue. Ce logiciel a montré sa puissance dans les consultations flexibles (requêtes floues de type Select). Malheureusement, cet outil ne permet pas à l'utilisateur ni de décrire le schéma de la BDF avec le langage FSQL ni de manipuler sa BDF.

Le but de ce travail est de présenter un outil qui en partant d'un script d'une base de données écrite en FSQL de le traduire en SQL sans obliger le concepteur de connaître la description de la base de méta connaissances Floues BMCF. De plus,

en ayant cet outil, nous pouvons enrichir l'outil FQ pour lui ajouter les fonctions du langage de Définition de Données (LDD) et du langage de manipulation des Données (LMD).

Dans ce mémoire, **le premier chapitre** est consacré à la présentation des concepts de base des ensembles flous et des bases de données, puis à la présentation des principales approches de modélisation des requêtes flexibles. Le **deuxième chapitre** étudie les bases de données floues (BDF), les différents modèles de représentation des données, les différents algèbres relationnelles floues et les langages d'interrogations de ces BDF. Le **troisième chapitre** présente l'architecture FIRST de la BDF introduite par Médina. Le **quatrième chapitre**, étudie un exemple d'implémentation d'une BDF ainsi que les problèmes rencontrés dans cette tâche. Le **cinquième** chapitre présente notre approche pour automatiser le processus de création d'une BDF. Nous terminons par une conclusion générale et les perspectives futures de notre travail.

Chapitre 1

Etat de L'art

1.1 Introduction

Une base de donnée (BD) est un ensemble structuré de données enregistrées sur des supports accessibles par l'ordinateur pour satisfaire simultanément plusieurs utilisateurs de façon sélective en un temps opportun. Elle est modélisée à l'aide d'un modèle de données qui, d'une part permet de définir la structure des données que nous souhaitons la stocker et qui, d'autre part offre un ensemble d'opérateurs destinés à manipuler ces données. Plusieurs modèles ont été définis. Parmi ceux-ci, nous distinguons le modèle relationnel. Ce chapitre présente dans sa première partie, les concepts de base utilisés dans la suite de ce travail et relatifs à la théorie des ensembles flous. Ensuite, nous présentons les bases de données relationnelles (BDR) et leur limites en terme d'intérrogation flexible. En fin, nous étudions les différentes approches de modélisation des requêtes flexibles.

1.2 Théorie des ensembles flous

La naissance du concept flou a été introduite par Lofti Zedeh [ZAD 65] en 1965 afin de représenter mathématiquement l'imprécision relative à certaines classes d'objets.

1.2.1 La logique floue

Une des caractéristiques du raisonnement humain est qu'il est basé sur des données imprécises ou incomplètes [GAB 01]. Ainsi, déterminer si une personne est de petite ou de grande taille est aisé pour n'importe qui d'entre nous, et cela sans nécessairement connaître sa taille.

Cependant, un ordinateur, est basé sur des données exactes. Il doit non seulement connaître la taille exacte de la personne mais également posséder un algorithme qui divise immanquablement une population en deux groupes biens distincts : les grands et les petits. Supposons que la limite soit de $1m65$. Je mesure $1m63$, suis-je vraiment petit ?
L'idée de la logique floue est de transmettre cette richesse du raisonnement humain à un ordinateur. En effet, c'est une logique qui ne se limite plus à deux valeurs de vérité mais à une infinité de valeurs dans l'intervalle [0,1], ce qui permet de traduire des affirmations du type **un peu vrai**, **très faux**, **autant vrai que faux**. En effet, un objet peut appartenir à un ensemble et en même temps à son complément. Ainsi, un individu de 1m63 pourra être à la fois grand et petit.

Définition 1.2.1.
La logique floue est une théorie qui, appliquée dans un calculateur, permet de gouverner une régulation, comme le ferait manuellement un opérateur expert [BEN 02].

Nous présentons dans ce qui suit les objectifs et les domaines d'applications de la logique floue.

Objectifs

L'utilisation de logique floue permet d'écarter et définir les nuances, d'expliciter les choix, d'affiner les modèles grâce à sa flexibilité et sa gradualité, de modéliser le langage naturel en respectant la réalité des tendances. En fait, c'est une théorie mathématique qui modélise les notions vagues du langage naturel, pour palier l'inadéquation de la théorie des ensembles classiques.

Applications

La logique floue est utilisée dans de nombreux domaines et surtout ceux qui sont accessibles au grand public.
Dans le domaine d'automatisme, la logique floue est beaucoup utilisée pour de nombreuses applications (pilotes d'avions, trains automatiques,...).
De nombreuses disciplines informatiques utilisent également la logique floue. Nous pouvons citer notamment l'intelligence artificielle (représentation de la base des connaissances dans les systèmes experts), les bases de données (manipulation des informations incomplètes et imprécises), la recherche d'information et la programmation (programmation floue ou à orientation linguistique) [BEN 02].

Ainsi que d'autre domaines comme la médecine (système expert MYCIN qui permet une aide aux diagnostics), la reconnaissance des formes,...

1.2.2 Les ensembles Flous

Basés sur la logique floue, les ensembles flous ont été introduits pour définir les ensembles qui n'ont pas de limites précises. Il y a donc une transition graduelle et non brutale entre l'appartenance complète et la non-appartenance [ZAD 65].

Définition 1.2.2.
Un ensemble flou F d'univers X (un sous-ensemble flou de X) est défini par une fonction d'appartenance μ_F qui a chaque élément x de X attribue une valeur de l'intervalle [0,1].

$$F = \{(x, \mu_F(x)); x \in X, \mu_F(x) \in [0, 1]\}$$

Cette valeur $(\mu_F(x))$ représente le degré d'appartenance de x à l'ensemble F. Par définition, si $(\mu_F(x) = 0)$ alors x n'appartient pas du tout à F et plus $(\mu_F(x))$ se rapproche de 1, plus la valeur de x appartient à F (si $(\mu_F(x) = 1)$ alors x appartient complètement à F) [LAM 00].

Exemple 1.2.1.
Si X est l'ensemble des villes tunisiennes, nous pouvons définir le sous-ensemble flou A des villes proches de Tunis. Ce sous-ensemble peut être décrit par la fonction d'appartenance suivante : $\mu_A(x) = max(0.1 - d(x,t)/100)$, où d(x,t) représente la distance entre la ville x et Tunis.

$$A = \{Tunis/1, Bizerte/0.35, Sousse/0.1\}$$

Exemple 1.2.2.
Pour évaluer la hauteur (H) d'une personne en utilisant des termes linguistiques, nous devons leurs associer des fonctions de représentation. Nous présentons dans cet exemple, l'évaluation de la hauteur (H) d'une personne selon les deux logiques : classique et floue. Pour cela, nous pouvons utiliser les mots, PETIT et GRAND.

En logique classique

La fonction de représentation du terme PETIT (resp. GRAND) représentée par la figure 1.1 va attribuer à toute personne ayant une hauteur entre 0 et 1m65 (resp. 1m65 et plus) une appartenance totale (degré = 1). Ainsi, une personne qui mesure 1m63 appartient nécessairement au sous-ensemble PETIT (avec un degré 1).

FIG. 1.1 – Représentation de l'âge par des ensembles classiques

En logique floue

La représentation de PETIT et GRAND se fait au moyen de deux fonctions qui calculent l'appartenance de chaque personne à l'ensemble flou PETIT et/ou GRAND.

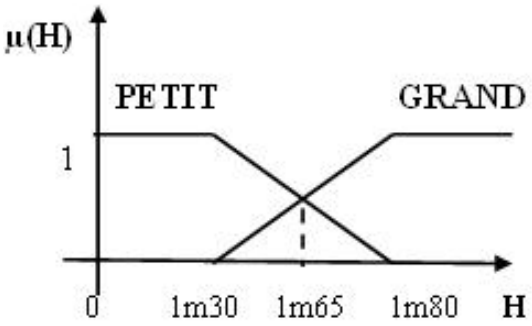

FIG. 1.2 – Représentation de l'âge par des ensembles flous

Ainsi, une personne qui mesure 1m63 appartient en même temps au sous-ensemble flou PETIT et au sous-ensemble flou GRAND avec un degré de 0,7 (pour PETIT) et de 0,3(pour GRAND).

Propriétés

Un ensemble flou F de l'univers X est caractérisé par [BEZ 93] :

- Son noyau, noté noy(F), qui représente l'ensemble des éléments de X pour lesquels la fonction d'appartenance $\mu_F(x)$ vaut 1 : $noy(F) = \{x \in X / \mu_F(x) = 1\}$.
- Son support, noté supp(F), qui représente l'ensemble des éléments de X appartenant, même très peu, à F, c-à-d, ayant $\mu_F(x)$ qui n'est pas nulle :

$$supp(F) = \{x \in X / \mu_F(x) \neq 0\}.$$

– Sa hauteur, notée h(F), qui représente la plus grande valeur prise par sa fonction d'appartenance : $h(F) = \sup_{x \in X} \mu_F(x)$
– Sa α-coupe qui représente l'ensemble contenant les éléments ayant un degré d'appartenance supérieur ou égal à α : α-coupe(F) $= \{x \in X / \mu_F(x) \geq \alpha\}$
– Un ensemble flou est dit normalisé s'il existe au moins un élément de complète appartenance ($\exists$x tel que $\mu_F(x) = 1$).

Pour une fonction d'appartenance $\mu_F(x)$ de forme trapézoïdale, le support, la hauteur et le noyau sont définis comme suit :

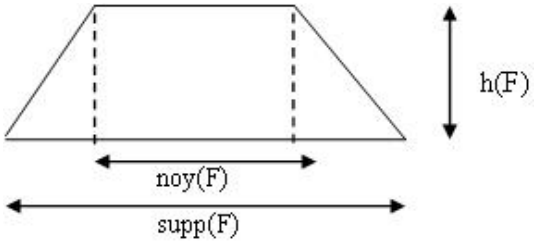

FIG. 1.3 – Caractéristiques d'un ensemble flou

Nous pouvons remarquer qu'un ensemble ordinaire n'est qu'un cas particulier d'ensemble flou dont la fonction d'appartenance n'attribut que les degrés 1 ou 0, c'est un ensemble avec une hauteur de 1 et un support égale au noyau (Figure 1.4).

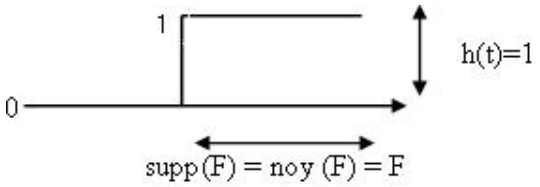

FIG. 1.4 – Correspondance entre les ensembles flous et les ensembles classiques

L'utilité des sous-ensembles flous repose sur l'identification d'un univers de discours convenable et la détermination des fonctions d'appartenance appropriées.

Exemple 1.2.3.
Soit la classe des grandes personnes, présentée dans la figure 1.2. Nous pouvons dire qu'une personne mesurant 1m30 n'appartient pas à cette classe et par suite sa fonction d'appartenance aux grandes personnes sera $\mu_F(1.30) = 0$, *par contre une autre qui mesure 1m70 appartient à cette classe avec une fonction d'appartenance* $\mu_F(1.70) \in]0, 1[$ *et plus sa taille se rapproche de 1m80 plus l'appartenance à cette classe serait forte* ($\mu_F(1.80) = 1$).

Opérations sur les ensembles flous

Nous commençons par définir les notions de normes et conormes triangulaires qui seront utilisées dans les opérations d'intersection et d'union.

Définition 1.2.3.
Une norme triangulaire (ou t-norme) T est une opération binaire sur l'intervalle [0,1]. Cette opération est associative, commutative, monotone et telle que T(a,1)=a.

Définition 1.2.4.
Une conorme triangulaire (t-conorme) $\perp$ est une opération binaire sur l'intervalle [0,1]. Cette opération est associative, commutative, monotone et telle que $\perp$(a,0)=a.

Définition 1.2.5.
Puisque le couple norme/conorme triangulaire, min/max est le plus utilisé dans la littérature, nous l'avons utilisé pour représenter les différentes opérations sur les ensembles flous.
Soient A et B deux ensembles flous définis sur l'univers X, nous avons les définitions suivantes [BEZ 93] [LAM 00] :
- *La complémentation de l'ensemble flou A :* $\forall x \in X, \mu_{\overline{A}}(x) = 1 - \mu_A(x)$
- *L'intersection de A et B :* $\forall x \in X, \mu_{A \cap B}(x) = min(\mu_A(x), \mu_B(x))$
- *L'union de A et B :* $\forall x \in X, \mu_{A \cup B}(x) = max(\mu_A(x), \mu_B(x))$
- *La différence de A moins B :* $\forall x \in X, \mu_{A-B}(x) = min(\mu_A(x), 1 - \mu_B(x))$
- *L'inclusion de A dans B :* $\mu_{A \subseteq B}(x) = min_{x \in X} \mu_A(x) \rightarrow \mu_B(x)$ *où* $\rightarrow$ *désigne une implication floue.* [1]

1.2.3 Les relations floues

Le concept de relation floue généralise celui de relation classique. Il met en évidence des liaisons imprécises ou graduelles entre les éléments d'un ou plusieurs ensembles [LAM 00].

Définition 1.2.6.
Une relation floue R sur les univers de références X1, X2,...,Xn est définie comme un ensemble flou du produit cartésien $X1 \times X2 \times ... \times Xn$ *ayant la fonction d'appartenance* μ_R.

[1] Les implications floues les plus utilisées sont celles de Kleene-Dienes (a$\rightarrow$b= max (1-a, b)), de Godel (a$\rightarrow$b=1 si $a \leq b$, b sinon), de Goguen (a$\rightarrow$b=1 si $a \leq b$, b/a sinon) et de Lukasiewiez (a$\rightarrow$b=1 si $a \leq b$, 1-a+b sinon).

Exemple 1.2.4.
La relation floue R : "approximativement égal à" peut être définie sur $\Re \times \Re$ *par la fonction :* $\mu_R(x,y) = \frac{1}{1+(x-y)^2}$

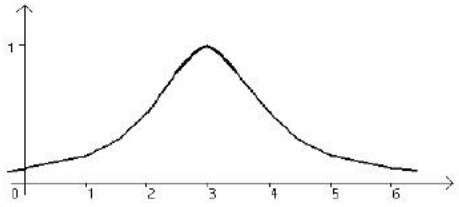

FIG. 1.5 – x approximativement égal à 3

La différence entre une relation floue et une relation classique (exacte) est que pour la première, toute valeur d'appartenance dans l'intervalle [0,1] est permise alors que pour la seconde, seules les valeurs 0 et 1 sont permises [BEZ 93].

Définition 1.2.7.
Une relation floue R sur X est dite :
- *symétrique si :* $\forall (x,y) \in X \times X, \mu_R(x,y) = \mu_R(y,x)$,
- *antisymétrique si :* $\forall (x,y) \in X \times X, (\mu_R(x,y) > 0$ *et* $\mu_R(y,x) > 0) \Longrightarrow x = y$,
- *refléxive si :* $\forall x \in X, \mu_R(x,x) = 1$,
- *transitive si :* $R \circ R \subseteq R$ *ie :* $\forall (x,y) \in X \times X, \mu_{R \circ R}(x,y) \leq \mu_R(x,y)$.

Définition 1.2.8.
Une relation de similarité est une relation réflexive, transitive et symétrique.

Une relation de similarité permet de modéliser la ressemblance et la proximité.

Définition 1.2.9.
Une relation d'ordre floue est une relation réflexive, transitive et antisymétrique.

Une relation d'ordre exprime la notion de préférence et d'antériorité.

1.2.4 Variable linguistique

Étant donnée que les termes sont moins précis que les nombres, le concept de variable linguistique se révèle ainsi approprié pour la description des connaissances imprécises et vagues.

Définition 1.2.10.
une variable linguistique est une fonction dont le domaine est un ensemble classique, et les valeurs sont des ensembles flous [LAM 00].
Chaque variable linguistique représentée un quintuple $(x, T(x), U, G, M)$ *où*
– *x est le nom de la variable,*
– *T(x) est l'ensemble des valeurs linguistiques que peut prendre x,*
– *U est l'univers de discours associé avec la valeur de base,*
– *G est la règle syntaxique pour générer les valeurs linguistiques de x,*
– *M est la règle sémantique pour associer un sens à chaque valeur linguistique.*

Exemple 1.2.5.
La variable linguistique `x = taille moyenne` *peut être définie avec un ensemble des termes : T={petite,moyenne,grande} qui forment son univers de discours* $U = [30cm, 220cm]$. *La variable de base est* `taille`. *Le terme* `petite` *représente une valeur linguistique. Nous pouvons l'interpréter, par exemple comme "les tailles plus petites que 150cm"*

Les fonctions d'appartenance trapézoïdales sont les plus utilisées dans la littérature pour exprimer l'imprécision des termes linguistiques. Cependant, il est difficile d'utiliser la même fonction d'appartenance pour modéliser un terme linguistique donné dans toutes les circonstances. Ainsi, il n'existe pas de concepts linguistiques ayant des distributions universelles. Par exemple, le terme "moyenne" qui caractérise la variable linguistique distance peut être décrit par la figure 1.6.

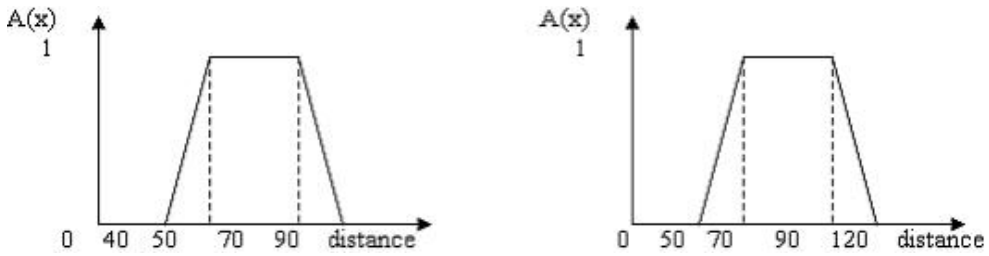

FIG. 1.6 – Deux modélisations de "distance moyenne"

1.2.5 Les modificateurs linguistiques

Toutes les descriptions imprécises d'une même variable linguistique V ne peuvent pas être décrites par la liste T_v. C'est pourquoi, nous préférons recourir à des caractéristiques intermédiaires à l'aide des modificateurs tels que "très", "environ"

qui permettent de moduler une description en l'atténuant ou en la renforçant. Ces modificateurs sont eux aussi définis par des sous-ensembles flous [HAC 02].

Exemple 1.2.6.
La fonction d'appartenance du sous-ensemble flou "très petit" peut être définie par : $\mu_{très\_petit}(x) = (\mu_{petit}(x))^2$

1.2.6 Fuzzification et Défuzzification

Définition 1.2.11.
La fuzzification est le processus de conversion d'une donnée précise en une donnée floue.

Elle consiste à définir les fonctions d'appartenance de toutes les variables précises (d'entrée) et à déterminer le degré d'appartenance de chacune d'elles à l'ensemble flou en question [BEZ 93].

Définition 1.2.12.
D'une façon similaire, la défuzzification se définie comme le processus de passer d'une donnée floue (variable linguistique) en une donnée exacte.

Plusieurs méthodes utilisées dans la défuzzification, la plus souvent utilisée est le calcul du centre de gravité.

La fuzzification et la défuzzification sont utilisées également dans les BD [HAC 02]. Dans ce cadre, elles consistent à assouplir les critères de recherche figurant dans les requêtes au moyen de sous-ensembles flous et transformer ainsi la requête originale en une requête flexible (et inversement). Ceci donne à l'utilisateur la possibilité d'avoir une image plus réaliste des réponses à sa requête. Dans le cas, où aucun élément ne satisfait complètement les conditions de la requête, l'utilisateur dispose des réponses les plus proches de la réponse désirée.

1.3 Les bases de données

1.3.1 Définitions et concepts de base

Une base de données (BD) représente l'ensemble (cohérent, intégré, partagé) des informations nécessaires au fonctionnement d'une entreprise mémorisée sur un

support permanent, dont la gestion est assurée par un logiciel appelé système de gestion de bases de données [STE 03].

Un système de gestion de bases de données (SGBD) est une collection de logiciels permettant de créer, de gérer et d'interroger efficacement une BD indépendamment du domaine d'application. Il permet d'interagir avec une BD pour satisfaire simultanément les besoins de plusieurs utilisateurs tout en assurant la sécurité, l'intégrité et la confidentialité indispensables lorsqu'un grand nombre d'utilisateurs variés veulent interagir simultanément avec les données de la base [MAR 01].

1.3.2 Modèles des bases de données

La notion de modèle de données est essentielle et c'est elle qui souvent motive le choix de l'utilisation d'une BD. En effet, la résolution d'un problème par un automate nécessite de représenter l'information sur un domaine traité appelé parfois mini-monde ou univers du discours sous une forme digitale qui soit interprétable et manipulable par un ordinateur. Un modèle peut se définir comme une représentation abstraite de l'information et éventuellement des opérateurs de manipulation de l'information. Plusieurs modèles ont été définis, nous citons :

Le modèle hiérarchique a été introduit par IMS d'IBM en 1964. Dans ce modèle, les données sont représentées par une structure en arbre où les données inférieures dépendent des données supérieures. Cette structure est conçue avec des pointeurs et détermine le chemin d'accès aux données. Ce modèle ne permet de représenter qu'un seul type d'association : père-fils ; d'où sa limitation à la représentation d'univers hiérarchique [GRI 02].

Le modèle réseau a été introduit par IDS2 en 1964. Dans ce modèle, la structure des données peut être visualisée sous la forme d'un graphe quelconque. Ce modèle permet la représentation de tout type d'association mais impose pour accèder à un objet de naviguer le long de la base via une succession de pointeurs [GRI 02].

Le modèle Relationnel a été formalisé par CODD en 1970. Dans ce modèle, les données sont stockées dans des tables (relations), sans préjuger de la façon dont les informations sont stockées dans la machine. Un ensemble de données sera donc modélisé par un ensemble de tables. Le succès du modèle relationnel auprès des

chercheurs, concepteurs et utilisateurs est dû à la puissance et à la simplicité de ses concepts. De plus, et contrairement à certains autres modèles, il repose sur des bases théoriques solides, notamment la logique mathématique (théorie des prédicats d'ordre 1) et la théorie des ensembles [CAM 01].

Le modèle Entité-Relation (E/R) appelé aussi Entité-Association (E/A) est un modèle de données de type conceptuel introduit par CHEN en 1976. Il est actuellement utilisé par plusieurs méthodes et outils d'aide à la conception des BD (MERISE, IDA, Yourdon, etc.). L'objectif d'un modèle Entité-Association est de décrire, au travers d'un diagramme et de façon logique, les données que nous souhaitons intégrer dans la base, ainsi que les relations qui les lient. Le modèle EA se base sur les deux concepts : entité et association. Une entité est une représentation d'un objet du monde réel (concret ou abstrait). Une association est une représentation d'un lien entre plusieurs entités, un lien où chaque entité liée joue un rôle déterminé [STE 03].

Le modèle Objet a été introduit par Atkinson et al. en 1989. Dans ce modèle, les données sont représentées sous forme d'objets au sens donné par les langages orientés objet. Ce modèle regroupe les concepts essentiels pour modéliser de manière progressive des objets complexes encapsulés par des opérations de manipulation associées. Il vise à permettre la réutilisation de structures et d'opérations pour construire des entités plus complexes [GRI 02].

Le modèle Objet-Relationnel a été introduit en 1992. Il est une tentative de réunion des concepts présents dans les modèles relationnels et objets. Cette réunion est réalisée en étendant le modèle relationnel pour lui conférer un certain nombre de qualités reconnues du modèle objet. La totalité des fonctions d'un SGBDR classique est préservée et les concepts qui font le succès de l'approche objet ainsi que de nouveaux types de données y sont intégrés [DAE 01].

1.4 Les Bases de Données Relationnelles

Nous détaillons dans cette partie les concepts de base des BDR.

1.4.1 Le modèle Relationnel

Le modèle relationnel est basé sur le concept mathématique de relation représenté logiquement comme une table à 2 dimensions [STE 03].

- **Relation** : c'est une table avec des colonnes et des lignes.
- **Attribut** : c'est un nom attribué à une colonne d'une relation.
- **Domaine** : c'est un ensemble de valeurs que peut prendre un ou plusieurs attributs.
- **Tuple** : c'est une ligne d'une relation.
- **Degré** : le d^0 d'une relation est le nombre de ses attributs.
- **Schéma de relation** : Le schéma d'une relation R est défini par le nom de la relation et la liste des attributs avec pour chaque attribut son domaine : $R = (A1 : D1, ..., An : Dn)$ ou, plus simplement $R(A1, A2, ..., An)$. Le schéma de relation est stable dans le temps.
- **Instance** : une instance d'une relation (ou relation) r de $R(A1, A2, ..., An)$ est un sous ensemble de produit cartésien $D1 \times D2 \times ... \times Dn$ ou $r \subseteq D1 \times D2 \times ... \times Dn$. On la note par r(R) [Rig 03]. La relation ou l'instance varie dans le temps.
- **Schéma de BD** : Le schéma d'une BD est l'ensemble des schémas de ses relations.

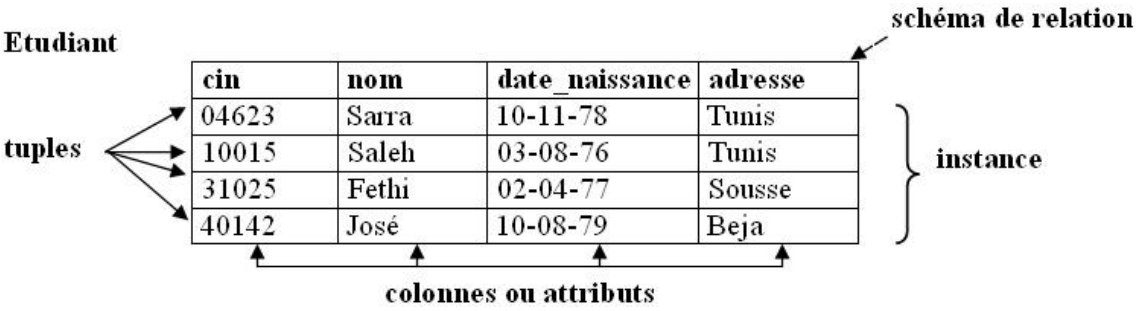

FIG. 1.7 – Concepts de bases du modèle relationnel

- **Cardinalité** : La cardinalité d'une relation est le nombre de tuples de la relation.
- **BD relationnelle** : collection de relations normalisées.
- **Identifiant d'une relation** : L'identifiant (aussi appelé clé) d'une relation est un ensemble minimum d'attributs de la relation, tel qu'il n'existe pas deux tuples ayant même valeur pour cet identifiant.
- **Les contraintes d'intégrité** : Le modèle relationnel comporte un certain nombre de règles permettant de garantir la cohérence des données. Ces règles sont divisées en deux catégories : les contraintes structurelles et les contraintes d'entreprise.

1.4.2 Les langages de manipulation de données relationnelles

Le modèle relationnel a été à l'origine proposé avec deux langages de manipulation de données (LMD), l'algèbre relationnelle et le calcul des tuples, équivalents en puissance et qui ont fixé l'ensemble des fonctions que tout LMD relationnel doit offrir [STE 03].
L'algèbre relationnelle est un ensemble d'opérateurs qui, à partir d'une ou deux relations existantes, créent en résultat une nouvelle relation temporaire. La relation résultat a exactement les mêmes caractéristiques qu'une relation de la BD et peut donc être manipulée de nouveau par les opérateurs de l'algèbre [codd 70]. Formellement l'algèbre relationnelle comprend :
- cinq opérateurs de base : sélection, projection, union, différence et produit,
- un opérateur syntaxique, renommer, qui ne fait que modifier le schéma et pas les tuples.

Le calcul de tuples (ou calcul relationnel) est constitué des langages issus du calcul des prédicats de la logique du premier ordre. Deux adaptations de ce calcul au modèle relationnel ont été proposées, qui toutes deux ont conduit à des langages utilisateurs :
- le calcul des tuples, qui a donné naissance au LMD QUEL du SGBD relationnel Ingres,
- le calcul des domaines qui a donné naissance à des LMD de type graphique comme QBE, proposé par IBM.

L'algèbre relationnelle et le calcul relationnel sont équivalents dans la mesure où ils ont la même puissance d'expression c-à-d toute expression de l'algèbre relationnelle possède une expression du calcul relationnel qui lui est équivalente.
SQL est le LMD relationnel le plus répandu du fait qu'il est la seule norme existante pour les LMD relationnels.

Le langage SQL

Le langage "Structured Query Language "(SQL) a été introduit dans les années 70 par IBM [GAM 03]. C'est un langage déclaratif[2], non procédural dans sa formulation, masquant assez bien le caractère algébrique des expressions. Il a une syntaxe

[2]SQL est déclaratif parce qu'il n'exige pas de l'utilisateur une connaissance de l'implémentation des opérateurs et des tables ou des chemins d'accès pour une formulation de la requête qui doit exprimer normalement les tuples qui doivent être inclus dans la réponse ainsi que des conditions à vérifier par ces tuples

relativement simple, mais affiche une facette de langage naturel, soit l'anglais élémentaire. Le langage SQL a fait l'objet de plusieurs efforts de normalisation qui ont débuté de façon sérieuse en 1989 et qui poursuivent toujours. Il y a plusieurs normes SQL, chacune enrichissant la version précédente : SQL-89, SQL-92, SQL-93. Il présente trois facettes fortement intégrées peu importe le SGBD relationnel :

- LID (Langage d'Interrogation des Données), pour la définition des requêtes d'interrogation sur les données contenues dans la base.
- LDD (Langage de Définition de Données), pour la définition des tables, des contraintes diverses et des vues relationnelles stockées dans le dictionnaire du SGBD.
- LMD (Langage de Manipulation de Données), pour la manipulation des tables et plus précisément les manipulations des tuples de relations.
- LCD (Langage de Control de Données) pour gérer la définition physique des accès (index), la spécification des fichiers physiques et la validation des opérations exécutées dans un contexte multiposte.

Format de base d'une requête

SELECT Liste des noms d'attributs du résultat
FROM Nom d'une relation (ou de plusieurs relations)
[WHERE Condition logique qui définit les tuples du résultat]

1.5 Exemple de Base de Données Relationnelle

Nous présentons dans cette partie un exemple de description de BDR, sa modélisation à l'aide du modèle E/R et le modèle Relationnel correspondant.

1.5.1 Description du monde réel

Soit la BD ENTREPRISE qui a pour but de gérer les employés, les départements et les projets d'une entreprise. Elle est décrite comme suit :
Un département possède un nom, un numéro identifiant et un directeur dont la date d'entrée est mémorisée. Un département possède un local. Il peut contrôler plusieurs projets mais un projet est contrôlé par un seul département. Chaque projet est caractérisé par un nom, un numéro et un budget. Un employé est décrit par un nom, une matricule unique, une adresse, un salaire, un sexe et une date de naissance. Un

employé possède un seul supérieur hiérarchique immédiat. Il possède aussi un niveau d'études et un rendement. Un employé est affecté à un seul département, mais il peut travailler sur plusieurs projets non nécessairement rattachés à son département. Le nombre d'heures par semaine passées par un employé sur chaque projet est mémorisé. Un poste est décrit par un code unique, un nom (profession) et exige un niveau d'études et une expérience. Chaque employé possède des personnes qui sont à sa charge (conjoint, enfants,etc.) ou dépendants. Chaque dépendant est décrit par : un prénom, un sexe, une date de naissance et un lien de parenté avec l'employé.

1.5.2 Modèle Entité Association

Suite à cette description, nous pouvons déduire le modèle E/R présenté dans la figure 1.8.

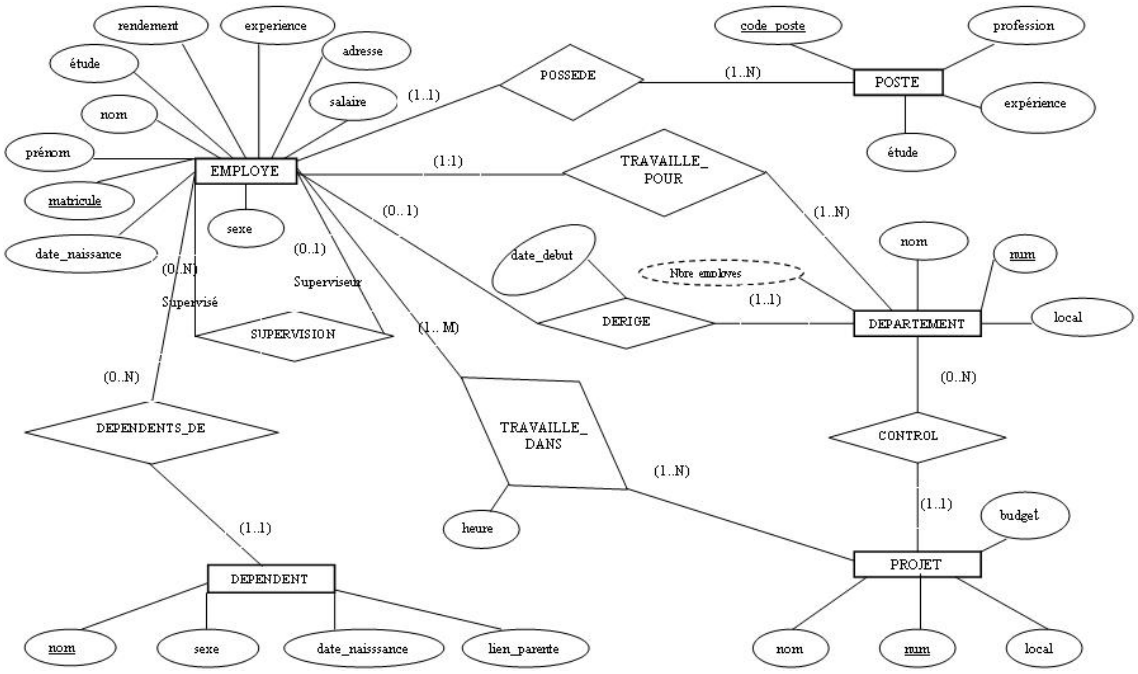

FIG. 1.8 – Modèle Entité/Association

1.5.3 Modèle relationnel

En appliquant un ensemble de règles de passage, nous pouvons déduire le modèle relationnel suivant :

`DÉPARTEMENT` (num_d, nom, matricule_directeur, date_début, local)

`POSTE` (code_poste, profession, études, experience)
`EMPLOYÉ` (matricule, *num_d* [3], *code_poste*, nom, prénom, sexe, adresse, date_naissance, salaire, études, rendement, expérience, ssoc_superviseur)
`PROJECT` (num_p, *num_d*, nom, budget)
`DEPENDENT` (*matricule*, prénom, sexe, date_naissance, lien_parenté)
`TRAVAILLE_DANS` (*matricule*, *num_p*, nbre_heure)

1.5.4 Notion de requête dans une BDR

Une requête de consultation (interrogation) d'une BDR se fait par la requête SELECT de SQL qui consiste à extraire les informations suivant une syntaxe bien déterminée. De ce fait, une condition est évaluée, si elle est valide l'information est extraite, sinon une valeur nulle est retournée (NULL).

Exemple 1.5.1.
Considérons les deux requêtes suivantes :
`Requête 1` *: "Trouver les employé ayant un salaire supérieur à 450 dinars habitants à TUNIS"*
Cette requête est modélisée en langage SQL comme suit :

*select * from EMPLOYÉ*
where salaire > 450 and adresse = 'TUNIS'

`Requête 2` *: "Trouver les départements ayant un nombre d'employé supérieur à 200 et qui ont fait des projets de budget inférieur à 5000 dinars"*
De même cette dernière suit la syntaxe suivante :

select num_d from DÉPARTEMENT, PROJET
where DÉPARTEMENT.num_d = PROJET.num_d
and budget < 5000

1.6 Limites des Bases de Données Relationnelles

Avec l'évolution de l'informatique, les BDR ont monté plusieurs limites : nous présentons dans cette partie les limites qui se rapportent à notre sujet.

[3] Les attributs écrits en italique et qui sont soulignés, représentent des clés étrangères : *code_poste* référence un poste, *num_d* référence un département, *num_p* référence un projet et *matricule* référence un employé

1.6.1 Types de données

Dans une BDR, les données doivent être en 3ième forme normale. Dans ce sens, une telle base ne permet de modéliser que des attributs simples.
Or, généralement, l'information de l'être humain est imprécise, nous avons couramment recours à dire que l'information est vraie à 50% ou à 70%.
Avec le concept des BDR, nous ne pouvons pas modéliser un texte pareil, alors que cette modélisation a été longuement utilisé dans le domaine de l'intélligence artificielle et de systèmes experts [BOS 02]. La question qui pose à ce niveau est : pourquoi ne pas offrir à l'utilisateur la possibilité de modéliser ses informations imprécises ?
Plusieurs travaux ont été proposés dans la littérature pour introduire la flexibilité des BD. La majorité de ces travaux ont utilisé le formalisme des ensembles flous pour modéliser les termes linguistiques et pour évaluer des prédicats comportant de tels termes. L'idée essentielle dans ces travaux consiste à étendre SQL et à ajouter une couche supplémentaire au-dessus d'un SGBDR pour évaluer ces prédicats.
Nous présentons dans ce qui suit, un exemple de ces requêtes.

1.6.2 Requêtes

L'interrogation classique des BDR utilise une requête pour trouver des éléments satisfaisant une condition booléenne, que nous évaluons comme classiques, comme dans l'exemple 1.4.1.
Néanmoins, dans certaines applications :

- l'utilisateur peut retrouver des éléments satisfaisant une condition booléenne et dans d'autres non. D'autre part, il peut trouver une difficulté à caractériser d'une manière précise et claire l'information qu'il recherche, c-à-d, il n'exprime pas directement son besoin avec une condition booléenne, mais au contraire graduellement avec des termes imprécis.

  Exemple 1.6.1.

  pour illustrer ces problèmes, nous proposons les requêtes suivantes :

  `Requête 1` *: "Trouver les employés* ***bien payés*** *de Tunis et qui sont les plus* ***jeunes****"*

  `Requête 2` *: "Trouver les numéros des départements à* ***gros*** *budget ayant au moins un employé de salaire* ***élevé*** *et d'âge* ***à peu près*** *40 ans"*

  `Requête 3` *: "Trouver les employés ayant un rendement* ***régulier*** *travaillant dans un département ayant un nombre d'employés* ***élevé****"*

`Requête 4` *: "Trouver les employés* ***jeunes*** *ayant un salaire* ***près de*** *750 dinars"*

– il peut également vouloir exprimer des préférences au niveau des critères de recherche avec éventuellement des différents degrés d'importance entre ces critères. Voici des exemples de ces requêtes.

Exemple 1.6.2.

Soit une personne qui désire rechercher, dans une BD d'annonces, un restaurant proche du centre ville avec des repas à prix abordable. Afin d'exprimer de telles préférences, cette personne peut formuler une requête "flexible" comportant les termes "proche" et "abordable". Il peut également exprimer le fait que le critère prix est plus important que celui de la distance.

`Requête 1` *: "Trouver la liste des restaurants* ***chinois proches*** *de son domicile pour y prendre un repas de prix* ***abordable****"*

Un autre exemple est celui d'un utilisateur qui consulte, via Internet, une BD d'offres de location de biens immobiliers. L'utilisateur souhaite trouver un appartement de préférence dans le **16**ème ou le **15**ème Arrondissement de **Tunis** ayant une surface "moyenne" et un loyer "modéré" avec une place de parking si possible. D'autre part, l'utilisateur souhaite que ses préférences soient considérées selon l'ordre décroissant : Ville, Arrondissement, Loyer, Surface et parking.

`Requête 2` *: "Trouver la liste des appartements dans le* ***15ième*** *ou le* ***16ième*** *arrondissement de* ***Tunis*** *avec un loyer* ***modéré*** *et une surface* ***moyenne*** *contenant un parking* ***si possible****"*

1.6.3 Discussion

Les requêtes citées ci-dessus ne peuvent pas être traitées par un SGBDR à cause de l'existence d'attributs comprenant des expressions vagues, incertaines et imprécises (sous forme de variables linguistiques). Si nous voulons exprimer ces requêtes avec une écriture mathématique, nous pouvons sortir de nos besoins. Ainsi, pour traduire la première requête en langage relationnelle, nous pouvons écrire :

select * from EMPLOYÉ where âge[4]<24 and âge>16
and salaire > 400 and adresse = "TUNIS"

Mais cette conversion ne traduit pas la réalité, car l'attribut "jeune" ne peut pas être exprimé par un simple intervalle mathématique [16, 24] à cause, d'une part, de

[4]L'attribut âge se calcule comme suit :
SELECT (TO_CHAR(SYSDATE,'yyyy') - TO_CHAR(date_naissance,'yyyy'))into âge FROM EMPLOYÉ;

la non uniformité d'une connaissance qui diffère d'une personne à une autre (une personne choisit l'intervalle [16, 24] et une autre l'intervalle [15, 28]) et d'autre part de la sensibilité des bornes choisies pour la définition d'un tel intervalle. Ainsi, pour un employé ayant 24 ans et un mois et ayant un salaire de 1800 dt sera exclu, alors qu'un autre âgé de 16 ans et ayant 400 dt sera accepté.

La requête 1 de l'exemple 1.6.2, contient à son tour les termes "proche" ou "abordable" qui devront être traduits en conditions booléennes pour être exécutés par les SGBD usuels, mais deux types d'attitudes sont envisageables :

- Donner à ces termes le sens correspondant à des valeurs idéales : chinois ; <1 Km (pour proche) ; l'intervalle [10dt, 15dt] (pour abordable) avec le risque de n'obtenir aucune réponse.
- Les donner le sens de tolérable : "chinois", "vietnamien", "japonais" ; <3Km ; l'intervalle [9dt, 25 dt] avec le risque d'obtenir une pléthore de réponses.

Dans les deux cas, les réponses retournées par le système seront équivalentes dans le sens où elles auront satisfait la condition imposée et il revient à l'usager de les ordonner [BOS 98]. Cet exemple illustre alors les contraintes imposées par ce cadre :

- Nécessité d'une transformation du besoin initial en une requête booléenne avec une sensibilité forte des bornes choisies (un restaurant situé à 1.1 Km et offrant un repas de 12dt seras écarté alors qu'un autre situé à 1Km et offrant un repas de 15 dt est conservée).
- Pauvreté du service rendu : pas d'ordonnancement des réponses avec le risque d'avoir des réponses vides ou pléthoriques et la combinaison de conditions est limitée à la conjonction et la disjonction.

Par ailleurs, dans cette requête, l'utilisateur pourrait souhaiter exprimer le fait que le critère de prix est plus important que celui de distance, ce qui conduit à introduire un mécanisme de gestion des importances relatives aux critères.

Les limites évoquées précédemment ont été mises en évidence dans différents travaux pour étendre les possibilités des systèmes fondés sur l'interrogation purement booléenne afin d'introduire la flexibilité dans les requêtes.

1.7 Requêtes flexibles

La plupart des travaux menés dans le domaine de l'interrogation des BD font l'hypothèse qu'une requête a pour objet la recherche des éléments satisfaisant une condition booléenne. S'il est vrai que beaucoup d'applications relèvent de ce besoin (notam-

ment en gestion), il en est d'autres pour lesquels cette approche n'est pas appropriée. En effet des problèmes apparaissent lorsque l'utilisateur ne s'exprime pas directement comme une condition booléenne (en tout ou rien), mais est au contraire graduel, en raison de l'existence de préférences au niveau de ces critères de recherche [BOS 98]. Ces problèmes peuvent être résolus par une interrogation souple appelée encore flexible.

Définition 1.7.1.
Une requête flexible est une requête qui comporte des descriptions imprécises et/ou des termes vagues qui peuvent comporter des préférences.

Les domaines d'applications de l'interrogation flexible sont très variés. L'interrogation flexible s'avère en effet pertinente dans de nombreux contextes : systèmes documentaires, pages jaunes, petites annonces, recherche d'images scéniques, recherche d'informations sur le web, interrogation de bases d'informations multimédia, intérrogation de BD possibiliste et probabiliste [BOS 03].

1.8 Approches de modélisation des requêtes flexibles

Nous pouvons regrouper les travaux visant à exprimer des préférences dans les requêtes flexibles en quatre catégories [BOS 98] :

1. Utilisation d'un critère complémentaire de classement.
2. Utilisation des distances associées aux domaines des attributs afin d'étendre l'égalité stricte.
3. Expression des préférences avec des termes linguistiques.
4. Modélisation de l'imprécision par la théorie de sous ensembles flous.

Nous détaillons dans la suite ces différentes catégories.

1.8.1 Critère complémentaire de classement

Dans cette approche, une requête se compose d'une condition C pour sélectionner des n-uplets et d'une préférence P pour les classer. DEDUCE2 et PREFERENCES sont parmis les systèmes qui ont utilisé cette approche.

Le système DEDUCE2

Dans ce système [CHA 82], une question se compose d'une condition booléenne (C1) et d'une condition imprécise (C2) exprimant les préférences de l'utilisateur. C2 peut contenir des termes linguistiques, comme "jeune", "bien-payé", connectés par ET/OU. Une condition imprécise élémentaire est modélisée par une fonction monotone d'un attribut, par exemple, "salaire proche de 500" est modélisée par une fonction décroissante (f(salaire) = |salaire - 500|). Un rang est attribué à chaque n-uplet de la BD par un tri sur la fonction considérée. Si C2 est constituée de deux termes T1 et T2, leur combinaison sera exprimée par :

$$r_{T1 \text{ AND } T2} = max(r_1, r_2)$$

$$r_{T1 \text{ OR } T2} = min(r_1, r_2)$$

r_1 et r_2 sont les rangs affectés à chaque n-uplet en considérant respectivement les termes T1 et T2.

Exemple 1.8.1.
soit la relation Produit (num, désignation, quantité-stock, prix-unitaire, ville-fournis) représentée dans la table 1.1 et la requête : "trouver les produits dont la ville du fournisseur est Tunis avec une préférence pour ceux qui coûtent environ 2500 et dont la quantité en stock est au voisinage de 900". Le terme "environ" est décrit par la fonction f(prix) = |prix - 2500|. Le terme "aux voisinage" est décrit par la fonction f(qtité) = |qtité - 900|. L'expression de cette requête sur la relation produit permet d'obtenir le résultat de la table 1.2.

num	désignation	quantité-stock	prix-unitaire	ville-fournis
12	Savon	800	1500	Tunis
14	Parfum	850	2500	Sfax
20	Shampoing	900	3000	Tunis
25	Dentifrice	800	1000	Tunis

TAB. 1.1 – Extension de la relation produit

D'après cette réponse, la combinaison conjonctive ou disjonctive n'a pas une sémantique bien fondée et peut conduire à des résultats contre-intuitifs et peu justifiables étant donné qu'elle consiste à agréger des rangs. Par ailleurs, le pouvoir d'expression du système DEDUCE2 est limité. Il ne considère que des termes décrits par des

	f(prix)	f(qtité)	rang prix	rang qtité	rang global
Savon	1000	100	2	2	2
Shampoing	500	0	1	1	1
Dentifrice	1500	100	3	2	3

TAB. 1.2 – Résultat de requête

fonctions monotones et vise uniquement à ordonner les éléments sélectionnés par la condition booléenne.

Le système PREFERENCES

Dans ce système, une question comporte une condition principale C et une préférence P. Une question peut s'écrire "C ;P" et s'interprète par : "trouver les éléments satisfaisant la condition C puis les ordonner selon P". PREFERENCES permet de combiner les clauses de préférences au moyen de deux constructeurs, l'imbrication (hiérarchie de conditions) et la juxtaposition (conditions de même importance) [LAC 87].
Si Rc est le sous ensemble d'éléments d'une relation R satisfaisant la condition C, l'imbrication des clauses de préférences $P_1, ..., P_n$ conduit à construire les ensembles H_i d'éléments de R satisfaisant les clauses de préférences P_1 à P_i mais pas P_{i+1}. De façon analogue, la juxtaposition des clauses de préférences $P1, ..., P_n$ permet de construire les ensembles J_i d'éléments de Rc satisfaisants i clauses de préférence.
La réponse retournée à l'utilisateur est l'ensemble H_i (resp.J_i) non vide d'indice i le plus élevé (ce qui correspond aux meilleurs éléments au sens de préférences).

Exemple 1.8.2.
Soit la relation simplifiée EMPLOYÉ de notre BD ENTREPRISE présentée dans la section 1.5 et la requête : trouver les noms des employés avec :
– *Une préférence p_1 pour ceux gagnant moins de 450 dt*
– *Une préférence p_2 pour ceux ayant plus de 27 ans.*
Nous avons la hiérarchie de préférence (p_1 ,p_2) et avec la consultation de la table 1.1, nous obtenons :
H_0 : Med Ali, Ramzi (P_1 fausse)
H_1 : Imen (P_1 vraie et P_2 fausse)
H_2 : Mouna (P_1 vraie et P_2 vraie)
De même pour la juxtaposition de P_1 et P_2, le résultat aurait été :

matricule	prénom	nom	adresse	salaire	âge	num_d
105	Ramzi	Guesmi	Tunis	500	31	02
106	Brahim	Daghbaji	Gabès	450	29	03
107	Imen	Ben Cheikh	Tunis	400	26	04
108	Mouna	Mallouli	Tunis	400	28	02
112	Med Ali	Ben Hssine	Tunis	500	24	02

TAB. 1.3 – Extension de la relation "Employé"

J_0 : Med Ali (P_1 fausse et P_2 fausse)
J_1 : Ramzi,Imen (P_1 vraie et P_2 fausse ou P_1 fausse et P_2 vraie)
J_2 : Mouna (P_1 vraie et P_2 vraie)

L'intérêt majeur de ce système consiste à éviter les formulations successives afin d'atteindre un ensemble souhaité de réponses. Cependant, nous constatons que nous ne pouvons distinguer que les éléments appartenant à des ensembles distincts et non pas les éléments d'un même ensemble, qui sont considérés tous équivalents. De plus, une telle formulation ne peut pas être supportée par un SGBD classique.

1.8.2 Distances associées aux domaines

Dans cette approche, les préférences sont directement intégrées aux conditions élémentaires à l'aide d'un opérateur de similarité (ou ressemblance). Celui-ci (noté $\approx$) étend l'égalité stricte et la condition élémentaire A$\approx$v est interprétée dans le cadre d'une distance définie sur le domaine de l'attribut A. L'idée est la suivante : v est la valeur idéale recherchée mais d'autres valeurs sont acceptables dans une moindre mesure. Plus A est proche de v, plus la distance est faible et si cette distance excède un seuil fixé, la condition n'est pas du tout satisfaite. Parmi les systèmes qui utilisent cette approche, nous pouvons citer ARES et VAGUE.

Le système ARES

Dans ce système [ICH 86], les conditions auxquelles l'utilisateur attache un seuil (S_i), sont connectées de façon uniquement conjonctive et l'interprétation de la requête :

$$(A_1 \approx v_1, S_1) \text{ et } \cdots \text{ et } (A_n \approx v_n, S_n)$$

s'opère en deux étapes :

– Sélection des n-uplets en utilisant les distances et les seuils :
$dist(A_1, v_1) \leq S_1$ et$\cdots$et $dist(A_n, v_n) \leq S_n$.
– Agrégation (ordonnancement) des n-uplets sélectionnés selon une distance globale :

$$dist\_glob = \sum_i dist(A_n, v_n) \ (\leq \sum_i S_n)$$

Exemple 1.8.3.
Soit la relation EMPLOYÉ de la table 1.3 et soit la requête : (salaire ≈ 400*, seuil=2) et (âge* ≈ 30*, seuil=1) avec les relations de distances spécifiées dans les tables 1.4 et 1.5.*

|sal1-sal2|	D
0	0
50	1
100	2
250	2

TAB. 1.4 – Distances sur les salaires (dist_sal)

|âge1-âge2|	d
0	0
1	1
2	1
3	2

TAB. 1.5 – Distances sur l'âge (dist_âge)

Imen et Med Ali sont écartés à cause de l'âge, il reste alors à sélectionner Mouna, Brahim et Ramzi qui ont les distances globales suivantes :

$dist\_glob(Mouna) = 0 + 1 = 1$ [5]
$dist\_glob(Brahim) = 1 + 1 = 2$
$dist\_glob(Ramzi) = 2 + 1 = 3$

[5] La distance globale de Mouna ($âge = 28$ et $salaire = 400$) se calcule comme suit : d'abord nous calculons les distances associées aux attributs salaire $dist\_sal(400, 400) = 0$ et âge $dist\_âge(28, 30) = 1$, puis nous calculons leur somme : $dist\_glob = 0 + 1 = 1$

puis les classer selon leur distance globale dans l'ordre suivant :
Mouna (1^{er}), Brahim ($2^{ième}$) et Ramzi($3^{ième}$)

Une requête dans ARES n'a pas une sémantique bien fondée car les distances considérées ne sont pas normalisées. En outre, la disjonction des conditions n'est pas autorisée dans ce système.

Le système VAGUE

Le système VAGUE utilise la même démarche que ARES [MOT 82]. Cependant, il considère les disjonctions, la pondération des conditions, la normalisation des distances par rapport aux seuils ainsi que l'emploi de la distance euclidienne pour la conjonction. Il procède au calcul de distance $dist_{pi}$ pour chaque condition imprécise P_i au moyen d'une métrique de donnée dont le rayon est noté r_i. Une métrique de donnée M pour un domaine D est définie comme une fonction $D \times D$ vers R satisfaisant les conditions suivantes :

1. $\forall x, y M(x, y) = 0$
2. $M(x, y) = 0 \Leftrightarrow x = y$
3. $M(x, y) = M(y, x)$
4. $\forall x, y, z M(x, y) \leq M(x, z) + M(z, y)$

Le rayon r_i associé à chaque métrique, représente la valeur maximale de satisfaction de la similarité. Par conséquent, la condition " $A \approx v$ " est satisfaite si et seulement si $M\ (A, v)\ \leq\ r$. Le système VAGUE transforme ensuite la distance $dist_{pi}$ en une distance $fdist_{pi}$ normalisée et pondérée par le degré d'importance w_i attribué par l'utilisateur à chaque condition imprécise P_i :

$$fdist_{pi} = \begin{cases} \dfrac{dist_{pi}(x)}{r_i} * w_i \text{ si } x \text{ satisfait } P_i \\ \infty \text{ sinon} \end{cases}$$

En cas de disjonction, la distance globale correspond à la plus petite valeur des distances associées aux conditions.

Les systèmes VAGUE et ARES présentent les limites suivantes :

– les réponses retournées à l'utilisateur sont, dans la plupart des cas, contre-intuitifs en raison de la présence de deux mécanismes différents utilisés successivement : la sélection et l'ordonnancement des n-uplets sélectionnés.

– Les conditions imprécises autorisées représentent uniquement une égalité étendue. Les conditions telles que "*bien-payé*" ne peuvent pas être exprimées à l'aide de l'opérateur de similarité $\approx$.

1.8.3 Préférences avec des termes linguistiques

Dans cette approche, l'idée est d'associer des termes de préférences aux conditions booléennes figurant dans la requête.
Parmi les systèmes qui se basent sur cette approche, nous pouvons citer MULTOS [RAB 90].

Le système MULTOS

Ce système permet d'exprimer la pondération entre les conditions de façon linguistique. L'évaluation d'une telle requête fait l'objet d'un traitement à deux étapes :
– La sélection à partir des valeurs plus ou moins acceptables pour chaque attribut,
– Classement des éléments sélectionnés après traduction numérique sur l'échelle [0,1] de toutes les étiquettes linguistiques (préférence et importance).
La valeur de classement est : $\begin{cases} P_{k,j} \times i_k \text{ si } X_k = v_{k,j} \\ 0 \text{ sinon} \end{cases}$
Pour tout critère de recherche de la forme :

$$X_k = v_{k,1}P_{k,1}, X_k = v_{k,2}P_{k,2}, \cdots X_k = v_{k,n}P_{k,n}i_k$$

où X_k désigne un attribut, $v_{k,i}$ une valeur dont la préférence est $P_{k,i}$ et i_k est l'importance du terme relatif à l'attribut X_k. En cas de disjonction de termes, la plus grande des valeurs obtenues pour chaque critère élémentaire de recherche sera retenue. Dans le cas de disjonction, la somme de ces valeurs sera calculée.

Exemple 1.8.4.
Pour exprimer que la condition d'âge (resp. de salaire) est plus ou moins satisfaite et qu'elle est plus importante que celle relative au salaire (resp. à l'âge), nous écrivons : $((\hat{a}ge = 29)$ *idéal,* $((\hat{a}ge = 30$ *ou* $\hat{a}ge = 31$ *ou* $\hat{a}ge = 28$ *ou* $\hat{a}ge = 27)$ *bon,* $((\hat{a}ge \leq 35$ *et* $ge \geq 32)$ *ou* $(\hat{a}ge \geq 23$ *et* $\hat{a}ge \leq 26))$ *tolérable*$)$ *élevé et* $((salaire \leq 400)$ *idéal,* $(salaire > 400$ *et* $salaire < 480)$ *tolérable*$)$ *moyen*$)$.
Les prédicats "idéal", "bon", et "tolérable" expriment les préférences sur les valeurs recherchées et les prédicats "élevé" et "moyen" indiquent l'importance des critères de recherche.

Soit la relation EMPLOYÉ de la table 1.3. Si les valeurs numériques associées aux étiquettes sont : idéal=1, bon=0.7, tolérable=0.3, élevé=1, moyen=0.5, la sélection disqualifie Med Ali et Ramzi dont les salaires ne conviennent pas.
En appliquant les formules mentionnées ci-dessus, nous aboutissons à un classement final :
1^{er} *Mouna,* $2^{ième}$ *Brahim et* $3^{ième}$ *Imen avec les valeurs respectives :* 1.2 [6] , 1.15 *et* 0.8.

1.8.4 Approche basée sur les ensembles flous

Avant de présenter l'approche basée sur les ensembles flous, nous mettons en relief les limites des approches décrites précédemment.

Limites des approches précédentes

Le traitement des requêtes flexibles dans tous les systèmes présentés précédemment se ramène à une sélection puis à un ordonnancement des éléments sélectionnés.
Par ailleurs, ces systèmes présentent plusieurs limitations :

1. leur comportement est souvent discontinu dans la plupart des cas (à l'exception de "PREFERENCES") au sens où un élément qui est tout à fait satisfaisant sur toutes les préférences sera sélectionné et classé alors qu'un élément idéal sur tous les critères sauf un est rejeté.
2. les préférences et le classement dans ces systèmes sont spécifiques. Ainsi, dans PREFERENCES, il n'y a pas de gradualité par rapport à une valeur de référence et dans l'approche fondée sur les distances, nous ne pouvons pas spécifier des conditions graduelles telles que "bien payé" ou "x bien plus grand que 80" puisqu'elles ne sont pas compatibles avec le caractère symétrique de la notion de distance ($dist(a+b,a) = dist(a-b,a)$).

Bosc [BOS 92,93] a montré que ces différents systèmes peuvent être décrits au moyen des sous ensembles flous.
La modélisation des requêtes flexibles par un formalisme unique présente l'avantage de mieux analyser le comportement de ces systèmes. Le recours aux ensembles flous

[6] Cette valeur se calcule comme suit : Mouna a 28 ans à laquelle correspond bon (0.7) comme préférence et élevé (1) comme importance et possède un salaire de 400 auquel est associé la préférence idéal (1) et l'importance moyen (0.5). La somme des deux valeurs de classement donne $0.7 * 1 + 1 * 0.5 = 1.2$

constitue un formalisme fédérateur pour l'expression et l'interprétation des requêtes flexibles.

Requêtes flexibles et ensembles flous

Cette approche se fonde sur les ensembles flous pour interpréter les critères de recherche flous. Ces derniers incluent :

- Des prédicats simples comportant des adjectifs comme "*jeune*" et "*large*".
- Des comparaisons entre deux attributs ou entre un attribut et une valeur donnée au moyen d'un opérateur relationnel imprécis comme "*beaucoup plus que*" et "*beaucoup moins que*".
- Des prédicats modulés par des modificateurs linguistiques comme "*très*", "*trop*", "*relativement*".
- Des prédicats quantifiés avec des quantificateurs flous comme "*la plupart*", "*environ la moitié*" et "*une douzaine*".
- Des prédicats composés (conjonction et disjonction des prédicats).

Plusieurs travaux ont utilisé la théorie des ensembles flous pour exprimer des requêtes flexibles.

Tahani [TAH 77] a introduit le concept de relation floue dans les SGBD. En effet, une relation floue RF est obtenue par l'application des critères flous sur une ou plusieurs relations de la BD. Chaque n-uplet t de RF est muni d'un degré d'appartenance, traduisant dans quelle mesure t appartient à RF. Cette mesure est obtenue par la restriction d'une relation R par un critère flou P. Elle est définie par :

$$\forall x \in X, \mu_{RF}(x) = min(\mu_R(x), \mu_P(x))$$

La conjonction et la disjonction des critères sont évaluées respectivement par min et max. dans cette approche, Tahani a défini une extension de l'algèbre relationnelle (sauf la division). Cependant, il n'offre pas un langage relationnel pour les utilisateurs.

Kacprzyk et Ziolkowski [KAC 86] ont traité les requêtes incluant des quantificateurs flous. Ils considèrent les requêtes de type :"chercher les éléments tels que Q parmi les conditions qui sont satisfaites" où Q est un quantificateur flou. Ce dernier est un quantificateur absolu comme "une douzaine", "environ3" ou un quantificateur relatif comme "presque tous", "la plupart". Ce quantificateur est représenté par un sous-ensemble flou défini sur R pour le premier type et défini sur [0.1] pour le dernier. Kacprzyk et Zadrozny [KAC 94] ont étendu cette approche et ont intégré

l'interrogation floue dans le SGBD Microsoft Access.
Bosc et Pivert [BOS 88] ont présenté une extension du langage relationnel SQL, nommée SQLf [BOS 95]. Ce langage a étendu pour les différents types de requêtes SQL (requêtes imbriquées, requêtes de division,...).
Médina et al. ont introduit un modèle pour les BDR floues, nommée GEFRED [MED 94a]. Ils ont présenté également un langage manipulant ces BD appelée FSQL [MED 94b]. Ce langage a subit plusieurs extensions par Galindo [GAL 98a]. Ce langage étend SQL pour introduire des quantificateurs flous, des constantes floues, des comparateurs flous,...
Les deux derniers travaux seront détaillés dans le chapitre qui suit.

1.9 Conclusion

Les BD sont devenues, sans conteste, le noyau de tout système informatique. Elles ont résolu la majorité des problèmes liés à la manipulation des données. Mais en contre partie, leur interrogation ne permet à l'utilisateur ni d'utiliser des termes linguistiques imprécis dans les critères de qualification des données recherchées, ni d'exprimer des préférences entre ces critères, ce qui présente souvent une demande légitime des utilisateurs.
Pour remédier à ces limites, deux concepts fondamentaux ont été introduit à savoir les requêtes flexibles et les BD floues.
Nous détaillerons dans le chapitre suivant les concepts de base des bases de données relationnelles floues.

Chapitre 2

Les bases de Données Floues

2.1 Introduction

Dans le cadre des BD, des valeurs imprécises peuvent apparaître dans de nombreuses situations. On peut citer la constitution d'entrepôts de données ou plus généralement la fusion de données dont les sources plus ou moins fiables apportent des informations non identiques, de bases contenant des données obtenues par des mécanismes de reconnaissance automatique délivrant des résultats intrinsèquement imprécis (reconnaissance d'images ou d'objets). Nous pouvons raisonnablement penser que dans le futur, de plus en plus d'applications vont recourir à l'utilisation de données imprécisément connues. Cette situation implique le stockage et la manipulation de telles informations dans des BD et donc l'extension des SGBD afin qu'ils offrent ce genre de fonctionnalités [BOS 88,98] [MED 94]. Cette extension est basée sur les ensembles flous qui offrent un formalisme puissant et permettant la flexibilité et la gradualité dans les requêtes. Dans ce chapitre, nous commençons par présenter les différents modèles des BDF, puis nous nous intéressons d'une part aux travaux de Bosc qui a introduit le langage SQLf et l'algèbre relationnelle étendue et d'autre part aux travaux de Médina qui a introduit le langage FSQL et l'algèbre relationnelle généralisée basée sur le modèle GEFRED.

2.2 Caractéristiques générales d'un modèle de BDF

Les aspects les plus importants de l'information vague que nous traitons habituellement sont : l'incertitude et l'imprécision. Le premier est dérivé des appréciations réalisées dans notre observation de la réalité.

Exemple : "Il me semble que Zied est un professeur d'Université".
Cependant, l'imprécision signifie à travers l'énoncé des concepts qui ne sont pas bien différenciés ou définis.
Exemple : "Zied est jeune", où jeune est un concept dont la sémantique peut varier suivant celui qui l'utilise et qui implique un ensemble de valeurs possibles pour l'âge de Zied.
Il est habituel que, dans nos conversations, splusieurs aspects e mélangent et c'est évident que les énoncés ont montré dans les deux exemples qu'ils fournissent un degré d'information sur le monde réel. Dans quelques cas, l'information qu'ils fournissent peut devenir insuffisante, mais dans d'autres peut ne pas l'être.
Toutes ces considérations ont mené à poser une convention que doit satisfaire un SGBD basé sur ces modèles . Cette convention possède les caractéristiques suivantes [PON 95] :

1. Le système doit fournir les mécanismes appropriés pour représenter l'information floue dans toutes ses formes.
2. Il doit offrir le cadre adéquat pour représenter et entreposer la signification de l'information floue qu'il héberge.
3. Il doit fournir un ensemble minimum d'opérateurs pour récupérer et traiter l'information floue.
4. Il doit satisfaire le plus possible les exigences du modèle relationnel. Comme nous avons vu, le modèle relationnel présente une série de caractéristiques qui ne sont pas complètes dans tout système de BD commercial. Il sera pris en compte, pour ce système, de penser que les modèles flous de BDR présentent, également, différentes adaptations à ce modèle.

2.3 Principaux modèles de bases de données floues

Le problème de la représentation et du traitement des informations "imprécises" a été étudié largement par plusieurs auteurs. Cependant, tous les modèles publiés pour donner la solution à ce problème ont leurs avantages, inconvénients et leurs limitations. Aussi, le terme "imprécision" inclut plusieurs significations ; il est peut être intéressant de distinguer : Que l'information que nous avons est incomplète, que nous ne savons pas si elle est certaine ou pas (incertitude), que nous l'ignorons totalement (inconnue) ou qu'elle n'est pas applicable à une certaine entité (indéterminée). Ces significations ne sont pas disjonctives quelquefois, mais peuvent plutôt

être unies dans une certaine information [GAL 99a]. Le problème n'est pas trivial, il est nécessaire de modifier la structure des relations et, avec celles ci, les opérations définies sur elles. Pour permettre d'entreposer l'information imprécise et de la consulter d'une façon flexible, cette information nécessite l'étude d'une multitude de cas particuliers qui ne se produisent pas dans le modèle classique.
Plusieurs modèles sont apparus dans la littérature, nous pouvons citer : le modèle Relationnel Flou [PON 95], le modèle de Buckles-Petry [BUC 82], le modèle d'Umano-Fukami [UMA 80], le modèle de Prade-Testemale [PRA 82], le modèle de Zemankova-Kaendel [ZEM 85]. Nous étudions dans cette section ces modèles en signalant leurs limites. Ensuite nous détaillons le modèle GEFRED de Medina-Pons-Vila [MED 94a] qui a pu surpasser ses limites.

2.3.1 Modèle Relationnel Flou

Le modèle de base des BD relationnelles floues (BDRF) se voit d'une forme très simple. Il consiste à ajouter un degré, habituellement dans l'intervalle [0,1], à chaque tuple [GAL 02] [CAS 96]. Ceci autorise à maintenir l'homogénéité des données dans la BD. Cependant, la sémantique qui est assignée à ce degré sera celle qui détermine son utilité et, par conséquent, cette sémantique sera utilisée dans les processus de consultation.
Plus formellement, une BD floue sera un ensemble de relations floues. Chaque relation est caractérisée par une fonction d'appartenance [PON 95] :

$$BD = \{R_1, R_2, ..., R_n\}$$

$$\mu_{R_i} : U_1 \times U_2 \times .... \times U_n \longrightarrow [0,1] \text{ et j=1...m}$$

Avec Ui le domaine de l'ième attribut de la relation y $\times$ le produit Cartésien Les principaux inconvénients de ce modèle sont :

- Il ne permet pas de représenter l'information imprécise que nous portons sur certain attribut, en particulier de quelques entités concrètes (comme la valeur "jeune" ou "vieux" pour l'attribut "âge").
- Il assigne à chaque tuple un caractère flou de forme globale sans déterminer quelle est la participation de chaque attribut.
- Il est incapable de satisfaire les exigences du modèle relationnel relatif à l'unicité, à l'accessibilité de chaque tuple et l'implémentation de quelques opérateurs relationnels.

2.3.2 Modèle de Buckles-Petry

C'est le premier modèle qui utilise les relations de similitude dans le modèle relationnel, proposé par Buckles et Petry [BUC 82], dans le quel les BD non floues sont un cas particulier de ce modèle. Une relation floue est définie comme un sous-ensemble du produit Cartésien $2^{D_1} \times 2^{D_2} \times .... \times 2^{D_n}$ où 2^{D_i} est un élément quelconque de l'ensemble des parties du domaine ($P(D_i)$). Les valeurs qui peuvent être représentées sont les suivantes :

– Ensemble fini de scalaires. Exemple {Blond, marron, roux}.
– Ensemble fini de nombre. Exemple {10, 11, 12}.
– Ensemble de nombres ou étiquettes flous. Exemple {petit, moyen, grand}

La relation de ressemblance qui existe sur chacun des domaines sert à représenter et diriger l'imprécision. Elle établit une mesure de similitude entre les différentes valeurs du domaine sur lequel elle est définie. Elle est définie par l'utilisateur et les valeurs de ressemblance sont comprises entre 0 et 1. (0 : Complètement différent ; 1 : Complètement semblable).

Dans une consultation (interrogation), l'utilisateur pose une question sur les tuples qui satisfaisaient une condition déterminée pour un seuil de similitude donné. Les tuples de la relation se regroupent en classe d'équivalences, suivant les relations et les seuils de similitudes définis sur eux.

Exemple 2.3.1.

Soit la relation Personnel décrite dans la table 2.1 et soit l'attribut couleur_de_cheveux défini sur le domaine $D_{couleur\_de\_cheveux}$ ={blond, châtain, rouille, noir}. Supposons que nous avons les relations de similitudes présentées par le tableau 2.2.

nom	âge	hauteur	couleur_de_cheveux
Med Ali	16	{grande, très-grande}	châtain
Sami	17	petite	noir
Ramzi	15	très-petite	blond
Sana	{15, 16}	moyenne	rouille

TAB. 2.1 – Extension de la relation Personnel

Soit la requête suivante : "donner les noms et les hauteurs des personnes ayant la couleur des cheveux semblable au châtain avec un degré 0.6". Dans cette requête le seuil de similitude est seuil($D_{couleur\_de\_cheveux}$)=0.6 et la classe d'équivalence est donnée dans le tableau 2.3.

	blond	**châtain**	**rouille**	**noir**
blond	1	0.6	0.4	0
châtain	0.6	1	0.5	0.1
rouille	0.4	0.5	1	0.8
noir	0	0.1	0.8	1

TAB. 2.2 – Relations de similitude pour l'attribut couleur de cheveux

nom	**âge**	**hauteur**	**couleur_de_cheveux**
{Med Ali, Sami}	{16, 17}	{très-grande, grande, petite}	{châtain, noir}
Ramzi	15	très-petite	blond
Sana	{15, 16}	moyenne	rouille

TAB. 2.3 – Définition des classes d'équivalences

Le résultat final (après l'application des opérateurs relationnels) est :

nom	**hateur**
{Med Ali, Sami}	{très-grande, grande, petite}

TAB. 2.4 – Résultats final de la requête

Les principaux inconvénients de ce modèle sont :
- Il ne modélise pas bien tous les aspects flous de l'information.
- Il ne garantie pas l'atomicité dans la représentation de l'information.
- Il ne garantie pas l'intégrité de la BD.
- Il peut donner lieu à des résultats confus lors de l'application des opérateurs relationnels aux classes d'équivalences.
- Il présente un résultat possédant plusieurs interprétations.

Cependant, nous pouvons signaler une série d'avantages :
- L'emploi de relations de ressemblance fournissant un outil approprié et intuitif pour représenter l'imprécision de concepts.
- L'utilisation de différents seuils pour chacun des attributs impliqués dans une consultation.

2.3.3 Modèle de Umano-Fukami

C'est l'un des premiers modèles des BDRF, il se base sur la théorie des possibilités [UMA 80]. La différence avec les autres modèles réside dans la manière de décrire l'information floue. Il utilise comme valeur :

- les distributions de possibilité
- les valeurs indéterminée, inconnu et nulle avec les sémantiques suivantes :

  Undefined : $\pi_{A(x)}(d) = 0, \forall d \in D$

  Unknown : $\pi_{A(x)}(d) = 1, \forall d \in D$

  Null = {1/Unknown, 1/Undefined}

  Avec D est l'univers de discours de A(x) et $\pi_{A(x)}(d)$ représente la possibilité que A(x) prend la valeur u de U.

Ce modèle se divise en trois sous-ensembles et le résultat de la consultation retourne :

- Les tuples qui satisfaisaient clairement la consultation.
- Les tuples qui satisfaisaient approximativement la consultation.
- Les tuples qui ne satisfaisaient pas clairement la consultation.

L'avantage de ce modèle est qu'il peut entreposer des distributions de possibilités, comme il peut assigner un degré d'appartenance à chaque tuple de la relation.

Exemple 2.3.2.

Soit la relation Dépendant donnée par le tableau 2.5.

nom	**âge**	**enfants**
Med Ali	23	Yassin
Amel	35	Youssef
Hbib	45	{Aymen, Sihem}
Kamel	jeune	Unknown
Tarek	Unknown	Undefined
Achref	{50,51}	Null

TAB. 2.5 – Extension de la relation Dépendant

Supposons que la distribution de possibilité de l'étiquette "jeune" est jeune = {0.3/15, 0.6/16, 0.8/17, 1/19, 1/20, 1/21 1/22, 1/23, 0.9/25, .8/ 25, 0.7/26}. Si nous établissons une requête de type : "donner les personnes qui ont plus de 25 ans", nous aboutissons au résultat comprenant les trois sous-ensembles suivants :

- *Clairement satisfaisant = {1/Amel, 1/Hbib, 1/Achref}*
- *Possiblement satisfaisant = {1/Kamel, 1/Tarek}*
- *Clairement non satisfaisant = {1/Med Ali}*

2.3.4 Modèle de Prade-Testmale

Ce modèle se base sur la théorie des possibilités pour représenter les données imprécises ou incomplètes. Il permet d'incorporer les données "incomplètes" ou "imprécises" de la théorie de Possibilité. La structure de donnée est similaire à celle utilisée dans Umano-Fukami. Il utilise des mesures de possibilité et de nécessité pour la satisfaction des conditions établies dans la consultation.
Les domaines autorisés dans ce modèle sont :
- Ensemble fini de scalaires. Exemple D={Blond, marron, roux}
- Ensemble fini de nombre. Exemple D={10, 11, 12}
- Ensemble de nombres ou étiquettes flous. Exemple D={petit, moyen, grand}

Ces domaines peuvent prendre les valeurs suivantes :

1. Une unique valeur d ∈ D parfaitement connu. Exemple âge(Ramzi)=19 ou couleur_cheveux(Med)=noir. Par contre, ce modèle ne considère pas le cas de différentes valeurs parfaitement connues (attributs multi-valués) comme par exemple langue_parlé (Med)= français et anglais.
2. Une valeur nulle qui n'inclut pas la valeur inconnue et non applicable.
3. Une distribution de possibilité sur le domaine de l'attribut. Par exemple, Hauteur (Ramzi) = "pas très haut", où pas très haut est un une valeur floue sur le domaine de la hauteur de l'attribut de la forme {0.8/1.60, 0.9/1.65, 1/1.70, 1/1.75, 0.8/1.80}

Exemple 2.3.3.
Soit la relation Etudiant (nom, âge, mathématiques, informatique) :
[8,10] représente un intervalle de valeur, alors que "bonne", "jeune", etc., représentent des distributions de possibilités. Pour répondre à une requête de type "trouver les étudiants dont la qualification en programmation (Note en informatique) est mieux que bonne", il faut définir la distribution de possibilité qui correspond à "bonne" et le comparateur flou "mieux que".

nom	**âge**	**mathématiques**	**informatique**
Amel	jeune	8	[8,10]
Hbib	Approximativement 24	7	undefined
Med Ali	[22, 24]	Légèrement mal	Pas très mal
Ramzi	19	8	9.5
Kamel	Unknown	bien	bien
Noureddine	très-jeune	9	Approximativement 5
Sami	22	3	7

TAB. 2.6 – Extension de la relation Etudiant

Soit par exemple : bonne = {0.3/7, 0.6/8, 0.9/9, 1/10}.

$$\mu_{mieux\ que}(u,v) = \begin{cases} 0 & si\ u - v \leq 2 \\ 0.5 & si\ u - v = 3 \\ 1 & si\ u - v \geq 4 \end{cases}$$

D'après cette règle, la composition d'un sous-ensemble flou A, en admettant l'univers de discours U et l'opérateur relationnel flou R dans $U \times V$ sera comme suit :

$$A \circ R = max_{u \in U}(\mu_A(u) \wedge \mu_R(u,v))/v \in V.$$

2.3.5 Modèle de Zemankova-Kaendel

Ce modèle se compose de trois parties [ZEM 85] :

- une BD de valeurs (BDV) dans laquelle s'organisent les données dans une forme similaire aux autres modèles possibilistes.
- une BD explicative où se stockent les définitions pour les sous ensembles flous et les relations floues.
- un ensemble de règles de traduction qui servent à la manipulation des adjectives.

La manipulation des données se base sur le modèle relationnel et le résultat d'une consultation se présente dans la forme de relations floues. Celles-ci contiennent deux champs : les valeurs de possibilités et les valeurs de certitudes que présente chaque tuple pour la consultation.

2.3.6 Discussion

Malgré que les modèles décrits précédemment ont présenté des propriétés désirables pour les BDF, aucun d'eux n'a satisfait complètement les caractéristiques d'un mo-

dèle de BDF [CAS 96].
Les différents niveaux "flous" qui peuvent être couvert par un SGBDRF sont :
– Obtenir de l'information floue à partir des données précises.
– Représenter et récupérer ces informations.
– Traiter ces informations.
Il reste encore le problème de stockage des données dans les BDF et la dépendance avec modèle relationnel, qui n'ont pas été traité par ces modèles.

2.3.7 Modèle GEFRED de Médina et al

Le modèle **GEFRED** (**GE**neralised model for **F**uzzy **RE**lationnel **D**atabase) a été proposé en 1994 par Medina, Pons et Vila [MED 94a]. Il constitue une synthèse éclectique des différents modèles publiés pour traiter le problème de la représentation et du traitement des informations floues au moyen des BDR. Un des principaux avantages de ce modèle est qu'il consiste à une abstraction générale qui permet de traiter différentes approches, même celles qui peuvent paraître très disparates. Il se base sur le *Domaine Flou Généralisé (D)* et sur la *Relation Floue Généralisée (R)*, qui incluent respectivement les domaines et les relations classiques. Nous présentons dans ce qui suit les concepts de base de ce modèle.

Définition 2.3.1.
Si U est le domaine du discours, P(U) est l'ensemble de toutes les distributions de possibilité défini sur U, y compris celles qui définissent les types Inconnu, Indéterminé et NULL (type 8, 9 et 10 de la table 2.7). Le ***Domaine Flou Généralisé****, est défini comme* $D \subseteq P(U) \cup NULL$.

Le Domaine Flou Généralisé constitue l'élément structurel qui organise la représentation des types de données de la table 2.7 [MED 94a].

Définition 2.3.2.
Une ***Relation Floue Généralisée****, R, est donné par deux ensembles "Entête" (Head H) et "Corps" (Body B), R = (H,B), définis comme ceci :*
– *L'Entête comporte un ensemble fixe de triple "attribut-domaine-compatibilité" (en anglais "attribute-domain-compatibility") où le dernier est optionnel,*

$$\mathcal{H} = \{(A_1 : D_1[, C_1]), (A_2 : D_2[, C_2]), ..., (A_n : D_n[, C_n])\}$$

Où chaque attribut A_j *a un sous domaine flou* D_j *(j=1,2,... n) qui n'est pas nécessairement différent aux autres et* C_j *est un "attribut de compatibilité" qui prend des valeurs dans l'intervalle [0,1].*

1. Une seule valeur linguistique (Comportement = bon, représenté par la distribution de possibilité, 1/bon)
2. Un seul nombre (Âge = 28, représenté par la distribution de possibilité, 1/28)
3. Un ensemble de distributions linguistiques mutuellement exclusives (Comportement = {bon, mauvais}, représenté par {1/bon, 1/mauvais}).
4. Un ensemble de distributions numériques possibles mutuellement exclusives (Âge = {20,21}, représenté par {1/20,1/21}).
5. Une distribution de possibilité dans un domaine linguistique {Comportement={0.6/mauvais, 0.7/normal}).
6. Une distribution de possibilité dans un domaine numérique (Âge = {0.4/23, 1.0/24, 0.8/25}, nombres flous ou étiquettes linguistiques).
7. Un nombre réel qui appartient à [0,1] référencié à un degré de réalisation (Qualité = 0.9).
8. Une valeur inconnue (UNKNOWN) avec une distribution de possibilité, **UNKNOWN** = {1/u : u ∈ U}.
9. Une valeur indéterminée (UNDEFINED) avec une distribution de possibilité, **UNDEFINED** = {0/u : u ∈ U}.
10. Une valeur nulle (NULL) donnée par NULL = {1/Unknown, 1/Undefined}.

TAB. 2.7 – Types de données dans **GEFRED**

– *Le Corps comporte un ensemble de tuples, appelés "tuples flous généralisés"(en anglais generalized fuzzy tuples), où chaque tuple est composé d'un ensemble de triple "attruibut-valeur-degré" (attribute-value-degrees) où le degré est optionnel,*

$$\mathcal{B} = \{A_1 : \tilde{d}_{i1}[, c_{i1}]), A_2 : \tilde{d}_{i2}[, c_{i2}]), ..., (A_n : \tilde{d}_{in}[, c_{in}])\}$$

Avec (i = 1,2..., m), et m est le nombre de tuples dans la relation et où le d_{ij} représente la valeur du domaine pour le tuple i et l'attribut A_j, et le c_{ij} est le degré de la compatibilité associé à cette valeur [MED 94a].

Le degré de compatibilité est utilisé pour savoir le degré avec lequel une valeur d'un attribut a satisfait la condition de la requête.

2.4 Algèbre relationnelle Floue

L'algèbre relationnelle a été étendue en vue d'introduire les concepts flous. Les principaux travaux amenés dans ce sens ont aboutit à introduire deux familles d'algèbres,

à savoir l'algèbre relationnelle étendue et l'algèbre relationnelle généralisée.

2.4.1 Algèbre relationnelle étendue

Nous nous plaçons maintenant dans le cadre des relations graduelles pour lesquelles les opérateurs de l'algèbre relationnelle vont être étendues de sorte que chacune d'elles opèrent sur une (des) relation(s) graduelle(s)[1] et retourne une telle relation. L'algèbre relationnelle étendue a été introduite par Bosc pour étendre l'algèbre relationnelle classique [BOS 98]. Dans ce sens, les opérations de l'algèbre relationnelle ont été étendues de façon naturelle aux relations floues, d'une part en les considérant comme des ensembles flous, d'autre part en introduisant des prédicats graduels, au lieu des prédicats booléens. En considérant deux relations R et S définies sur le même ensemble d'attribut X, nous avons :

- Union : $\mu_{R\cup S}(x) = max(\mu_R(x), \mu_S(x))$
- Intersection : $\mu_{R\cap S}(x) = min(\mu_R(x), \mu_S(x))$
- Différence : $\mu_{R-S}(x) = min(\mu_R(x), 1 - \mu_S(x))$
- Produit cartésion : $\mu_{R\times S}(xy) = max(\mu_R(x), 1 - \mu_S(y))$

Les opérations de sélection, projection et jointure appliquées à des relations graduelles sont définies par :

- Sélection : $\mu_{\sigma\varphi(R)}(x) = min(\mu_R(x), \mu_\varphi(x))$ où φ est un prédicat graduel
- Projection : $\mu_{\pi_\gamma(R)}(u) = max_v \mu_R(uv)$ où γ est un sous-ensemble de X, u prend ses valeurs dans γ et v dans (X-γ) (s'il y a des n-uplets doubles le plus grand sera choisi).
- Jointure : $\mu_{R_{A\theta B}S}(xy) = min(\mu_R(x), \mu_S(y), \mu_\theta(x.A, y.B)))$ où A (resp. B) est un sous-ensemble de X (resp. Y), a et b sont compatibles (càd sont définis sur des domaines de valeurs identiques), θ est un comparateur (éventuellement graduel) binaire. x.A (resp. y.B) désigne la valeur du composant a (resp. b) du tuple x (resp. y). On a aussi $R_{A\theta B}S = \sigma_{A\theta B}(R \times S)$.
- Division : plusieurs définitions qui sont à l'origine de plusieurs travaux, parmi elles :
  - Si x désigne une valeur de X, A et B des ensembles d'attributs compatibles, la division de R(A,X) par S(B, Y) est définie par une implication : $x \in R_{A\div B}S \Leftrightarrow [x \in \Pi_x(R)$ et $(\forall a, a \in \Pi_y(S) \Rightarrow (a, x) \in R)]$

    L'extension de la division consiste à considérer $\Pi_x(R)$ comme un référentiel qui sera généralisé par le support, remplacer l'implication usuelle par une impli-

[1] une relation graduelle est une relation floue (définition 1.2.6)

cation floue et interpréter le quantificateur universel comme une conjonction généralisée [BOS 97], soit :
$\mu_{R_{A\div B}S}(x) = min(\mu_{\Pi_x(support(R))}(x), min_s \quad \mu_S(s) \longrightarrow \mu_R(s.B, x))$
ou encore $\forall x \in \Pi_x(support(R)), \mu_{R_{A\div B}S}(x) = min_s \quad \mu_S(s) \longrightarrow \mu_R(s.B, x))$

– Une autre forme a été présenté dans [BOS 98] pour étendre la division usuelle : $R_{A\div B}S = \Pi_x(R) - \Pi_x(\Pi_x(R) \times \Pi_B(S) - R)$ à la division étendue : $R_{A\div B}S = \Pi_x(support(R)) - \Pi_x(\Pi_x(support(R)) \times \Pi_B(S) - R)$ à la double condition que l'opération " support " soit introduite et qu'un ensemble adéquat d'opérateurs de différence ensembliste soit disponible. Par exemple, en partant de R-S=R$\cap\overline{S}$ et en choisissant la norme "min" pour l'intersection, nous retrouvons l'interprétation donnée par l'implication Kleene-Dienes à condition que S soit normalisée.

Exemple 2.4.1.
Soit la relation EMPLOYÉ (matricule, prénom, nom, salaire, âge, adresse) et la relation DÉPARTEMENT (num_ d, budget, nbre_ employé, local). Considérons la requête :" trouver les numéros de département à gros budget *ayant au moins un employé de salaire* élevé *de moins de 40 ans". Cette requête s'écrit :*
$\Pi_{ndep}(\sigma_{budget="gros"}(DPARTEMENT)) \cap \Pi_{dep}(DPARTEMENT_{ndep=dep}$
$(\sigma_{age<40 et salaire="eleve"}(EMPLOY)))$

Exemple 2.4.2.
On considère les deux tables correspondantes aux relations **TRAVAILLE_DANS** *(matricule, num_ p, nbre_ heure,μ_{longue_dure}) et* **PROJET***(num_ p, nom, local, num_ d, budget, μ_{gros_budget}) obtenues après avoir appliquer les conditions floues : nbre_ heure=* "longue durée" *et budget=*"gros" *et présentées dans les tables 2.8 et 2.9.*
Soit aussi la requête : "trouver les employés auxquels nous avons affecté pendant des longues heures tous les projets à gros budget*".*
En partant de l'expression $R_{A\div B}S = \Pi_x(R) - \Pi_x(\Pi_x(R) \times \Pi_B(S) - R)$, nous pouvons conclure que :
$TRAVAILLE\_DANS_{num\_p\sharp \div num\_p\sharp} PROJET = \Pi_x(support(TRAVAILLE\_DANS) -$
$\Pi_x(\Pi_x(support(TRAVAILLE\_DANS) \times \Pi_B(PROJET) - TRAVAILLE\_DANS)$
$\Pi_x(support(R)) = \Pi_{matricule}(support(TRAVAILLE\_DANS)) = \{105, 107\};$
$\Pi_B(S) = \Pi_{number\sharp}(PROJET) = PROJET = \frac{0.6}{01}, \frac{0.4}{02}, \frac{1}{03}$
$\Pi_x(support(R)) \times \Pi_B(s) = \{\frac{0.6}{<105,01>}, \frac{0.4}{<105,02>}, \frac{1}{<105,03>}, \frac{0.6}{<107,01>}, \frac{0.4}{<107,02>}, \frac{1}{<107,03>}\}$
$\Pi_x(support(R)) \times \Pi_B(s) - R = \{\frac{0.4}{<105,02>}, \frac{0.1}{<105,03>}, \frac{0.4}{<107,02>}\}$
$\Pi_x(\Pi_x(support(R)) \times \Pi_B(s) - R) = \{\frac{0.4}{105}, \frac{0.4}{107}\}$
$\Pi_x(support(R)) - \Pi_x(\Pi_x(support(R)) \times \Pi_B(s) - R) = \{105, 107\} - \{\frac{0.4}{105}, \frac{0.4}{107}\} = \{\frac{0.6}{105}, \frac{0.6}{107}\}$

matricule	num_p	nbre_heure	$\mu_{longue\_\text{durée}}$
105	01	3000	1
105	02	1000	0.3
105	03	2700	0.9
105	04	1500	0.5
107	01	3200	1
107	02	850	0.2
107	03	3000	1

TAB. 2.8 – Relation floue des heures à longue durée de la table TRAVAILLE_DANS

num_p	nom	local	num_d	budget	μ_{gros_budget}
01	éclairage	Tunis	02	80000	0.6
02	peinture	Gabes	04	50000	0.4
03	autoroute	Sousse	03	140000	1
04	peinture	Nabeul	02	800	0
05	climatisation	Zaighouan	04	900	0

TAB. 2.9 – Relation floue des projets à gros budget

Avec ses définitions, les propriétés de l'algèbre relationnelles sont conservées, comme la distributivité de l'union par rapport à l'intersection (et vice-versa), la décomposition des conjonctions et des disjonctions, la distributivité de la sélection par rapport à l'union, l'intersection et la différence de la projection par rapport à l'union ou encore la décomposition de la sélection par rapport à la jointure. Outre ces opérations usuelles, d'autres opérations sont étendues comme la semi-jointure et l'anti-jointure.

2.4.2 Algèbre Relationnelle Généralisée

Le modèle GEFRED décrit une Algèbre Relationnelle Floue Généralisée pour manipuler les relations floues généralisées [MED 94a]. Les opérations classiques, Union, Intersection, Différence, Produit Cartésien, Projection, Jointure et Sélection, sont étendues pour leur permettre d'opérer d'une manière cohérente.
La Sélection et la Jointure se basent dans leurs opérations sur l'utilisation de comparateurs flous. Ce modèle adopte un cadre classique pour construire les comparateurs flous qui commencent par la signification d'opérations de comparaison définis sur

le domaine du discours, jusqu'à étendre cette signification au traitement de valeurs des domaines floues généralisées.
Avant d'atteindre les opérations de l'algèbre relationnelle floue généralisée, nous allons tout d'abord présenter les définitions des opérateurs de comparaison floues utilisées dans cette algèbre.

Opérateurs de Comparaison Floues

Les opérateurs de comparaison flous sont redéfinis dans le but de les adapter à la nature des données présentées dans GEFRED.

Définition 2.4.1.
Soit U le domaine du discours considéré, le `Comparateur Étendu`*, θ, se défini comme toute relation floue sur U qui peut être exprimé selon la façon suivante :*

$$\theta : U \times U \longrightarrow [0,1]$$

$$\theta(u_i, u_j) \longrightarrow [0,1]$$

avec $u_i, u_j \in U$

Le Comparateur Étendu rassemble toutes les modalités de la relation qui existent parmi les valeurs de domaine du discours considéré U, en prenant en considération la nature spécifique de ces valeurs et le caractère, classique ou flou, de la relation qui existe entre eux [MED 94a].
Avec cette définition, le Comparateur Étendu, θ, nous permet de modeler, dans un chemin logique, les opérateurs de la comparaison suivants :

- Comparateurs classiques de l'Algèbre Relationnelle tel que : $=$,$\neq$, $>$, $\geq$, $<$, $\leq$. Par exemple, l'opérateur "égal étendu" serait exprimé par $=_e (d_i, d_j) = \delta(d_i, d_j)$ où $\delta(d_i, d_j)$=1 pour $d_i = d_j$ et $\delta(d_i, d_j)$=0 pour $d_i \neq d_j$.
- Comparateurs flous tel que "approximativement égal", "beaucoup de plus grand que",... Ce genre de comparateur sera exprimé par des fonctions d'appartenances. Par exemple, l'opérateur "approximativement égal" pourrait être modelé par le comparateur étendu correspondant $\backsimeq$ avec la fonction d'appartenance suivante : $\mu : (d_i, d_j) = e^{-\beta|d_i - d_j|} \quad avec \quad \beta > 0$
- Comparateurs de Similitude qui opèrent sur des données scalaires avec une relation de similitude établie entre eux.

Définition 2.4.2.

Soit U le domaine du discours considéré, D le domaine flou généralisé construit sur lui et un comparateur étendu défini sur U. Considérons une fonction Θ^θ définie par :

$$\Theta^\theta : D \times D \longrightarrow [0,1]$$

$$\Theta^\theta(d_1, d_2) \longrightarrow [0,1]$$

Θ^θ est dit `Comparateur Flou Généralisé` sur D abouti par le comparateur étendu , s'il vérifie : $\Theta^\theta(\widetilde{d_1}, \widetilde{d_2}) = \theta_e(d_1, d_2) \forall d_1, d_2 \in U$.
Où $\widetilde{d_1}, \widetilde{d_2}$ représentent les distributions de possibilité ,$1/d_1$, $1/d_2$, induits respectivement par les valeurs d_1, d_2.

Opérations de l'algèbre relationnelle floue

Union Floue Généralisée

Définition 2.4.3.

Soient R et R' deux relations floues généralisées données par :

$$R = \begin{cases} \mathcal{H} = \{(A_1 : D_1[, C_1]), ..., (A_n : D_n[, C_n])\} \\ \mathcal{B} = \{(A_1 : \widetilde{d}_{i1}[, c_{i1}]), ..., (A_n : \widetilde{d}_{in}[, c_{in}])\} \end{cases} \quad (1)$$

$$R' = \begin{cases} \mathcal{H}' = \{(A'_1 : D'_1[, C'_1]), ..., (A'_n : D'_n[, C'_n])\} \\ \mathcal{B}' = \{(A'_1 : \widetilde{d'}_{k1}[, c'_{k1}]), ..., (A'_n : \widetilde{d'}_{kn}[, c'_{kn}])\} \end{cases} \quad (2)$$

*Avec i = 1,...,m et k = 1,...,m', soient m et m' les cardinalités respectives et n et n' les degrés respectifs, et par suite l'*Union Floue Généralisée *de R et R' sera définie comme :*

$$R \cup R' = \begin{cases} \mathcal{H}_\cup = \{(A_1 : D_1[, C_1]), ..., (A_n : D_n[, C_n])\} \\ \mathcal{B}_\cup = \begin{cases} \mathcal{B}^v_\cup = \mathcal{B}^v \cup \mathcal{B}'^v \\ \mathcal{B}^c_\cup = \{[c"_{i1}], ..., [c"_{in}]\} \end{cases} \end{cases} \quad (3)$$

$$c"_{ij} = \begin{cases} max(c_{ij}, c'_{ij}) & si\ \exists c_{ij}\ et\ \exists c'_{ij} \\ c_{ij} & si\ \exists c_{ij}\ et\ \nexists c'_{ij} \\ c'_{ij} & si\ \exists c'_{ij}\ et\ \nexists c_{ij} \\ 0 & si\ \nexists c_{ij}\ et\ \nexists c'_{ij}\ et\ \exists C_n\ et\ \exists C'_{n'} \end{cases} \quad (4)$$

Avec i = 1,...,m" et m" sera la cardinalité de l'union.
c_{ij} sera le degré de compatibilité de d"$_{ij}$ pour le tuple i dans la relation R et c'_{ij} le degré de compatibilité de d"$_{ij}$ pour le tuple i dans la relation R'.

Par conséquent, l'Union Floue Généralisée de deux relations floues généralisées contient les tuples qui appartiennent à l'union des deux relations, adoptant, pour le degré de compatibilité de chaque tuple, $c"_{ij}$, le "maximum" des degrés de compatibilité des valeurs dans les relations origines comme il est montré dans l'équation 4.

Intersection Floue Généralisée

Définition 2.4.4.
*Soient R et R' deux relations floues généralisées, compatibles avec l'union, données par les équations (1) et (2). L'*Intersection Floue Généralisée *de R et R' sera définie comme ceci :*

$$R \cap R' = \begin{cases} \mathcal{H}_\cap = \{(A_1 : D_1[, C_1]), ..., (A_n : D_n[, C_n])\} \\ \mathcal{B}_\cap = \begin{cases} \mathcal{B}^v_\cap = \mathcal{B}^v \cap \mathcal{B}'^v \\ \mathcal{B}^c_\cap = \{[c"_{i1}], ..., [c"_{in}]\} \end{cases} \end{cases} \quad (5)$$

$$c"_{ij} = \begin{cases} min(c_{ij}, c'_{ij}) & si\ \exists c_{ij}\ et\ \exists c'_{ij} \\ c_{ij} & si\ \exists c_{ij}\ et\ \nexists c'_{ij} \\ c'_{ij} & si\ \exists c'_{ij}\ et\ \nexists c_{ij} \end{cases} \quad (6)$$

Avec i = 1,...,m et m" sera la cardinalité de l'union. c_{ij} sera le degré de compatibilité de d"$_{ij}$ pour le tuple i dans la relation R et c'_{ij} le degré de compatibilité de d"$_{ij}$ pour le tuple i dans la relation R'.

Par conséquent, l'Intersection Floue Généralisée de deux relations floues généralisées contient les tuples qui appartiennent à l'intersection des deux relations, adoptant, pour le degré de compatibilité de chaque tuple dans l'intersection, $c"_{ij}$, le "minimum" des degrés de compatibilité adoptés des valeurs dans les relations origines comme il est montré dans l'équation 6.

Différence Floue Généralisée

Définition 2.4.5.
Soient R et R' deux relations floues généralisées données par les équations (1) et (2). La Différence Floue Généralisée *de R et R' sera définie par :*

$$R - R' = \begin{cases} \mathcal{H}_- = \{(A_1 : D_1[, C_1]), ..., (A_n : D_n[, C_n])\} \\ \mathcal{B}_- = \begin{cases} \mathcal{B}^v_- = \mathcal{B}^v \\ \mathcal{B}^c_- = \{[c"_{i1}], ..., [c"_{in}]\} \end{cases} \end{cases} \quad (7)$$

$$c"_{ij} = \begin{cases} min(c_{ij}, 1 - c'_{ij}) & si\ \exists C_n\ et\ \exists C'_{n'}\ et \quad i \in \mathcal{B}^v \cup \mathcal{B}'^v \\ c_{ij} & si\ \exists C_n\ et\ \nexists C'_{n'} \\ c'_{ij} & si\ \nexists C'_{n'}\ et\ \exists C_n\ et \quad i \in \mathcal{B}^v - \mathcal{B}'^v \\ 1 - c'_{ij} & si\ \nexists C_n\ et\ \nexists C'_{n'}\ et \quad i \in \mathcal{B}^v - \mathcal{B}'^v \end{cases} \quad (8)$$

Avec i = 1,...,m et m" sera la cardinalité de l'union où c_{ij} sera le degré de compatibilité de d"$_{ij}$ pour le tuple i dans la relation R et c'_{ij} le degré de compatibilité de d"$_{ij}$ pour le tuple i dans la relation R'.

Par conséquent, la Différence Floue Généralisée de deux relations Floues Généralisées contient les tuples qui appartiennent à la différence des deux relations, adoptant, pour le degré de compatibilité de chaque tuple dans la différence, $c"_{ij}$, les valeurs indiquées dans l'équation 8.

Produit Cartésien Flou Généralisée

Définition 2.4.6.
Soient R et R' deux relations floues généralisées données par l'équation (1), par suite le Produit Cartésien Flou Généralisé de R et R' sera définie comme une relation floue généralisée donnée par :

$$R \times R' = \begin{cases} \mathcal{H}_\times = \mathcal{H} \cup \mathcal{H}' \\ \mathcal{B}_\times = \mathcal{B} \times \mathcal{B}' \end{cases} \quad (9)$$

Projection Floue Généralisée

Définition 2.4.7.
Soit R une relation floue généralisée donnée par l'équation (1), et soit X un sous ensemble de H exprimé par :
$X \subseteq \mathcal{H}, X = \{(A_s : D_s[, C_{s'}]) : s \in S, s' \in S'; S, S' \subseteq \{1, ..., n\}\}$
La Projection Floue Généralisée de R dans X, $\mathcal{P}$(R ;X) est une relation floue généralisée donnée par :

$$\mathcal{P}(R; X) = \begin{cases} \mathcal{H}_\mathcal{P} = X \\ \mathcal{B}_\mathcal{P} = \{(A_s : d_{is}[, c_{is'}])\} \end{cases} \quad (10)$$

Avec $s \in S, s' \in S'$ et $S, S' \subseteq \{1, ..., n\}$.

La projection peut être emportée sur tout sous-ensemble dans l'*entête*, qui peut inclure "les valeurs d'attributs" et "les attributs de compatibilité". Le résultat de la projection est une "sélection verticale" sur le *corps* de la relation.

Sélection Floue Généralisée

Définition 2.4.8.
Soit R une relation floue généralisée donnée par l'équation (1). Soit aussi une constante $\widetilde{a} \in D$, Θ^θ *un comparateur flou généralisé et soit* $\gamma \in$ *[0,1] un "seuil de compatibilité". Alors la Sélection Floue Généralisée réalisée dans R avec la condition induite par* Θ^θ *composée avec* $\widetilde{a}$ *et l'attribut* A_k, *qualifiée par* γ, $S(R; \Theta^\theta(A_k, \widetilde{a} \geq \gamma)$, *est une relation donnée par :*

$$S(R; \Theta^\theta(A_k, \widetilde{a}) \geq \gamma) = \begin{cases} \mathcal{H}_S = \{(A_1 : D_1[, C_1]), ..., (A_n : D_n[, C_n])\} & (11) \\ \mathcal{B}_S = \{(A_1 : d_{r1}[, c_{r1}]), ..., (A_k : d_{rk}[, c'_{rk}]), ..., (A_n : d_{rn}[, c_{rn}])\} \end{cases}$$

Avec $c'_{rk} = \Theta^\theta(\widetilde{d}_{rk}, \widetilde{a}) \geq \gamma$.
Où r = 1,...,m' et m' est la cardinalité de la sélection.

Jointure Floue Généralisée

Définition 2.4.9.
Soient R et R' deux relations floues généralisées qui ne sont pas nécessairement différentes, données par les équations (1) et (2). Soit Θ^θ un comparateur flou généralisé *et soit* $\gamma \in$ *[0,1]* un seuil de compatibilité *et par suite* la Jointure Flou Généralisée *menée dans* $R \times R'$ *avec la condition* Θ^θ *produite dans les attributs A pour R et dans A' pour R', qualifiée par* γ, $\infty(R \times R'; \Theta^\theta(A_i, A'_i) > \gamma)$ *est donnée par :*

$$R^\infty = \begin{cases} \mathcal{H}^\infty = \mathcal{H} \times \mathcal{H}' & (12) \\ \mathcal{B}^\infty = \{(A_1 : \widetilde{d}_{r1}[, c_{r1}]), ..., (A_i : d_{ri}[, c''_{ri}]), ..., (A'_{i'} : \widetilde{d}'_{ri'}[, c"_{ri'}]), ..., (A'_{n"} : \widetilde{d}'_{rn}[, c'_{rn"}])\} \end{cases}$$

Avec $n" = n = n'$, *où r est la cardinalité de la jointure et*

$$c"_{ri} = c"_{ri'} = \Theta^\theta(d_{ri}, d_{ri'}) \geq \gamma.$$

La Jointure Floue Généralisée peut se comprendre comme un genre de Sélection Floue Généralisée appliquée sur le Produit Cartésien Flou Généralisé de deux relations, et son résultat est la variation des degrés de la compatibilité des attributs sur lesquels la jointure est exécutée en supprimant les tuples qui ne présentent pas un "degré de jointure" supérieur ou égal au seuil établi.

2.5 Langages de manipulation des bases de données floues

La manipulation de toute BD nécessite obligatoirement de définir un langage permettant, à partir des commandes qu'il offre, de répondre à toutes les informations demandées. Les langages qui ont introduit les concepts flous dans les BDF sont : SQLf et FSQL.

2.5.1 SQLf

A la base du SQL, le "standard database query language", qui offre des constructions simples et puissantes en BDR, le SQLf a été proposé par Bosc en 1995 pour remédier aux problèmes posés par le langage SQL dans les requêtes flexibles [BOS 95].

Bloc de base

La structure en trois clauses : "select", "from" et "where" du bloc de base SQL est conservée dans SQLf. La clause "from" ne subit aucune différence et les différences concernent essentiellement deux points : le calibrage du résultat qui peut se traduire par un nombre de réponses désirées (noté n) ou un seuil qualitatif (noté t) ou les deux et la nature des conditions autorisés. Par conséquent, en SQLf, la formulation du bloc de base est l'expression (E1) :

`Select [distinct][n|t|n,t] <attributs> from <relations> where <condition floue>`

où <condition floue> peut contenir à la fois des conditions booléennes et graduelles reliées par des connecteurs. La clause "where" peut contenir plusieurs constituants d'une condition graduelle à savoir des prédicats de base correspondants à des adjectifs ou des comparaisons utilisant un opérateur relationnel imprécis (environ, beaucoup plus ... que, etc.), des prédicats modifiés grâce aux modificateurs linguistiques (très, relativement,etc.) et des combinaisons de prédicats par l'emploie de connecteurs binaires ou n-aires.

Comme dans le cas usuel, l'expression (E1) est interprétée comme la restriction du produit cartésien des relations impliquées, suivie d'une projection sur les attributs mentionnés, puis du calibrage du nombre de réponses. La projection n'élimine pas les doublets éventuels (comme en SQL) mais donne les éléments ayant degré le plus élevé.

En SQLf, un bloc multi-relation combine une projection, des restrictions et des

jointures algébriques. Ainsi, une jointure floue apparut dans la requête :
`select distinct R.A, S.B from R, S where` f_{C_R} `and` f_{C_S} `and (R.C` θ `S.D)`
où (R.C θ S.D) est la condition de jointure floue (R.C "à peu prés égal à" S.D, R.C " très supérieur à " S.D,...), f_{C_R} (resp. f_{C_S}) étant une expression de sélection sur R (resp. S). Un couple de valeur (a,b) de la relation résultante Rf a le degré :
$\mu_{Rf}(a,b) = max_{r \in R \text{ et } r.A=a \text{ et } s \in S \text{ et } s.B=s} min(\mu_{f_{C_R}}(r), \mu_{f_{C_S}}(s), \mu_\theta(r.C, s.D))$

Exemple 2.5.1.
Soit la relation EMPLOYÉ (matricule, prénom, nom, salaire, âge, adresse), où en plus du prédicat "bien-payé", nous avons ajouté le prédicat "plus jeunes" et soit la requête : "trouver les employés "bien-payé" de Tunis et qui sont "les plus jeunes".

En SQLf, la requête s'écrit :
select matricule, nom from EMPLOYÉ where adresse="Tunis"
and salaire="bien-payé" and âge="plus_jeune"

matricule	num_d	nom	prénom	salaire	âge	adresse
105	02	Guesmi	Ramzi	500 (0.6)	31 (0.35)	Tunis
106	03	Daghbaji	Brahim	450 (0.4)	29 (0.3)	Gabes
107	04	Ben Cheikh	Imen	400 (0.2)	26 (0.5)	Tunis
108	02	Mallouli	Mouna	400 (0.2)	28 (0.4)	Tunis
112	02	Ben Hssine	Med Ali	700 (0.6)	24 (0.6)	Tunis

TAB. 2.10 – la relation EMPLOYÉ avec les prédicats "bien payés" et "plus jeunes"

*Et par suite nous aboutissons au classement suivant, un classement qui se base sur les fonctions d'appartenance des prédicats graduels "*bien payé*" et "*plus jeunes*".*

matricule	prénom	Fonction d'appartenance
112	Med Ali	min(1, 0.6, 0.6)
105	Ramzi	min(1, 0.6, 0.35)
107	Imen	min(1, 0.2, 0.5)
108	Mouna	min(1, 0.2, 0.4)

TAB. 2.11 – Classement des résultats retournés par la requête

Prédicat avec bloc imbriqué

Certains prédicats SQL sont construits à l'aide d'un bloc appelé imbriqué ou sous-requête. Ainsi, l'opérateur "in" traduit l'appartenance de la valeur d'attribut du tuple courant à l'ensemble de valeurs retourné par une sous-requête. De même, en SQLf, un bloc construit une relation floue. L'opérateur "in" est étendu donc pour exprimer l'appartenance à un ensemble flou et le prédicat "A in (select B, ...)" est défini par : $\mu_{AinSR}(a) = max_{b \in support(SR) \wedge b=a} \quad \mu_{SR}(b)$ où SR est l'ensemble flou retourné par la sous requête. La non-appartenance est définie par :
$\mu_{\text{A not in SR}}(a) = 1 - \mu_{\text{A in SR}}(a) = 1 - max_{b \in support(SR) \wedge b=a} \quad \mu_{SR}(b)$
Un second type d'imbrication repose sur l'emploi du mot-clé "exists" dans un prédicat du type "exists (select...)" qui rend vrai si l'ensemble retourné par la sous-requête contient au moins un élément (est non vide) et "faux" sinon. Dans le cas où l'ensemble construit par la sous-requête est flou, l'extension de cette opération pose la question de déterminer dans quelle mesure un ensemble flou est vide. L'interprétation retenue se fond sur l'α-coupe la plus élevée non vide de la sous-requête, soit : $\mu_{exists}(SR) = max_{x \in support(SR)} \quad \mu_{SR}(x)$.

Exemple 2.5.2.
Soient les relations EMPLOYÉ : EMP (matricule, prénom, nom, salaire, âge, adresse) et DÉPARTEMENT : DEP (num_ d, budget, nbre_ employé, local), et soit les requêtes suivantesc :

- *"Trouver les employés (5meilleurs) bien payés travaillant dans un département à assez grand nombre d'employés". Cette requête peut s'écrire selon plusieurs formes :*
  1. select 5 distinct matricule, nom from EMP, DEP where EMP.salaire = "bien-payé" and EMP.num_ d=DEP.num_ d and DEP.nbre_ employé="élevé"
  2. select 5 matricule, nom from EMP where salaire = "bien-payé" and num_ d in (select num_ d from DEP where nbre_ employé = "élevé")
  3. select 5 matricule, nom from EMP where salaire = "bien-payé" and exists (select num_ d from DEP where num_ d = EMP.num_ d and nbre_ employé = "élevé")
- *"Trouver les départements qui n'ont aucun employé jeune".*
  select nom, num_ d from DEP where not exists (select num_ d from EMP where âge = "jeune" and num_ d = DEP.num_ p)

Il existe d'autres possibilités d'imbrications dans SQL (en particuliers quantificateurs

et comparaison scalaire) qui ont été étendues dans SQLf [BOS 95].

Partitionnement des relations et division

Partitionnement

La notion de groupement de tuples existant en SQL est conservée en SQLf où les conditions permettant la sélection d'ensembles de n-uplets sont étendues à des conditions graduelles. Tout comme en SQL, les fonctions d'agrégat (min,sum,...) sont utilisées comme constituant d'une condition ensembliste booléenne. En SQLf, les fonctions d'agrégat interviennent en paramètres de prédicats graduels dans le cadre de requêtes du type :
`select A, liste-ag from R where bc group by A`
`having` $fc_1(ag_1(B_1))$ `and/or ... and/or` $fc_p(ag_p(B_p))$
où liste-ag représente une liste optionnelle d'agrégats et bc désigne une condition booléenne car une fonction d'agrégat ne peut s'appliquer qu'à un ensemble usuel.

Exemple 2.5.3.
Soit la requête "trouver les départements (5meilleurs) avec leurs nombre d'employés mâle dont la moyenne des salaires est aux environs de 500 dt" s'écrit :
select 5 num_d, count(*) from EMPLOYÉ where sexe="mâle"
group by num_d having around (avg(salaire), 500)

Expression de la division étendue

L'expression de la division dans SQLf est exprimée comme suit [BOS 95] :
`select X from R where` fc_R `group by X`
`having set(A) contains (select B from S where` fc_S`)`
où "contains" représente une inclusion graduelle fondée sur l'implication souhaitée : Dienes, Godel, Goguen ou Lukasiewiez.

Exemple 2.5.4.
Soit la requête de l'exemple 8 : "trouver les employés auxquels nous avons affecté pendant de longues heures tous les projets à gros budget". En SQLf, cette requête s'écrit :
Select *matricule* from *TRAVAILLE_DANS* where *nbre_heure="élevé"* group by *matricule* having *set (num_p)* contains *(*select *num_p* from *PROJET* where *budget="gros").*

Cette requête nécessite le calcul des degrés d'inclusion de {0.6/01, 0.4/02, 1/03} dans {1/01, 0.3/02, 0.9/03}(resp {1/01, 0.2/02, 1/03}) pour 105 (resp 107) qui conduit aux mêmes résultats que précédemment ("contains" de Kleene_ Dienes) : Car l'inclusion délivre un degré et dérive de l'inclusion usuelle vue comme l'implication : $A \subseteq B(\forall x, \mu_A(x) \Rightarrow \mu_B(x))$ *par :* $deg(A \subseteq B) = min_{x \in X} \quad \mu_A(x) \longrightarrow \mu_B(x)$ *où* $\longrightarrow$ *désigne une implication floue, et puisqu'il existe plusieurs implications floues, nous prenons à titre d'exemple et à titre de comparaison avec les résultats obtenus dans l'exemple 7 celle de Kleene_ Dienes* $a \longrightarrow b = max(1\text{-}a,b)$ *et par suite*
$deg(PROJET \subseteq TRAVAILLE\_DANS) = min(max(1\text{-}\mu_{PROJET}(x), \mu_{TRAVAILLE\_DANS}(x)))$
$= min(max(1\text{-}0.6,1), max(1\text{-}0.4,0.3), max(1\text{-}1,0.9)) = 0.6$ *pour 105*
$= min(max(1\text{-}0.6,1), max(1\text{-}0.4,0.2), max(1\text{-}1,0.6)) = 0.6$ *pour 107*
c-à-d, pour l'implication floue de Kleene_ Dienes, nous aboutissons à : {0.6/105,0.6/107}.

2.5.2 FSQL

Le langage FSQL [MED 94b][GAL 98b] est une extension du SQL afin de permettre des requêtes flexibles. Cela veut dire que toutes les requêtes valides dans SQL sont aussi valides dans FSQL. Cette flexibilité vient du fait que nous pouvons introduire des attributs flous, des constantes floues, des comparateurs flous, des qualificateurs et quantificateurs flous,...

Syntaxe et sémantique du langage FSQL

Les commandes de base dans ce modèle se divisent en deux inclinaisons : le langage de manipulation de données (LMD) et le langage de définition de données (LDD).

Le LMD de FSQL

Dans le LMD, la commande la plus utilisée et la plus complexe est la commande de consultation de données, SELECT, bien que les autres commandes tels que INSERT, DELETE et UPDATE soient aussi intéressants.

Nouveauté dans le SELECT flou

La commande SELECT est une commande forte, complexe et flexible, très facile à utiliser dans des consultations simples et assez délicate dans des consultations complexes. Quelquefois, pour simplifier l'écriture et la compréhension des consultations compliquées, nous pouvons utiliser des vues intermédiaires comme sous-consultation

de la BD. Le langage FSQL incorpore également des nouveautés pour autoriser le traitement des informations imprécises [Gal 98a]. Fondamentalement, les extensions faites à cette commande sont les suivantes :

– **Les étiquettes linguistiques** : Si un attribut est accessible à un traitement flou, alors des étiquettes linguistiques peuvent être définies sur lui, précédées par le symbole $ pour les distinguer facilement. Nous pouvons distinguer deux types d'étiquettes :

1. étiquettes pour des attributs dans un domaine flou ordonné : chaque étiquette de ce type associe une distribution de possibilité trapézoïdale, comme il est indiqué dans la figure 2.1. Ainsi, nous pouvons définir l'étiquette $jeune pour l'attribut âge d'une personne avec les valeurs $\alpha = 16, \beta = 22, \gamma = 32 et \delta = 32$ (en nombre d'année).

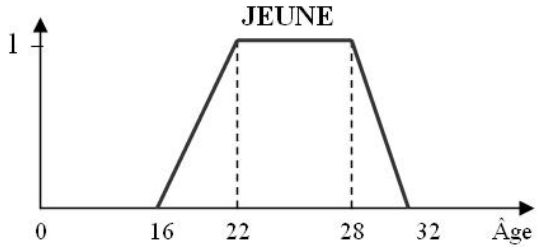

FIG. 2.1 – Exemple d'étiquette linguistique pour l'attribut Âge

2. Étiquettes pour des attributs dans un domaine non ordonné (type 1,3 et 5 de la table 11). Une relation de similitude est définie entre deux étiquettes de ce domaine. Le degré de similitude est dans l'intervalle [0,1]. Par exemple, pour la couleur des yeux d'une personne, nous pouvons attribuer un degré de 0.6 pour les étiquettes $châtain et $vert.

– **Les comparateurs flous** : En plus des comparateurs usuels (=,>, etc.), SQLF inclut les comparateurs flous présentés dans Table 2.12. Leurs définitions sont détaillées dans la section suivante. Comme dans SQL, les comparateurs flous sont capables de comparer une colonne (ou attribut) à une constante ou deux colonnes de même type.

Les comparateurs de nécessité sont plus restrictifs que les comparateurs de possibilité, et donc leur degré de satisfaction est toujours plus petit que le degré de satisfaction obtenu par leur comparateur de possibilité correspondant. En général, les comparateurs demandent que la condition soit exécutée à un certain degré, même si ce n'est pas d'une manière complète, alors que les comparateurs

Comparateurs flous pour		Signification
La Possibilité	La Nécessité	
FEQ	NFEQ	Egalité Floue (Fuzzy EQual) (Possiblement/Nécéssairement)
FGT	NFGT	Plus Grand que floue (Fuzzy Greater Than)
FGEQ	NFGEQ	Plus Grand ou Egal que floue (Fuzzy Greater or Equal)
FLT	NFLT	Plus petit que floue (Fuzzy Less Than)
FLEQ	NFLEQ	Plus Petit ou Egal floue (Fuzzy Less or equal)
MGT	NMGT	Beaucoup Plus Grand que (Much Greater Than)
MLT	NMLT	Beaucoup Plus Petit que (Much Greater Than)

TAB. 2.12 – Les comparateurs flous de FSQL

de possibilité mesurent à quelle valeur (degré) c'est possible que la condition soit exécutée.

- **Seuil de réalisation (d'exécution) (γ)** : pour chaque condition simple, un seuil minimum de réalisation (1 par défaut) doit être établi, avec le format général suivant : <condition> THOLD γ ;
  indiquant que la condition doit être accomplie à un degré minimum $\in$ [0,1]. Le mot réservé THOLD est facultatif et peut être substitué par un comparateur classique ($=$, $\leq$, etc.), donc modifiant la signification de la requête. Le mot THOLD est équivalent au comparateur $\geq$.
- **La fonction CDEG** (<attribut>) : Cette fonction montre une colonne avec les degrés de réalisation de l'attribut spécifié entre parenthèse. S'il y a des opérateurs logiques dans la condition, le calcul du degré de compatibilité se fait en utilisant le minimum (norme triangulaire) et le maximum (co-norme triangulaire) comme il est montré dans la table 13, bien que l'utilisateur puisse changer facilement cette fonction [GAL 98b].

<condition>	CDEG(<condition>)
<cond1> AND <cond2>	min (CDEG(<cond1>), CDEG(<cond2>))
<cond1> OR <cond2>	max (CDEG(<cond1>,CDEG(<cond2>))
NOT <cond1>	1- CDEG(<cond1>)

TAB. 2.13 – Constantes floues dans FSQL

- **Les constantes floues** : FSQL a introduit de nouvelles constantes basées sur le concept flou. Les valeurs de ces constantes sont entreposées dans la BMCF qui sera définie dans le chapitre suivant.

Constantes floues	**Signification**
UNKNOWN UNDEFINED NULL	Valeur inconnu mais l'attribut est applicable (type 8 de la table 2.7) L'attribut n'est pas applicable ou n'a pas de sens (type 9 de la table 2.7) Une ignorance total : nous ne connaissons rien sur lui (type 10 de la table 2.7)
$[a,b,c,d] $label $[n,m] #n	Trapèze flou (a≤b≤c≤d) Etiquette linguistique : trapézoïdale ou scalaire (définie dans la BMCF) Intervalle "entre n et m" (a = b = n et c = d = m) Valeur floue "Approximative" (b = c = n et n-a = d-n = marge)

TAB. 2.14 – Les constantes floues de FSQL

– **La condition IS** : cette condition est utilisée avec le même format que dans le standard SQL, mais elle inclut les trois premiers types de constantes floues définies dans la table 2.14.
<Fuzzy_Attribute> IS [NOT] (UNKNOWN|UNDEFINED|NULL).
Si l'attribut n'est pas flou et la constante est NULLE, alors la signification d'une telle constante est différente et prend la signification donnée par le DBMS.
– **Les qualificateurs flous** : Dans FSQL, les quantificateurs flous sont de deux natures, absolus et relatifs [GAL 00]. Ils peuvent s'écrire avec l'une des deux formes suivantes :
$Quantificateur FUZZY[$\rho$] (condition_floue) THOLDτ
$Quantificateur FUZZY [$\rho$] (condition_floue1) ARE (condition_floue2) THOLD τ
Où $Quantificateur est un quantificateur (absolu ou relatif) précédé par le symbole $ pour le distinguer des autres identificateurs.
– **Les commentaires** : FSQL permet d'incorporer des commentaires dans les requêtes, afin qu'ils ne seront pas tenus dans l'analyse et dans l'exécution.

Exemple 2.5.5.

Soit la requête "trouver les jeunes employés (avec un seuil de 0.75) qui habitent à Tunis et ayant un salaire supérieur ou égal à la distribution trapézoïdale [300,500,700,100]". Cette requête s'écrit :

select nom, ***CDEG****(âge), salaire from EMPLOYÉ where âge* ***FEQ*** *$jeune 0.75 and salaire* ***FGEQ*** *$[300,500,700,1000] and adresse = 'Tunis';*

Nous remarquons que

– *Le seuil minimum est mis à 0.75 et le mot THOLD n'apparaît pas parce qu'il est facultatif. Ainsi, dans la table résultante, la colonne CDEG(âge) prend les*

valeurs dans l'intervalle [0.75,1].

- *Puisque nous utilisons le comparateur FGEQ, le γ et le δ du trapézoïde ne seront pas utilisés, c.-à-d., si le salaire de l'employé est égal ou dépasse 500, le degré sera 1. Naturellement, s'il est égal ou est moins de 300, le degré sera zéro.*
- *Si un employé a un salaire dont la distribution de possibilité est [50,150,300,500], son degré de réalisation sera 0.5 et il n'apparaîtra pas dans le dernier résultat parce que le minimum a été mis à 0.75.*

Comparateurs flous

Dans la littérature, plusieurs méthodes de comparaison des nombres flous ont été introduites. Elles peuvent être classées en deux catégories : celles qui utilisent une fonction de l'ensemble des nombres flous à un ensemble ordonné et celles qui utilisent des relations floues [GAL 99b].

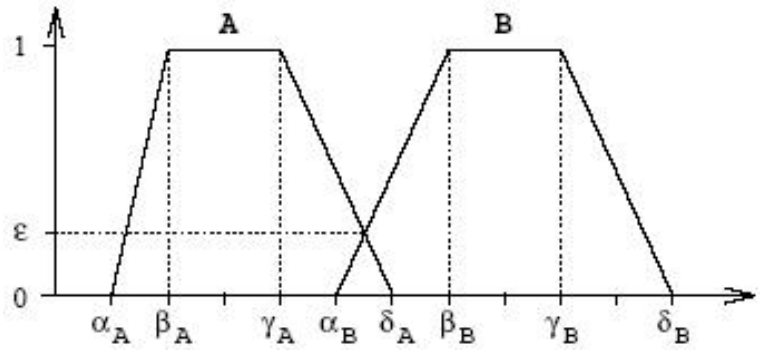

FIG. 2.2 – Comparaison de deux distributions de possibilités trapézoïdales

Supposons que nous voulons comparer deux distributions de possibilités A=$[\alpha_A, \beta_A, \gamma_A, \delta_A]$ et B=$[\alpha_B, \beta_B, \gamma_B, \delta_B]$ comme par exemple celles représentées dans la figure 2.2. Nous devons utiliser la fonction CDEG pour exprimer le degré de compatibilité d'une comparaison floue. Le résultat de cette comparaison doit dépendre, naturellement, du comparateur utilisé représenté dans la table 2.15.

Quantificateurs flous

Comme dans le modèle classique, FSQL utilise des quantificateurs qui peuvent être absolus ou relatifs [GAL 00].

Les quantificateurs absolus peuvent résoudre des questions sur le nombre total de tuples résultant d'une certaine consultation, en disant (ou répondant à) si ce nombre

F_Comp (comparateur flou)	L'opérateur possibiliste CDEG(A F_Comp B)=	L'opérateur de nécessité CDEG(A F_Comp B)=
FEQ / NFEQ	$= \sup_{d \in U} \min(A(d), B(d))$ où U est le domaine de A et B, A(d) est le degré de possibilité pour d∈U dans la distribution A.	$= \inf_{d \in U} \max(A(d), B(d))$ où U est le domaine de A et B, A(d) est le degré de possibilité pour d∈U dans la distribution A
FDIF / NFDIF	= 1 CDEG(A NFEQ B)	= 1 CDEG(A FEQ B)
FGT/ NFGT	$= \begin{cases} 1 & \text{si } \gamma_A \geq \delta_B \\ \frac{\delta_A - \gamma_B}{(\delta_B - \gamma_B) - (\gamma_A - \delta_A)} & \text{si } \gamma_A < \delta_B \text{ et } \delta_A > \gamma_B \\ 0 & \text{sinon} \end{cases}$	$= \begin{cases} 1 & \text{si } \alpha_A \geq \delta_B \\ \frac{\beta_A - \gamma_B}{(\delta_B - \gamma_B) - (\alpha_A - \beta_A)} & \text{si } \alpha_A < \delta_B \text{ et } \beta_A > \gamma_B \\ 0 & \text{sinon} \end{cases}$
FGEQ / NFGEQ	$= \begin{cases} 1 & \text{si } \gamma_A \geq \beta_B \\ \frac{\delta_A - \alpha_B}{(\beta_B - \alpha_B) - (\gamma_A - \delta_A)} & \text{si } \gamma_A < \beta_B \text{ et } \delta_A > \alpha_B \\ 0 & \text{sinon} \end{cases}$	$= \begin{cases} 1 & \text{si } \alpha_A \geq \beta_B \\ \frac{\beta_A - \alpha_B}{(\beta_B - \alpha_B) - (\alpha_A - \beta_A)} & \text{si } \alpha_A < \beta_B \text{ et } \beta_A > \alpha_B \\ 0 & \text{sinon} \end{cases}$
FLT / NFLT	$= \begin{cases} 1 & \text{si } \beta_A \leq \alpha_B \\ \frac{\alpha_A - \beta_B}{(\alpha_B - \beta_B) - (\beta_A - \alpha_A)} & \text{si } \beta_A > \alpha_B \text{ et } \alpha_A < \beta_B \\ 0 & \text{sinon} \end{cases}$	$= \begin{cases} 1 & \text{si } \delta_A \leq \alpha_B \\ \frac{\gamma_A - \beta_B}{(\alpha_B - \beta_B) - (\delta_A - \gamma_A)} & \text{si } \delta_A > \alpha_B \text{ et } \gamma_A < \beta_B \\ 0 & \text{sinon} \end{cases}$
FLEQ / NFLEQ	$= \begin{cases} 1 & \text{si } \beta_A \leq \gamma_B \\ \frac{\delta_B - \alpha_A}{(\beta_A - \alpha_A) - (\gamma_B - \delta_B)} & \text{si } \beta_A > \gamma_B \text{ et } \alpha_A < \delta_B \\ 0 & \text{sinon} \end{cases}$	$= \begin{cases} 1 & \text{si } \alpha_A \leq \gamma_B \\ \frac{\gamma_A - \delta_B}{(\gamma_B - \delta_B) - (\delta_A - \gamma_A)} & \text{si } \delta_A > \gamma_B \text{ et } \gamma_A < \delta_B \\ 0 & \text{sinon} \end{cases}$
MGT / NMGT	$= \begin{cases} 1 & \text{si } \gamma_A \geq \delta_B + M \\ \frac{\gamma_B + M - \delta_A}{(\beta_A - \alpha_A) - (\gamma_B - \delta_B)} & \text{si } \gamma_A < \gamma_B + M \text{ et } \delta_A > \gamma_B + M \\ 0 & \text{sinon} \end{cases}$	$= \begin{cases} 1 & \text{si } \gamma_A \geq \delta_B + M \\ \frac{\gamma_B + M - \delta_A}{(\alpha_A - \beta_A) - (\delta_B - \gamma_B)} & \text{si } \alpha_A < \gamma_B + M \text{ et } \beta_A > \gamma_B + M \\ 0 & \text{sinon} \end{cases}$
MLT / NMLT	$= \begin{cases} 1 & \text{si } \beta_A \leq \alpha_B - M \\ \frac{\beta_B - M - \alpha_A}{(\beta_A - \alpha_A) - (\alpha_B - \beta_B)} & \text{si } \beta_A > \alpha_B - M \text{ et } \alpha_A < \beta_B - M \\ 0 & \text{sinon} \end{cases}$	$= \begin{cases} 1 & \text{si } \delta_A \leq \alpha_B - M \\ \frac{\beta_B - M - \gamma_A}{(\delta_A - \gamma_A) - (\alpha_B - \beta_B)} & \text{si } \delta_A > \alpha_B - M \text{ et } \gamma_A < \beta_B - M \\ 0 & \text{sinon} \end{cases}$

TAB. 2.15 – Définition des comparateurs flous dans la famille de Possibilité/Nécessité utilisant deux distributions de possibilité trapézoïdales : A et B de la Figue 2.2

est "*grand*", "*petit*", "*beaucoup*", "*un peu*", "*beaucoup*",... Dans ce cas, le quantificateur dépend d'une seule quantité.

Les quantificateurs relatifs peuvent résoudre des questions dans lesquelles la vérité du quantificateur dépend de deux quantités. Ce type de quantificateurs est utilisé dans les expressions comme "*la majorité*", "*la minorité*", "*approximativement la moitié*",...

Les quantificateurs flous absolus se représentent comme des distributions de possibilité trapézoïdale dans l'intervalle [0,+ ∞] et les relatifs comme des distributions de possibilité dans l'intervalle [0,1]. C'est-à-dire, un quantificateur Q est représenté comme une fonction Q dont le domaine dépend de son type, si c'est absolu ou relatif :

$$Q_{abs} : R^{+} \longrightarrow [0, 1]$$

$$Q_{rel} : [0, 1] \longrightarrow [0, 1]$$

Les quantificateurs flous (absolu ou relatif) peuvent être représentés comme des distributions de possibilité trapézoïdales. En plus, les quantificateurs peuvent être utilisés dans deux familles d'expressions [GAL 00], où Q est le Quantificateur et A et B sont des prédicats (condition flous ou non) :

1. "Q éléments de l'ensemble X qui complètent A" : par exemple, "*la majorité* des joueurs de l'équipe X sont *très bons*".
2. "Q éléments de l'ensemble X qui complètent B complètent aussi A" : par exemple, "*la majorité* des joueurs de l'équipe X qui sont *grands* sont aussi *très bons*".

Exemple 2.5.6.

Sélectionner les équipes du basket-ball dans lesquelles la majorité (avec un seuil minimum de 0.5) de leurs joueurs sont grands et aussi très bons (avec les seuils 0.75) :

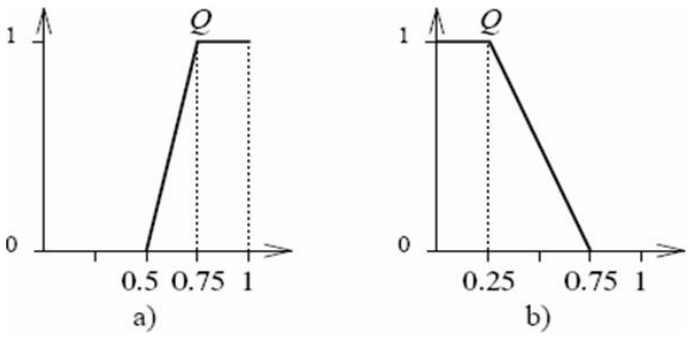

FIG. 2.3 – Quantificateurs flous relatifs

select Equipe, CDEG(*)
from Joueurs
group by Equipe
having $la_majorité *(hauteur FEQ* $grand *0.75)*
are (Qualité FEQ $très-bonne *0.75) THOLD 0.5 ;*
La majorité est un quantificateur relatif qui sera défini dans la BMCF[2] *avec la forme, par exemple de la figure 2.3a.*

2.6 Les commandes LDD de FSQL

Le langage FSQL étend déjà les commandes existantes dans SQL, mais il incorpore aussi d'autres nouvelles commandes. L'architecture oùcle langage FSQL va être exécuté distingue dans la définition des données trois types d'attributs. Cette classification est inspirée du modèle GEFRED, pour cela, et pour le moment, nous disons qu'il y a trois types d'attributs qui seront détaillés dans le chapitre suivant. Les principales commandes LDD de FSQL sont [GAL 99a] : **CREATE TABLE** : C'est la phrase qui inclut plus de nouvelles (voir exemple) :

1. *Type de Données floues* : Il peut être l'un des 3 attributs flous type1 (**FTYPE1** ou **CRISP**), type2 (**FTYPE2** ou **POSSIBILISTIC)** et type3 (**FTYPE3** ou **SCALAR**).
2. *Sous consultation, constantes, conditions et expressions floues.*
3. *Restrictions de colonne* : Dans cette partie, plusieurs restrictions peuvent être imposés, ainsi que ceux qui existent déjà pour les attributs classiques (NOT NULL, NOT UNDEFINED, NOT UNKNOWN, NOT LABEL, NOT CRISP, NOT TRAPEZOID, NOT INTERVAL, NOT APPROX, ONLY LABEL, ONLY LABEL OR UNKNOWN, CHECK).

CREATE VIEW : Cette commande permet de créer des vues avec sous consultations floues. Le format est le même comme dans SQL mais nous pouvons admettre des sous consultations floues (SELECT flou).

Exemple 2.6.1.
La commande suivante crée une vue avec les données des employés jeunes (avec un degré supérieur à 0.8) et gagnent plus que les employés bien-payés *(avec un degré minimal 0.6) :*

[2]la BMCF (Base de Méta Connaissances Floues) représente la partie de la BDF qui permet de sauvegarder les informations floues (section 3.5)

CREATE VIEW EMPLOYÉ_JEUNE AS
SELECT matricule, Nom, Âge, CDEG(Âge) FROM EMPLOYÉ
WHERE Âge FEQ $jeune 0.8 AND
Salaire FGT $élevé 0.6
READ ONLY ;

CREATE LABEL : C'est une nouvelle commande, exclusive de FSQL, qui sert à créer (autoriser) des étiquettes dans le domaine d'un attribut flou concret. Elle possède deux formats, dont le plus utilisé est :
CREATE LABEL nom_étiquette ON [schema.]table.attribut VALUES alfa, beta, gamma, delta ;

Exemple 2.6.2.
CREATE LABEL jeune ON EMPLOYÉ.Âge VALUES 18,22,30,35 ;

CREATE NEARNESS : C'est une nouvelle commande, spécifique de FSQL, qui sert à créer des étiquettes pour les attributs flous Type 3. Avec cette commande toutes les étiquettes qui vont appartenir à l'attribut devraient être définies avec le rapport (degré) de ressemblance entre elles. Son format est :
CREATE NEARNESS ON [schema.]table.attribut LABEL liste_des_etiquettes VALUES liste_des_similitudes ;

Exemple 2.6.3.
CREATE NEARNESS ON EMPLOYÉ.Rendement
LABELS Mauvais, Régulier, Bon, Excellent
VALUES .8, .5, .1,
.7, .5,
.8 ;

ALTER TABLE, ALTER VIEW : Ce sont des commandes pour modifier une table/vue déjà créée. Son format est semblable à celui de SQL mais elle permet de définir des types de données et des restrictions floues.
ALTER LABEL : C'est une commande exclusive à FSQL pour modifier une étiquette d'un attribut flou de type 1 ou 2. Son format est semblable à celui de CREATE LABEL en changeant le mot réservé CREATE par ALTER .
ALTER NEARNESS : C'est une commande exclusive de FSQL pour modifier les étiquettes d'un attribut flou Type 3 et/ou leurs degrés de ressemblance.
DROP TABLE, DROP VIEW : C'est une commande pour effacer une table/vue (flou ou non). Son format est identique à celle dans SQL.

DROP LABEL : C'est une commande exclusive de FSQL qui sert à effacer une ou toutes les étiquettes définie(s) sur un attribut flou de type 1 ou 2.
DROP NEARNESS : C'est une commande exclusive de FSQL qui sert à effacer les étiquettes floues de type 3.

2.7 Conclusion

Les informations que nous pouvons disposer sont souvent incomplètes, imprécises ou émanent de sources hétérogènes. Ceci ne doit évidemment pas être un obstacle à leur gestion par des systèmes d'informations avancés. Or, une limite de ces systèmes est qu'ils ne peuvent pas traiter ce genre d'informations. Pour accomplir une méthode permettant de résoudre ces problèmes, le langage de manipulation de BD, SQL a été étendu à FSQL et SQLf (selon l'algèbre utilisée) en adjoignant des concepts flous. Nous allons se baser dans ce qui suit, sur le modèle GEFRED et également sur le langage FSQL pour présenter les autres concepts des BDF.

Chapitre 3

Implémentation du modèle GEFRED

3.1 Introduction

La mise en place de tout système nécessite une étude bien détaillée pour assurer son bon fonctionnement. Pour cette raison, il est indispensable de définir la communication entre les différents composants de ce système.
L'architecture proposée par Medina pour une BDRF se compose d'un ensemble de composants qui interagissent entre eux afin d'assurer le fonctionnement adéquat de cette base. L'objectif de cette architecture est de pouvoir stocker et traiter des informations imprécises, floues, et par la suite formaliser une méthode pour y faire. En effet, il est absolument indispensable de définir, d'une part la manière d'entreposer ces nouveaux types d'information, et d'autre part, le canal de communication avec la BDRF. Ce canal de communication permet aux utilisateurs, soit directement ou à travers des applications, d'extraire et d'entreposer leurs données imprécises.
Nous présentons dans ce chapitre les principaux composants qui constituent cette architecture.

3.2 Implémentation de FIRST

Dans [MED 94b], il a été exposé un module pour étendre les capacités d'un SGBDR classique pour qu'il puisse représenter et manipuler des informations "imprécises". Ce module est appelé FIRST (A Fuzzy Interface for Relational Systems), il utilise GEFRED comme modèle théorique et les ressources du modèle relationnel classique pour représenter ce type d'information.

3.2.1 Schéma général de FIRST

La figure 3.1 montre le schéma général de FIRST. Ses principales composantes sont :

- **SGBDR** : toutes les opérations conçues pour l'extension floue apportée par cette implémentation se traduisent par une demande au SGBD host (en générale en SQL). Ces demandes se réalisent en employant le langage SQL ou FSQL selon le type des conditions et des attributs. Nous verrons que les requêtes FSQL sont procédées par le serveur FSQL.
- **BD** : Elle sauvegarde toute information dans un format relationnel. La seule différence est que cette BD autorise le stockage d'information floue dans ces tables.
- **Base de Méta Connaissances Floues (BMCF)** : Elle étend le dictionnaire ou le catalogue du SGBD pour sauvegarder les informations de la BDRF dans un format relationnel (figure 3.6). Elle stocke les attributs qui admettent un traitement flou et les informations de chacun d'eux selon leur type.
- **Serveur FSQL** : Son objectif est d'extraire les requêtes écrites avec le langage FSQL et les traduire dans un langage compris par le SGBDR, le langage SQL. Pour effectuer cette traduction, il utilise l'information qui se trouve dans la BMCF.
- **Client FSQL, FSQL FORM Flou, Client Visuel** : Ce sont des programmes qui établissent une interface entre l'utilisateur et le serveur FSQL.

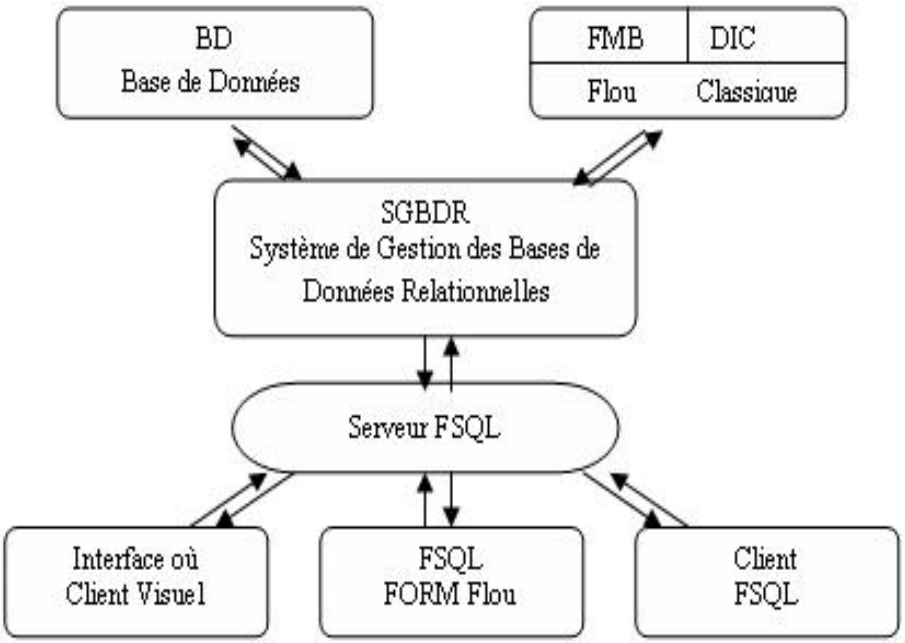

FIG. 3.1 – Schéma général de FIRST

Nous présentons dans la suite la façon avec laquelle les données floues sont représentées , et comment elles sont modélisées dans le SGBD Oracle.

3.2.2 Représentation des connaissances floues

Les éléments qui font partie du traitement imprécis peuvent avoir plusieurs représentations. De ce fait, une distribution de possibilité normalisée peut être représentée par différents types de fonctions (paraboles, hyperboles, ...). Nous utiliserons la représentation trapézoïdale comme celle représentée dans la figure 3.2.
Pour les différents types de données qui constituent la définition des domaines flous généralisés, classés dans la table 2.7, les critères de représentation suivants sont adoptés :

- **Données précises** : La représentation utilisée est celle proposée par le SGBDR. C'est-à-dire, les formats adoptés pour les chaînes alphanumériques, les valeurs numériques, les fichiers, l'heure,...
- **Données imprécises** : Les données de nature imprécise supportées par GEFRED peuvent être classées en deux groupes avec des représentations différentes pour chacun d'eux : sur un référentiel ordonné et sur un référentiel discret non ordonné. Le "référentiel" comprend le sous-domaine de l'attribut en question. Par exemple, l'attribut âge peut être considéré flou pour entreposer des valeurs comme "jeune" (voir la figure 2.1), "adulte", etc. mais le sous-domaine sur lequel les distributions de possibilité sont construites est ordonné et il correspond aux nombres d'années possibles de l'âge.

  1. **Données imprécises dans un référentiel ordonné**
     Ce groupe de données contient des distributions de possibilité sur des domaines continus ou discrets ayant une relation d'ordre existante. Le type 6 de la table 2.7 appartient à ce groupe. Chaque donnée de ce type lui est associée une fonction d'appartenance. Les représentations possibles pour ce type de données sont :
     - **Distribution de possibilité trapézoïdale** : Cette représentation détermine la fonction d'appartenance en utilisant les quatre paramètres $[\alpha, \beta, \gamma, \delta]$, tel qu'il est indiqué dans la figure 3.2
     - **Etiquettes linguistiques** : Les données exprimées au moyen d'une étiquette linguistique font la référence à un concept imprécis, qui, parfois, associe une distribution de possibilité. Par exemple, l'étiquette linguistique "jeune", peut être associée à une distribution de possibilité trapézoïdale

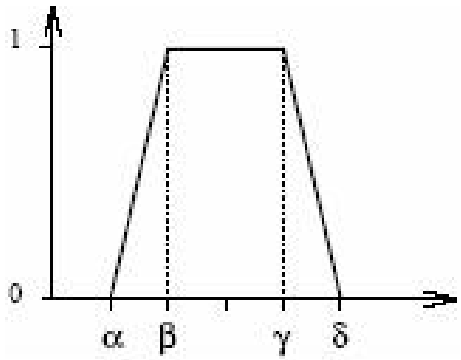

FIG. 3.2 – Distribution de possibilité trapézoïdale

comme le montre la figure 2.1.

– **Valeurs approximatives** : Elles représentent le concept imprécis "approximativement n" au moyen d'une valeur, appelée "marge". Elles utilisent la distribution de possibilité triangulaire (3.3) comme fonction d'appartenance.

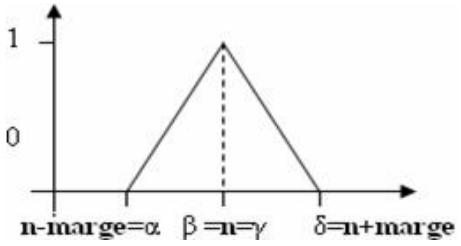

FIG. 3.3 – Distribution de possibilité pour "Approximativement n" (#n)

– **Intervalles de possibilité** : Ce sont un cas particulier de distributions de possibilité trapézoïdales dans lesquelles les inclinaisons des deux côtés du trapèze sont infinies et par conséquent toutes les valeurs parmi les deux extrémités sont les seules qui sont complètement possibles (possibilité 1), comme c'est montré dans la figure 3.4.

2. **Données imprécises dans un référentiel non ordonné**
   Ce groupe de données est construit sur des sous-domaines discrets non ordonnés dans lequel, des "relations (rapports) de ressemblance" existent entre les valeurs qui le constituent. Les différents types que nous pouvons représenter à l'intérieur de ce groupe sont :
   – **valeurs linguistiques simples** : Ce type est considéré comme une distribution de possibilité avec un seul couple de données dans lequel la valeur de possibilité est 1 : (1,d).

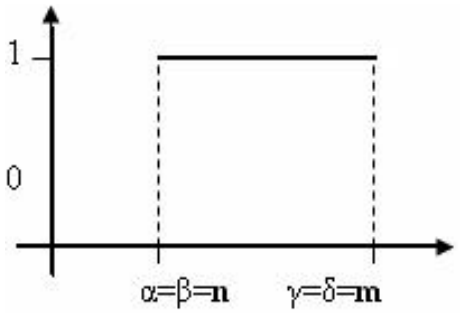

FIG. 3.4 – Distribution de possibilité pour l'intervalle [n,m]

– **Distribution de possibilité** : À un fait (donnée) imprécis(e) de ce type est associée une représentation formée par les valeurs du domaine du discours et les valeurs de possibilité qui leurs correspondent, $(p_1, d_1), , (p_n, d_n)$. Une des p_i doit être égale à 1 afin que la distribution soit normalisée.

Les données "imprécises" déjà décrites peuvent avoir en plus les trois valeurs suivantes :

3. **UNKNOWN (Inconnu, mais applicable)** : Un fait (donnée) de ce type reflète l'ignorance concernant la valeur prise par un attribut. Nous savons, cependant que l'attribut peut prendre quelques valeurs du domaine du discours. Par conséquent, le type UNKNOWN (INCONNU) est représenté au moyen de la distribution de possibilité, $\{1/u, u \in U\}$, où U est le sous-domaine de discours. La figure 3.5.a montre la représentation graphique cette distribution de possibilité qui prend la valeur 1 pour le sous-domaine entier.
4. **UNDEFINED (n'est pas applicable)** : Lorsqu'un attribut prend la valeur UNDEFINED (INDÉTERMINÉ), il reflète le fait qu'aucune des valeurs du domaine sur lequel c'est défini n'est applicable. Pour cela, il est représenté au moyen de la distribution de possibilité, $\{0/u, u \in U\}$, où U est le sous-domaine de discours. Sa distribution de possibilité est montrée dans la figure 3.5.b qui prend la valeur 0 pour le sous-domaine entier.

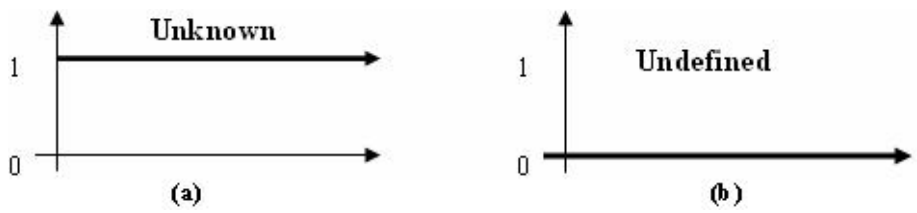

FIG. 3.5 – Distribution de possibilité pour les types Unknown et Undefined

5. **NULL (Ignorance absolue)** : cette valeur est appliquée lorsque aucune information sur l'attribut est présente, parce que nous ne la savons pas (UNKNOWN) ou parce que elle est non applicable (UNDEFINED). De ce fait, elle est représentée au moyen de l'ensemble $\{1/UNKNOWN, 1/UNDEFINED\}$.

3.3 FIRST sous Oracle

Le SGBD choisi dans l'implémentation de la BDF est Oracle, à cause de sa variabilité, sa grande extension et sa capacité de programmer des paquets (avec les fonctions et les procédures) interne au système, dans son propre langage, PL/SQL qui est avéré pour être complètement efficace. Bien sûr, cette architecture peut être implémentée avec d'autres SGBD [GAL 98a]. Dans [MED 95], il a été mentionné une description générale des niveaux sur lesquels sont implémentés les principaux modules de FIRST.

- **Niveau de BD** : La BD est cette collection de données persistantes qui constitue la représentation d'une partie de la connaissance de l'univers. Comme notre système s'intéresse à la représentation des connaissances imprécises, il doit mentionner la forme dans laquelle s'entrepose la BD. Par conséquent, il doit étendre la représentation des données pour héberger ce type d'information.
- **Niveau des classes du système** : Dans un SGBDR classique, il existe une partie du système dans laquelle se collectent toutes les informations que le gestionnaire a besoin de savoir à propos des données qu'il entrepose, "données sur les données" (quelquefois nommées "métas données"). Cette partie est la BMCF qui permet de représenter cette information au moyen des tables (relations) organisées suivant un schéma semblable à celui employé dans la BD.
- **Niveau du gestionnaire de FIRST** : Le gestionnaire possède des connaissances implémentées sur des opérations de nature imprécise pour les porter et les traiter. Le serveur FSQL est la partie du système qui contient les règles qui se chargent d'implémenter ces opérations.

3.4 Attributs flous

Dans le système déjà décrit, trois types d'attributs susceptibles de traitement imprécis existent. La classification adoptée est basée sur les approches de représentation et de traitement des données "imprécises". Ils sont classés selon le type du domaine

qu'ils supportent et selon la possibilité de stockage des informations imprécises ou sans imprécision.

3.4.1 Les attributs flous de Type 1

Ce sont des attributs avec des "données précises", classiques sur lesquels nous pouvons définir des étiquettes linguistiques. Ces d'attributs subissent la même représentation que les données précises. Cependant, ils ont des informations dans la BMCF où se sauvegardent leurs étiquettes. Ces attributs n'autorisent pas d'entreposer des valeurs floues.

3.4.2 Les attributs flous de Type 2

Ce sont des attributs qui rassemblent "des données précises aussi bien que imprécises dans un référentiel ordonné" (sous forme de distributions de possibilité). Ce sont une extension du type 1, autorisant le stockage des informations imprécises, tel que : "il a approximativement vingt ans". Avec ces attributs, nous pouvons aussi utiliser et stocker toutes les constants floues définies dans la table 2.14. Ces attributs prennent le type des données dont la représentation a été décrite dans la section 3.2.2. Ils permettent aussi la représentation des informations incomplètes dans la forme d'UNKNOWN (INCONNU), UNDEFINED (INDEFINI) et NULL.

Types de valeurs	**Attributs de la BD pour chaque type 2**				
	FT	**F1**	**F2**	**F3**	**F4**
UNKNOWN (inconnu)	**0**	NULL	NULL	NULL	NULL
UNDEFINED (indéterminé)	**1**	NULL	NULL	NULL	NULL
NULL (nul)	**2**	NULL	NULL	NULL	NULL
CRISP (exacte)	**3**	d	NULL	NULL	NULL
LABEL (étiquette)	**4**	FUZZY_ID	NULL	NULL	NULL
INTERVAL (intervalle)	**5**	n	NULL	NULL	m
APPROXIMATELY (approximative)	**6**	d	d-marge	d+marge	marge
TRAPEZE [α,β,γ,δ] (trapèze)	**7**	α	$\beta - \alpha$	$\gamma - \delta$	δ

TAB. 3.1 – Représentation interne des attributs flous de type 2

Dans le Tableau 3.1, nous découvrons le système utilisé pour représenter les attributs flous de Type 2. Ainsi, nous remarquons qu'un attribut flou de Type 2, appelé par exemple F, est composé, en réalité, par 5 attributs classiques

- **FT** : entrepose le type de la valeur que peut prendre l'attribut en question. D'après ce qui est vu, ce peut être : 0 pour UNKNOWN, 1 pour UNDEFINED,... La lettre T est ajoutée au nom de l'attribut.
- **F1, F2, F3 et F4** : Les attributs dont le nom est formé par l'addition (concaténation) des nombres 1, 2, 3 et 4 au nom de l'attribut sauvegardant la description des paramètres qui définissent les données et qui dépendent du type de la valeur (FT) à qui ils appartiennent :
  - **UNKNOWN, UNDEFINED, NULL** : Ces 3 valeurs n'ont pas besoin de paramètre, pour cela elles reçoivent tous NULL (cette valeur est comprise comme le NULL du SGBD et non pas comme le NULL de la valeur floue).
  - **CRISP** (exacte) : Une valeur de type exacte a besoin d'un seul paramètre, F1, dans lequel la valeur exacte en question sera entreposée.
  - **LABEL** (étiquette) : Également, une valeur de type étiquette nécessite seulement un paramètre pour entreposer l'identificateur associé à cette étiquette (FUZZY_ID). Cet indicateur est utile pour accéder à la BMCF et obtenir la description associée à cette étiquette.
  - **INTERVAL** (intervalle) : nécessite les deux extrémités de l'intervalle [n,m] pour être entreposé respectivement dans F1 et F4.
  - **APPROXIMATELY** (approximative) : Cette valeur nécessite seulement la valeur entreposée dans F1 qui correspond à la valeur centrale de la distribution de possibilité triangulaire, n (figure 3.3). Cependant, pour réduire les opérations (autant mathématiques comme l'accès aux données), utilisant les attributs F2, F3 et F4, nous entreposons respectivement la valeur de n-marge, n+marge et marge. La valeur de la marge est une valeur entreposée dans la BMCF pour chaque attribut flou, qui dépend de la signification de cet attribut.
  - **TRAPEZE** (trapèze) : nécessite d'entreposer les 4 valeurs qui identifient un trapèze : $[\alpha, \beta, \gamma, \delta]$. Dans F2 et F3 quelques opérations sont entreposées pour simplifier les équations utilisant ce type de données.

3.4.3 Les attributs flous de Type 3

Ce sont les attributs du "domaine discret non ordonné avec ressemblance". Ces attributs prennent des valeurs linguistiques simples (SIMPLE) ou des distributions

de possibilité (DISTR. POS.) sur les domaines linguistiques dont la représentation a été décrite dans la section 3.2.2, comme par exemple, la valeur {0.8/Tunis, 0.2/Sfax} qui exprime qu'une ville est plus possible être proche de Tunis que Sfax. Ils acceptent aussi les données UNKNOWN, UNDEFINED et NULL.
La table 3.2 montre le système utilisé pour représenter ces attributs. Comme nous voyons, un attribut flou type 3, appelé par exemple F, est composé, en fait, par un nombre variable d'attributs classiques :

- **FT** : Le type de valeur qui correspond à la donnée qui va être sauvegardée. Il peut être : UNKNOWN (0), UNDEFINED (1), NULL (2), SIMPLE (3) et DISTRIBUTION de POSSIBILITE (4).
- **Liste de n paires**, avec $n \geq 1$, de type (valeur de possibilité, étiquette), (FP1,F1), ... ,(FPn,Fn) : dans ces attributs, se sauvegardent les données de la distribution de possibilité. Dans une valeur de type SIMPLE seulement le premier couple est utilisé et la valeur de possibilité sera 1 (pour être normalisé).

Types de valeurs	Attributs de la BD pour chaque type 3					
	FT	**FP1**	**F1**		**FPn**[1]	**Fn**
UNKNOWN (inconnu)	**0**	NULL	NULL	...	NULL	NULL
UNDEFINED(indéterminé)	**1**	NULL	NULL	...	NULL	NULL
NULL (nul)	**2**	NULL	NULL	...	NULL	NULL
SIMPLE	**3**	p	d	...	NULL	NULL
DISTR. POS.	**4**	p1	d1	...	*p*n	*d*n

TAB. 3.2 – Représentation interne des attributs flous type 3

Dans une valeur du type DISTR. POS, nous pouvons entreposer jusqu'à n paires, dans chacune d'elles la valeur de possibilité sera dans l'intervalle [0,1]. Nous pouvons utiliser moins de n paires en initialisant le reste des champ à NULL. Toutes ces caractéristiques ainsi que les objets définis sur ce type d'attribut sont entreposés dans la BMCF.

3.5 Base de Méta connaissances Floues (BMCF)

Comme nous l'avons vu dans la section précédente, certains types d'information qui existent sur les attributs déjà décrits doivent être entreposé dans un chemin accessible par le système. La BMCF, en anglais Fuzzy Meta Knowledge Base, sera responsable d'organiser cette information en fonction de la nature imprécise de ces

attributs. Dans FIRST, la BMCF floue est considérée comme une extension du catalogue du système, elle organise l'information au moyen de l'usage des tables ou des relations. Les éléments du traitement imprécis entreposés dans la BMCF sont les suivants :

- Les attributs de la BD qui subissent un traitement imprécis.
- La classe d'information imprécise retenue : les type des attributs flous (FTYPE1, FTYPE2 ou FTYPE3) et la longueur maximale des distributions de possibilité pour les attributs de type 3.
- Les objets flous définis dans l'environnement de la BD, comme par exemple les quantificateurs flous.
- La définition des objets flous pour chaque attribut :
  - Les étiquettes linguistiques (pour les attributs flous de type 1, 2 et 3).
  - La marge des valeurs approximatives et la distance minimale pour considérer deux valeurs comme très séparées (pour les attributs flous de type 1 et 2).
  - Les relations de ressemblance (pour les attributs flous 3).
- La description de ces objets.

3.5.1 Relations dans la BMCF

La figure 3.6 montre les relations (les tables) dans la BMCF, ses attributs, ses clefs primaires (soulignées) et ses clefs étrangères (avec les flèches). OBJ# est utilisé comme l'identificateur de la relation, et COL# comme la colonne ou identificateur de l'attribut (comme Oracle). Les relations de la BMCF sont :

- **FUZZY_COL_LIST** : Elle contient une description des attributs de la BD susceptibles du traitement flou. Elle décrit les attributs flous identifiés par (OBJ#, COL#). Le type d'attribut F_TYPE est entre 1 et 3, LEN est la valeur maximale de distribution de possibilité des attributs de type 3, c-à-d., le nombre maximal de couple (valeur de possibilité, étiquette) admettant une distribution de possibilité dans cet attribut.
- **FUZZY_OBJECT_LIST** : Elle contient une liste d'objets flous qui sont définis dans les colonnes de la BD. Ces objets flous peuvent être des étiquettes linguistiques, des qualificateurs flous et des quantificateurs flous.
- **FUZZY_LABEL_DEF** : Elle définit les étiquettes linguistiques qui utilisent des fonctions trapézoïdales.
- **FUZZY_APPROX_MUCH** : Elle sauvegarde les valeurs "marge" et "much" pour les types 1 et 2.

- **FUZZY_NEARNESS_DEF** : Elle sauvegarde les mesures de voisinage (ressemblance) entre les différentes valeurs des attributs type 3.
- **FUZZY_COMPATIBLE_COL** : Elle sauvegarde les attributs flous compatibles entre eux, ç-à-d., ceux qui utilisent les mêmes étiquettes linguistiques. De cette manière, il n'est pas nécessaire de définir les étiquettes et les degrés de ressemblance pour chacune d'elles.
- **FUZZY_QUALIFIERS_DEF** : Elle définit les qualificateurs flous.

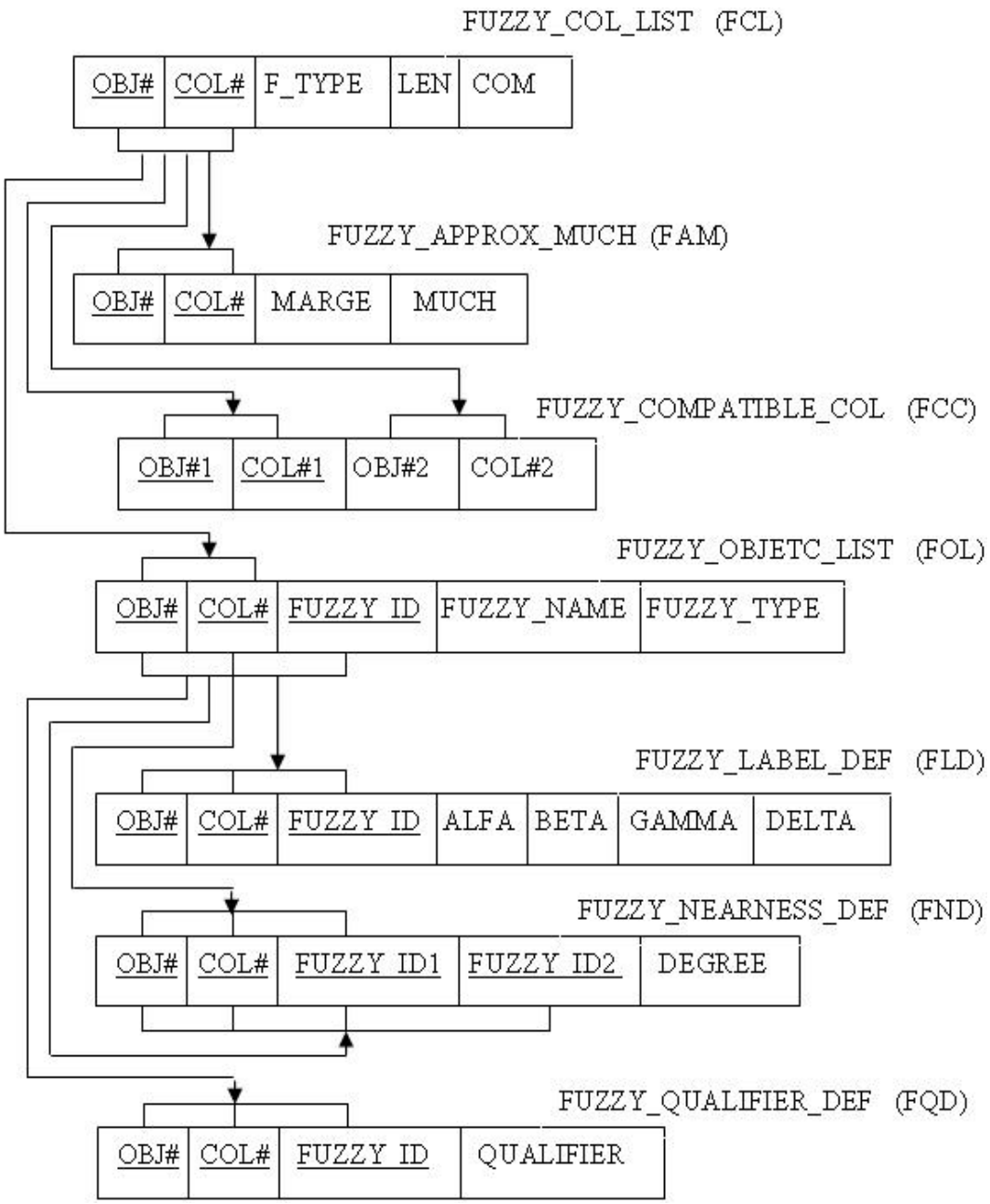

FIG. 3.6 – Schéma de la BMCF

3.5.2 Vues de La BMCF

Sur les tables de FIRST quelques vues ont été créées. Elles sont utile pour consulter les différents aspects d'une BD floue d'une façon claire et simple. Ces vues sont les suivantes :

- **LABELS_FOR_OBJCOL** : Elle sert à obtenir ou à consulter facilement les étiquettes définies sur les attributs flous type 1 ou 2 et les paramètres du trapèze possibiliste associés.
- **LABELS_OBJCOL_T3** : Elle sert à obtenir ou à consulter facilement les étiquettes définies sur les attributs flous Type 3, aussi bien que le degré de ressemblance parmi eux.
- **ALL_COMPATIBLES_T3** : Elle sert à obtenir et à consulter facilement les attributs flous Type 3 compatibles entre eux. De cette forme, nous pouvons savoir lesquels des attributs flous Type 3 qui n'ont pas d'étiquettes définies sur eux et lesquels qui ont. Aussi, elle nous montre aussi la longueur maximale autorisée pour les distributions de possibilité des attributs flous Type 3.

3.6 Le serveur FSQL (FSQL Server)

Il a été programmé totalement en PL/SQL et il inclut trois types de fonctions [GAL 98a] :

- **Fonction de la translation (FSQL2SQL)** : Elle réalise l'analyse syntaxique, sémantique et lexicale de la requête FSQL. Si des erreurs sont détectées, elle produira une table avec toutes ces erreurs. S'il n'y a pas d'erreurs, la requête FSQL est traduite en une requête SQL standard. La requête SQL résultante inclut une référence aux fonctions suivantes.
- **Fonctions de représentation** : Ces fonctions sont utilisées pour mettre l'attribut flou dans une façon compréhensible pour l'utilisateur évitant ainsi le format interne (compliqué).
- **Fonctions de comparaison floues** : Elles sont utilisées pour comparer les valeurs floues et pour calculer les degrés de la compatibilité (fonction CDEG).

La fonction de translation permet de remplacer les attributs flous du SELECT par appels aux fonctions de représentation, les conditions floues par appels aux fonctions de comparaison floues et la fonction CDEG par les appels aux fonctions de comparaison floues et d'autres fonctions s'il existe quelques opérateurs logiques.

3.6.1 Implémentation du serveur FSQL

L'objectif du serveur FSQL est de réussir à traduire une requête FSQL à une requête SQL, au moyen d'appels à ses fonctions. Pour ceci, plusieurs paquets (packages) PL/SQL ont été créés [GAL 99a] afin que chaque partie du serveur soit dans un paquet différent. Principalement, le serveur FSQL se compose d'un point de vue conceptuel des modules suivants :

- **Analyseur lexical** : Il est chargé de vérifier si la requête FSQL est correcte de point de vue lexicale, en produisant une liste avec les jetons (mots) de la phrase.
- **Analyseur syntaxique** : Il vérifie si la phrase est correcte syntaxiquement. Il utilise une grammaire qui produit des phrases admettant les extensions de FSQL.
- **Analyseur sémantique et convertisseur** : Il vérifie si la phrase est correcte de point de vue sémantique et, en même temps, il produit la phrase SQL équivalente.
- **Fonctions de représentation et de comparaison floue** : Ces fonctions sont utilisées pour représenter les attributs flous afin qu'ils soient compréhensibles par l'utilisateur. De plus, elles permettent d'éffectuer l'opération de traduction.

3.6.2 Fonctionnement du serveur FSQL

Le processus d'appel au serveur FSQL est schématisé dans la figure 3.7. En général, pour une requête FSQL, les étapes suivantes sont exécutées :

1. Le client FSQL envoie une requête FSQL au serveur FSQL.
2. LE serveur FSQL analyse la requête, et si elle est correcte, il génère une requête SQL. Dans cette phase, le serveur FSQL utilise les informations contenues dans la FMB.
3. Une fois la requête a été générée en SQL, le programme client la lira.
4. Le programme Client peut envoyer la requête SQL à toute BD cohérente avec la FMB. Dans l'exécution de cette requête, les fonctions du Serveur FSQL seront utilisées (fonctions de représentation et fonctions de comparaison floues).
5. Finalement le client recevra les données résultantes et il les affichera.

Les étapes 3 et 4 doivent être éliminées pour augmenter l'efficacité du système mais, selon cette façon de présentation nous remarquons une indépendance entre la phase de traduction (étapes 1, 2 et 3) et la phase de consultation (étapes 4 et 5).

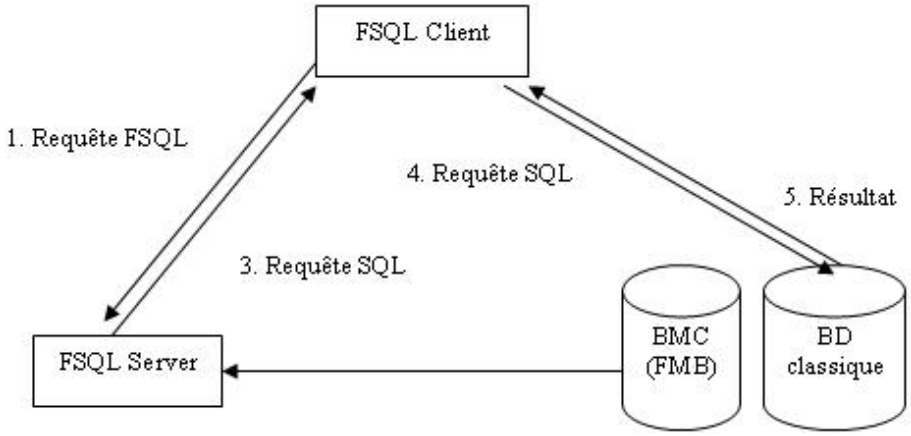

FIG. 3.7 – Schéma de la BMCF

3.7 Le Client FSQL (FSQL CLIENT : FQ)

Le Client FSQL (FQ) est un programme indépendant qui sert comme une interface entre l'utilisateur et le Serveur FSQL [GAL 99b]. L'utilisateur introduit une requête FSQL et le programme client communique avec le serveur et avec la BD dans le but d'obtenir le résultat final. La fonction de translation du Serveur FSQL est la seule fonction qui sera exécutée directement par le client.
FQ version 1.3, est simple, il permet d'éffectuer des consultations (interrogations) floues avec le langage FSQL, bien qu'il puisse aussi être utilisé pour faire des consultations classiques avec SQL. Le programme FQ a été présenté internationalement pour la première fois dans [GAL 98c]. L'installation du serveur FSQL, de FQ et les autres éléments qui assurent son fonctionnement sont présentés dans[GAL 99a].

3.7.1 Objectifs et Fonctionnement

Le programme FSQL Client est responsable d'envoyer une requête écrite en FSQL au Serveur FSQL et obtenir de ce dernier les erreurs contenues dans cette requête. S'il n'y a pas d'erreur, le Client doit recevoir sa requête FSQL traduite en SQL. La relation entre le Client FSQL et le Serveur FSQL est faite d'une manière transparente. L'utilisateur ne voit rien, il n'a donc pas besoin d'avoir une grande connaissance sur le fonctionnement du Serveur FSQL.

3.7.2 Modélisation d'une requête dans FQ

Le logiciel FQ offre une interface interactive simple, qui permet l'édition de requêtes SQL et FSQL. En effet, il étend les concepts de bases du SQL classique en ajoutant les comparateurs et les constantes floues définies dans le langage FSQL.

FIG. 3.8 – Fenêtre principale de FQ

Edition d'une requête SQL simple

Les concepts de base du SQL restent encore valides dans le FQ à savoir les opérateurs logiques, ensemblistes, les comparateurs classiques (exactes). Donc nous pouvons établir une requête classique, comme étant dans l'éditeur SQL Plus.

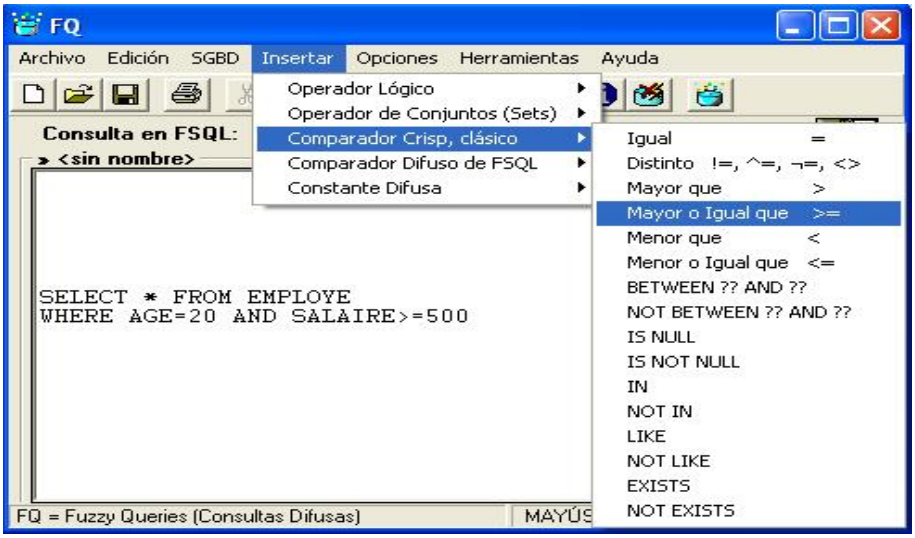

FIG. 3.9 – Les comparateurs classiques

Edition d'une requête SQL Floue

En plus des opérations classiques du langage SQL, le FSQL Client offre les différents comparateurs et constantes floues présentés précédemment (section 2.5.2).

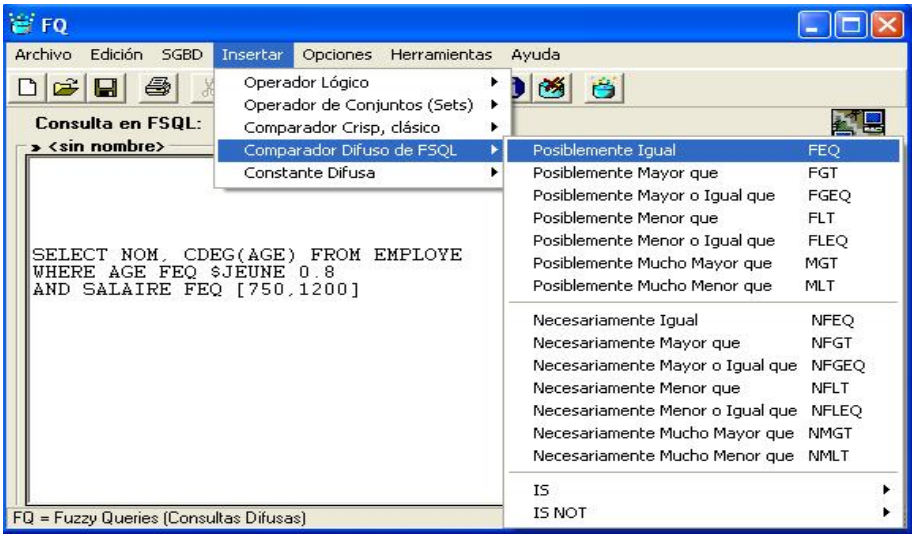

FIG. 3.10 – Les comparateurs flous dans FQ

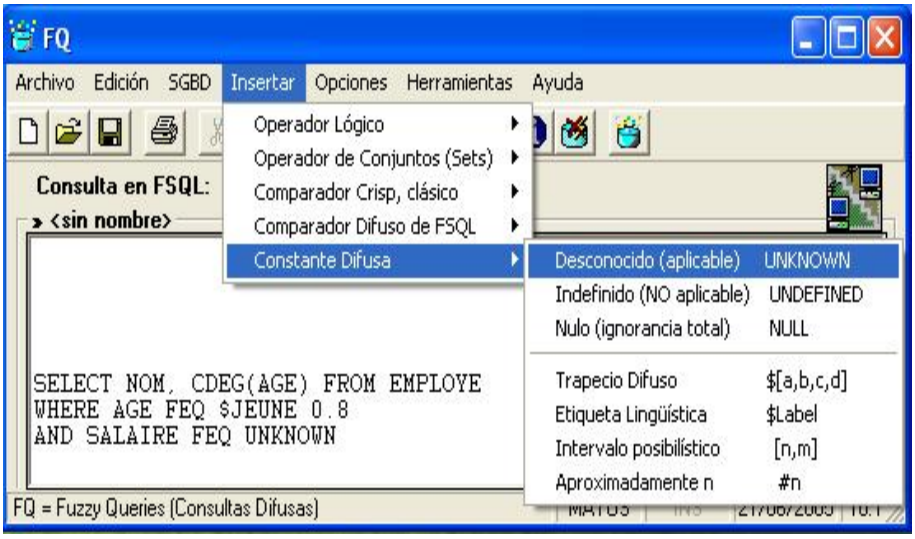

FIG. 3.11 – Les constantes floues dans FQ

Cette section peut varier largement d'un programme client à un autre, mais l'utilisateur doit écrire sa requête en mode textuel, donc il doit avoir une connaissance sur le langage SQL et de son extension FSQL.

3.7.3 Limites de FQ

Comme tout programme, FQ présente des points forts et des points faibles. Parmi ses inconvénients, son menu et ses différents boutons qui sont écrits en espagnol, une langue non standard. Nous verrons aussi, dans le chapitre suivant, que les seules requêtes traitées par FQ sont de type SELECT.
Or avant la consultation de toute BDF, il faut qu'elle soit créée. La création d'une BDF avec ses objets se fait par le langage de définition de données inclut dans FSQL. Néanmoins, FQ ne permet pas de modéliser les commandes (CREATE TABLE, CREATE LABEL,...).

3.8 Conclusion

Nous avons présenté dans ce chapitre l'architecture d'une BDRF basée sur le modèle GEFRED avec tous ses composants et le mécanisme de son fonctionnement. Cette architecture présente un logiciel d'interrogation de la BDF puissant dans les requêtes de type SELECT mais ne modélise pas les commandes LDD de FSQL. Nous détaillerons cette limite dans le chapitre suivant.

Chapitre 4

Implémentation d'une BDF à l'aide du modèle GEFRED

4.1 Introduction

La mise en place de tout système nécessite une étude bien détaillée. Cette étude doit respecter les besoins des utilisateurs. Une question se pose dans le cadre des BDF est l'implémentation d'une BDF décrite avec le langage FSQL sous un SGBDR. L'architecture FIRST présentée dans le chapitre précédent a proposé une solution à ce problème, en transformant le script FSQL en un script SQL équivalent, et en modifiant la BMCF correspondante. Cette transformation est faite manuellement par le concepteur de la BDF qui est obligé, dans ce cas, de connaitre la structure de données interne de la BMCF, et les transformations que doit subir chaque attribut flou.
Le but de ce chapitre est d'expliquer cette transformation avec un exemple simple de BDF et montrer les difficultés et les problèmes rencontrés par un utilisateur simple de la DBF.

4.2 Implémentation d'une BDRF sous ORACLE 8i

Nous proposons dans ce qui suit la mise en oeuvre de notre BD ENTREPRISE présentée dans le chapitre 1 en l'enrichissant avec les concepts flous pour qu'elle puisse supporter les requêtes flexibles.

4.2.1 Description de la BDF

Nous présentons dans cette partie la description de la BD ENTREPRISE pour qu'elle puisse supporter le concept de requêtes flexibles. Nous avons ajouté des nouveaux objets flous en particulier les étiquettes linguistiques, les relations de similitudes,... Cette description est comme suit :
Un département possède un nom, un numéro unique et un directeur dont la date d'entrée est mémorisée. Un département possède un **local**. Il peut contrôler plusieurs projets mais un projet est contrôlé par un seul département. Chaque projet est caractérisé par un nom, un numéro et un **budget**. Un employé est décrit par un nom, une matricule unique, une adresse, un **salaire**, un sexe et une date de naissance. Un employé possède un seul supérieur hiérarchique immédiat. Il possède aussi un **niveau d'études** et un **rendement**. Un employé est affecté à un seul département, mais il peut travailler sur plusieurs projets non nécessairement rattachés à son département. Le **nombre d'heures** par semaine passé par un employé sur chaque projet est mémorisé. Un poste est décrit par un code unique, un nom (**profession**) et exige un niveau d'études et une **expérience**. Chaque employé possède des personnes qui sont à sa charge (conjoint, enfants,etc.) ou dépendants. Chaque dépendant est décrit par : un prénom, un sexe, une date de naissance et un lien de parenté avec l'employé. Certains des attributs décrits précédemment peuvent être interrogés par des requêtes flexibles (floues). Ils possèdent les caractéristiques suivantes :

SALAIRE : possède les étiquettes linguistiques définies sur les distributions de possibilité trapézoïdales suivantes : `BAS`(50,80,120,180), `MOYEN`(150,300,400,550), `ÉLEVÉ`(400,600,800,1000). Une valeur approximative sur les valeurs de cet attribut est de ± 10 et la valeur minimale pour considérer deux valeurs de cet attribut comme totalement différentes est de 50. La figure 4.1 montre une représentation de ces étiquettes.

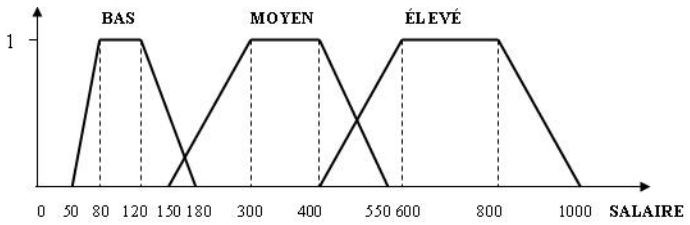

FIG. 4.1 – Etiquettes linguistiques de l'attribut SALAIRE

NOMBRE_HEURE : possède les étiquettes linguistiques : `PETIT`(0,2,7,8), `MOYEN`(5,10,15,20), `ÉLEVÉ` (15,20,25,30). Une valeur approximative sur les valeurs de cet attribut est de ±2 et la valeur minimale pour considérer deux de ces valeurs comme totalement différentes est de 4. Ces attributs ne peuvent entreposer que des valeurs exactes. L'interrogation flexible effectuée sur eux comporte l'utilisation des valeurs approximatives, des distributions de possibilité trapézoïdales, des étiquettes linguistiques, des intervalles, notamment les valeurs `UNKNOWN`,`UNDEFINED` et `NULL`.
BUDGET : possède les étiquettes linguistiques : `PETIT`(100,500,1000,1800), `MOYEN`(1500,5000,10000,15000), `GROS`(12000,20000,50000,10000). Ces valeurs sont en dinars Tunisien. Une valeur approximative est de ś±200 et la valeur minimale pour considérer deux valeurs de budget comme totalement différentes est de 1000.
Supposons que les attributs ÂGE et EXPÉRIENCE obéissent aux mêmes règles concernant l'interrogation flexible avec la possibilité d'entreposer en plus des valeurs exactes, des valeurs imprécises (floues) dans un référentiel ordonné tel que UNKNOWN, UNDEFINED, NULL, des étiquettes linguistiques, des intervalles, des valeurs approximatives et des distributions de possibilités. Nous citons les suivants :
ÂGE : nous avons ajouté cet attribut à la place de date_naissance pour faciliter la représentation et pour ne pas répéter le calcul DATE_SYSTEME - DATE_NAISSANCE chaque fois que nous avons besoin de l'âge d'une personne[1]. Il possède les étiquettes linguistiques définies sur les distributions de possibilité trapézoïdales suivantes : `JEUNE`(18,22,30,35), `ADULTE`(25,32,45,50), `VIEUX`(50,55,62,70). Une valeur approximative a une marge de 5 et la valeur minimale pour considérer deux valeurs de cet attribut comme totalement différentes est de 10. La figure 4.2 montre une représentation des étiquettes linguistiques définis sur cet attribut.

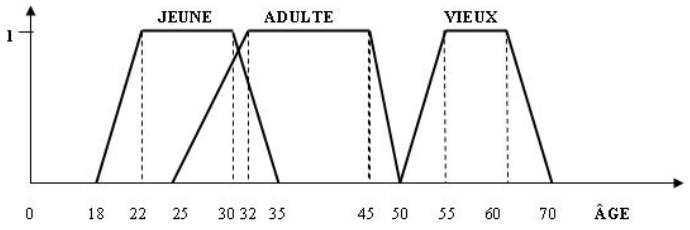

FIG. 4.2 – Étiquettes linguistiques de l'attribut ÂGE

EXPÉRIENCE : possède les étiquettes linguistiques : `PETITE`(2,3,5,6), `BONNE`(5,7,10,12),

[1]L'attribut ÂGE se calcule comme suit :
SELECT (TO_CHAR(SYSDATE,'yyyy') - TO_CHAR(DATE_NAISSANCE,'yyyy'))into ÂGE FROM EMPLOYÉ;

`SUFFISANTE`(7,8,15,20), `GRANDE`(12,15,50,50).
Ces valeurs dépendent des nombres d'années travaillés par un employé. Une valeur approximative est de ±2 et la valeur minimale pour considérer deux expériences comme totalement différentes est de 5.
Supposons aussi que les attributs LOCAL, PROFESSION, ÉTUDES et RENDEMENT peuvent entreposer des valeurs imprécises ayant des ressemblances entre elle.
LOCAL : entrepose les villes avec les degrés de similarités entre elles (tableau 4.1) :

Degré de similarité	**Centre ville**	**Bardo**	**Ariana**	**Ben Arous**	**Nabeul**	**Sousse**
Centre ville	1	0.95	0.7	0.6	0.4	0.2
Bardo	0.95	1	0.8	0.5	0.3	0.2
Ariana	0.7	0.8	1	0.5	0.3	0.1
Ben Arous	0.6	0.5	0.5	1	0.7	0.4
Nabeul	0.4	0.3	0.3	0.7	1	0.3
Sousse	0.2	0.2	0.1	0.4	0.3	1

TAB. 4.1 – Relations de similitude pour les valeurs de l'attribut Local

PROFESSION : entrepose les professions disponibles dans cette entreprise avec les degrés de similarités entre elles(tableau 4.2).

Degré de similarité	**Directeur**	**Ingénieur**	**Technicien**	**Administratif**	**Secrétaire**
Directeur	1	0.8	0.5	0.6	0.4
Ingénieur	0.8	1	0.8	0.2	0.2
Technicien	0.5	0.8	1	0.8	0.6
Administratif	0.6	0.2	0.8	1	0.7
Secrétaire	0.4	0.2	0.6	0.7	1

TAB. 4.2 – Relations de similitude pour les valeurs de l'attribut Profession

ÉTUDES : définit le niveau d'études d'un employé et d'un poste. Les valeurs que peut avoir cet attribut possèdent des ressemblances entre elles. Les degrés de ces ressemblances sont décrits dans la table 4.3.
RENDEMENT entrepose la valeur que peut avoir un rendement d'employé et le

Degré de similarité	Secondaire	Diplôme	Licence	Doctorat
Secondaire	1	0.5	0.1	0.05
Diplôme	0.5	1	0.6	0.2
Licence	0.1	0.6	1	0.7
Doctorat	0.05	0.2	0.7	1

TAB. 4.3 – Relations de similitude pour les valeurs de l'attribut Etude

Degré de similarité	MAUVAIS	RÉGULIER	BON
MAUVAIS	1	0.3	0.2
RÉGULIER	0.3	1	0.7
BON	0.2	0.7	1

TAB. 4.4 – Relations de similitude pour les valeurs de l'attribut rendement

degré de similarité avec les autres valeurs (table 4.4).
Sur ces attributs, nous pouvons entreposer des valeurs tels que `UNKNOWN`, `UNDEFINED`, `NULL`, des valeurs simples et des distributions de possibilités (en fonction de nombre de valeurs qu'entrepose un attribut).
Nous supposons que cette BD peut répondre aux requêtes suivantes :
`Requête 1` : "Trouver les employés `jeunes` qui travaillent sur un projet à `gros` budget"
`Requête 2` :"Trouver les employés et leurs expériences qui sont aptes à travailler dans des postes qui exigent une expérience `suffisante`".
`Requête 3` :"Trouver les employés qui peuvent être acceptés pour un poste d'`ingénieur` ayant une `Bonne` expérience.

4.2.2 Détermination des attributs flous

En appliquant les règles de Medina et Galindo [GAL 99a], nous pouvons déduire les attributs flous suivants :
- Attributs flous type 1 : `SALAIRE FTYPE1(10,50)`, `BUDGET FTYPE1(200,1000)`, `NBRE_HEURE FTYPE1(2,4)`.
- Attributs flous type 2 : `ÂGE FTYPE2(5,10)`, `EXPÉRIENCE FTYPE2(2,5)`.
- Attributs flous type 3 : `RENDEMENT FTYPE3(1)`, `ÉTUDES FTYPE3(1)`, `PROFESSION`

`FTYPE3(1), LOCAL FTYPE3(1).`

Notons que cette déduction n'est pas évidente. Il faut bien maîtriser les concepts d'attributs flous FTYPE1, FTYPE2 et FTYPE3 pour pouvoir affirmer que tel attribut appartient à telle classe de type. En fait, la détermination de type des attributs est une étape très importante dans le processus de création d'une BDF. Plusieurs critères interviennent dans cette tâche. Nous présentons quelques exemples illustrant notre fondement dans le choix des types des attributs de notre BD ENTREPRISE :

- **SALAIRE** : cet attribut entrepose le salaire de chaque employé. En effet, en se basant sur le fait qu'il ne peut entreposer que des valeurs exactes et qu'il peut être interrogé par des valeurs floues, cet attribut doit être de **type 1**.
  La question qui se pose est : pourquoi ne peut-il pas être de type 2 ?
  La réponse n'est pas assez simple. Dans notre BD, le salaire est très important (une entreprise a un budget bien déterminé, des impôts, des revenues, ...), donc ce n'est pas logique d'entreposer des valeurs floues. Par contre, une interrogation de type "donner les employés ayant un salaire BAS" sera possible. La figure 4.1 montre la définition des étiquettes linguistiques définies sur cet attribut. La valeur de la marge pour les valeurs approximatives est de 10 et la distance minimale pour considérer deux valeurs de cet attribut comme très séparés est de 50.
- **ÂGE** : entrepose l'âge de chaque employé. C'est un attribut flou de **type 2**, nous pouvons entreposer tous les types de données qui ont été montrées dans le tableau 3.1. En fait le choix du type 2 est dû à la difficulté et en général à la non nécessité de savoir l'âge exact d'une personne, bien que ce sera énormément utile de savoir son âge approximatif. De cette forme nous pouvons dire qu'une personne est JEUNE, ADULTE, VIELLE ou qu'il a approximativement 8 ans.
- **ÉTUDES** : entrepose le niveau d'études d'un employé. Les valeurs que peut avoir cet attribut sont SECONDAIRE, DIPLÔME, LICENCE, DOCTORAT qui ont des relations de ressemblance. Ces valeurs présentent des ressemblances entre elles. Cet attribut sera donc de **type 3**. Nous pouvons dire qu'un diplômé ressemble à un licencié avec un degré 0.8 et qu'un secondaire ressemble à un docteur avec un degré 0.1 (presque différents).
  Nous remarquons aussi que dans cette BDF, cet attribut est utilisé deux fois : la première dans la table EMPLOYÉ suivi par la valeur 1 entre parenthèse et la deuxième dans la table POSTE suivi par la valeur 4. Ce nombre qui est indiqué entre parenthèse indique le nombre maximal de donnée, c'est-à-dire le nombre maximale de distribution de possibilité représentés sur cet attribut. Par exemple,

pour un nombre égal à 4 dans la table POSTE, nous pouvons chercher un employé avec un niveau d'études égal à 0.2/SECONDAIRE, 0.6/DIPLOME,1/LICENCE et 0.8/DOCTORAT. Cette interrogation s'interprète par le fait que nous cherchons un employé avec les préférences d'études licencié, docteur, diplômé, secondaire ordonnées selon leurs degrés resp. 1, 0.8, 0.6, 0.2. Ce qui n'est pas le cas pour un nombre égal à 1, car dans la table EMPLOYÉ, un employé possède un niveau d'études qui est connu c'est-à-dire diplômé avec une distribution de possibilité égale à 1 (1/DIPLOME, 1/LICENCE,...).

- **RENDEMENT** : entrepose une qualification du rendement de chaque employé. Cet attribut ne peut pas entreposer des valeurs exactes donc il ne peut pas être de type 1, comme il présente des ressemblances dans ces valeurs. Par conséquent, nous lui attribuons le **type 3**.
- **LOCAL** : Cet attribut a été implémenté avec la longueur 3 (valeur LEN de la table FCL (section 3.5.1)), afin qu'il permette d'indiquer qu'un département est situé entre 3 régions, avec un degré différent. Par exemple, la valeur 0.5/Centre ville, 1/Bardo, 0.7/Ariana qui s'interprète par le fait que le département en question est localisé dans le quartier de Bardo, plus près du quartier de Ariana que de Centre-ville. Le rapport de ressemblance (similitude) entre les différentes régions ou quartiers dépendra de la distance existante parmi eux. Par conséquent, nous lui attribuons le **type 3**.

4.2.3 Modélisation de la BDF

La BDF modélisant la description décrite ci-dessus sera décrite avec les mêmes tables que nous avons définis dans le cas de la BDR (voir chapitre1), mais où les attributs sont déclarés de types floues : FTYPE1, FTYPE2 et FTYPE3.
La BDF sera décrite en langage FSQL comme suit :

```
CREATE TABLE DÉPARTEMENT(
NUM_D VARCHAR2(4) NOT NULL,
NOM VARCHAR2(20) NOT NULL,
MATRICULE_DIRECTEUR VARCHAR2(4) NOT NULL,
DATE_DÉBUT DATE NOT NULL,
LOCAL FTYPE3(3) DEFAULT UNKNOWN NOT NULL,
PRIMARY KEY (NUM_D));
```

CREATE TABLE **POSTE**(
CODE_POSTE VARCHAR2(4) NOT NULL,
PROFESSION FTYPE3(1) NOT NULL,
ÉTUDES FTYPE3(4) NOT NULL,
EXPÉRIENCE FTYPE2(2,5) NUMBER(2) NOT NULL,
PRIMARY KEY (CODE_POSTE));

CREATE TABLE **EMPLOYÉ**(
MATRICULE VARCHAR2(4) NOT NULL,
NUM_D VARCHAR2(4) NOT NULL,
CODE_POSTE VARCHAR2(4) NOT NULL,
NOM VARCHAR2(20) NOT NULL,
PRENOM VARCHAR2(20) NOT NULL,
SEXE VARCHAR2(1) NOT NULL,
ADRESSE VARCHAR2(40) NOT NULL,
ÂGE FTYPE2(5,10) NUMBER(3) DEFAULT UNKNOWN NOT NULL,
SALAIRE FTYPE1(10,50) NUMBER(7) NOT NULL,
ÉTUDES FTYPE3(1) DOMAIN POSTE.ÉTUDES NOT NULL,
RENDEMENT FTYPE3(1) NOT NULL,
EXPÉRIENCE FTYPE2(2,5) DEFAULT UNKNOWN NUMBER(2) NOT NULL,
MATRICULE_SUPERVISEUR VARCHAR2 (4) NOT NULL,
PRIMARY KEY (MATRICULE, NUM_D, CODE_POSTE),
FOREIGN KEY (NUM_D) REFERENCES DÉPARTEMENT ON DELETE CASCADE,
FOREIGN KEY (CODE_POSTE) REFERENCES POSTE ON DELETE CASCADE);

CREATE TABLE **PROJET**(
NUM_P VARCHAR2(4)NOT NULL,
NUM_D VARCHAR2(4) NOT NULL,
NOM VARCHAR2(15) NOT NULL,
BUDGET FTYPE1(200,1000) NUMBER(7) NOT NULL,
PRIMARY KEY (NUM_P,NUM_D),
FOREIGN KEY (NUM_D) REFERENCES DEPERTEMENT ON DELETE CASCADE);

CREATE TABLE **DÉPENDENT**(
MATRICULE VARCHAR2(4) NOT NULL,

NOM VARCHAR2(15) NOT NULL,
DATE_NAISSANCE DATE NOT NULL,
LIEN_PARENTE VARCHAR2(20) NOT NULL,
PRIMARY KEY (MATRICULE,NOM),
FOREIGN KEY (MATRICULE) REFERENCES EMPLOYÉ ON DELETE CASCADE) ;

CREATE TABLE **TRAVAILLE_DANS**(
MATRICULE VARCHAR2(4) NOT NULL,
NUM_P VARCHAR2(4) NOT NULL,
NBRE_HEURE FTYPE1(2,4) NUMBER(4) NOT NULL,
PRIMARY KEY (MATRICULE, NUM_P),
FOREIGN KEY (MATRICULE) REFERENCES EMPLOYÉ ON DELETE CASCADE, FOREIGN KEY (NUM_P) REFERENCES PROJET ON DELETE CASCADE) ;

4.2.4 Implémentation de la BDF

L'implémentation de cette BD est faite sous Oracle 8i. Vu que Oracle 8i ne supporte pas le langage FSQL, la BD doit être décrite en langage SQL. Pour cela, comme nous l'avons décrit précédemment, Medina transforme le script FSQL, *manuellement* en un script SQL présenté au SGBD Oracle pour créer la BDF et pour remplir la BMCF correspondante. Cette opération est une opération délicate qui nécessite la connaissance de la BMCF.

a) Script de la BDF en langage SQL

Dans cette partie, nous exposons les scripts nécessaires pour l'implémentation de notre BDF. Nous commençons par la création des tables de cette BD.

CREATE TABLE **DÉPARTEMENT**(
NUM_D CHAR(4) NOT NULL,
NOM CHAR(20) NOT NULL,
MATRICULE_DIRECTEUR CHAR(4) NOT NULL,
DATE_DÉBUT DATE NOT NULL,
LOCAL**T** NUMBER(1) DEFAULT **0** NOT NULL
check (LOCALT BETWEEN 0 AND 4),
LOCALP1 NUMBER(3,2),
LOCAL1 NUMBER(3),

LOCALP2 NUMBER(3,2),
LOCAL2 NUMBER(3),
LOCALP3 NUMBER(3,2),
LOCAL3 NUMBER(3),
PRIMARY KEY (NUM_D));

CREATE TABLE **POSTE**(
CODE_POSTE CHAR(4) NOT NULL,
PROFESSION**T** NUMBER(1) NOT NULL,
CHECK (PROFT BETWEEN 0 AND 4),
PROFESSIONP1 NUMBER(3,2),
PROFESSION1 NUMBER(3) NOT NULL,
ÉTUDES**T** NUMBER(1) NOT NULL
CHECK (ETUDEST BETWEEN 0 AND 4),
ÉTUDESP1 NUMBER(3,2),
ÉTUDES1 NUMBER(3),
ÉTUDESP2 NUMBER(3,2),
ÉTUDES2 NUMBER(3),
ÉTUDESP3 NUMBER(3,2),
ÉTUDES3 NUMBER(3),
ÉTUDESP4 NUMBER(3,2),
ÉTUDES4 NUMBER(3),
EXPERIENCE**T** NUMBER(1) NOT NULL
check (EXPERIENCET BETWEEN 0 AND 7),
EXPÉRIENCE1 NUMBER(2),
EXPÉRIENCE2 NUMBER(2),
EXPÉRIENCE3 NUMBER(2),
EXPÉRIENCE4 NUMBER(2),
PRIMARY KEY (CODE_POSTE));

CREATE TABLE **EMPLOYÉ**(
MATRICULE CHAR(4) NOT NULL,
NUM_D CHAR(4) NOT NULL,
CODE_POSTE CHAR(4) NOT NULL,
NOM CHAR (20) NOT NULL,

PRÉNOM CHAR (20) NOT NULL,
SEXE CHAR(1) NOT NULL,
ADRESSE CHAR(40) NOT NULL,
ÂGE**T** NUMBER(1) DEFAULT **0** NOT NULL
CHECK (AGET BETWEEN 0 AND 7)
CONSTRAINT NULL_INVALIDO_ÂGE CHECK (AGET<>2)
CONSTRAINT UNDEFINED_INVALIDO_ÂGE CHECK (AGET<>1),
ÂGE1 NUMBER(3),
ÂGE2 NUMBER(3),
ÂGE3 NUMBER(3),
ÂGE4 NUMBER(3),
SALAIRE NUMBER(7) NOT NULL,
ÉTUDES**T** NUMBER(1) NOT NULL
CHECK (ETUDEST BETWEEN 0 AND 4),
ÉTUDESP1 NUMBER(3,2),
ÉTUDES1 NUMBER(3) NOT NULL,
RENDEMENT**T** NUMBER(1) NOT NULL
CHECK (RENDEMENTT BETWEEN 0 AND 4),
RENDEMENTP1 NUMBER(3,2),
RENDEMENT1 NUMBER(3) NOT NULL,
EXPÉRIENCE**T** NUMBER(1) NOT NULL
check (EXPERIENCET BETWEEN 0 AND 7),
EXPÉRIENCE1 NUMBER(2),
EXPÉRIENCE2 NUMBER(2),
EXPÉRIENCE3 NUMBER(2),
EXPÉRIENCE4 NUMBER(2),
MATRICULE_SUPERVISEUR CHAR (4) NOT NULL,
PRIMARY KEY (MATRICULE,NUM_D,CODE_POSTE),
FOREIGN KEY (NUM_D) REFERENCES DÉPARTEMENT ON DELETE CASCADE,
FOREIGN KEY (CODE_POSTE) REFERENCES POSTE ON DELETE CASCADE);

CREATE TABLE **PROJET**(
NUM_P CHAR(4)NOT NULL,
NUM_D CHAR(4) NOT NULL,
NOM CHAR(50) NOT NULL,

```
BUDGET NUMBER(7) NOT NULL
PRIMARY KEY (NUM_P,NUM_D),
FOREIGN KEY (NUM_D) REFERENCES DÉPARTEMENT ON DELETE CASCADE);

CREATE TABLE DÉPENDENT(
MATRICULE CHAR(4) NOT NULL,
NOM CHAR (20) DEFAULT NULL,
DATE_NAISSANCE DATE DEFAULT NULL,
LIEN_PARENTE CHAR(20) DEFAULT NULL,
PRIMARY KEY (MATRICULE,NOM));

CREATE TABLE TRAVAILLE_DANS(
MATRICULE CHAR(4) NOT NULL,
NUM_P CHAR(4) NOT NULL,
NBRE_HEURE NUMBER(4) NOT NULL,
PRIMARY KEY (MATRICULE,NUM_P));
```

b) Remplissage de la Base de Méta Connaissances Floues

La création des tables en langage FSQL nécessite la mise à jour de la BMCF, par conséquent, nous exposons dans ce qui suit les scripts nécessaires pour cette mise à jour.

– Définition des types d'attributs : Types 1, 2 et 3

```
INSERT into FCL values (EMPLOYÉ,ÂGE,2,1,USER||'.EMPLOYÉ.ÂGE');
INSERT into FCL values (EMPLOYÉ,SALAIRE,1,1,USER||'.EMPLOYÉ.SALAIRE');
INSERT into FCL values (EMPLOYÉ,ÉTUDES,3,1,USER||'.EMPLOYÉ.ÉTUDES');
INSERT into FCL values (EMPLOYÉ,RENDEMENT,3,1,USER||'.EMPLOYÉ.RENDEMENT');
INSERT into FCL values (EMPLOYÉ,EXPÉRIENCE,2,1,USER||'.EMPLOYÉ.EXPÉRIENCE');
INSERT into FCL values (DÉPARTEMENT,LOCAL,3,1,USER||'.DÉPARTEMENT.LOCAL');
INSERT into FCL values (TRAVAILLE_DANS,NBRE_HEURE,1,1,USER||'.TRA_DANS.NBRE_HEURE');
INSERT into FCL values (PROJET,BUDGET,1,1,USER||'.PROJET.BUDGET');
INSERT into FCL values (POSTE,PROFESSION,3,1,USER||'.POSTE.PROFESSION');
INSERT into FCL values (POSTE,ÉTUDES,3,1,USER||'.POSTE.ÉTUDES');
INSERT into FCL values (POSTE,EXPÉRIENCE,2,1,USER||'.POSTE.EXPÉRIENCE');
INSERT into FCC values (EMPLOYÉ,ÉTUDES,POSTE,ÉTUDES);
```

– Définition des valeurs "marge" et "much"

INSERT into FAM values(EMPLOYÉ,ÂGE,5,10) ;

INSERT into FAM values(EMPLOYÉ,SALAIRE,10,50) ;

INSERT into FAM values(EMPLOYÉ,EXPÉRIENCE,2,5) ;

INSERT into FAM values(DÉPARTEMENT,NBRE_EMPLOYÉ,5,10) ;

INSERT into FAM values(TRAVAILLE_DANS,NBRE_HEURE,2,4) ;

INSERT into FAM values(PROJET,BUDGET,100,1000) ;

INSERT into FAM values(POSTE,EXPÉRIENCE,2,5) ;

c) Définition des étiquettes linguistiques en langage FSQL

Pour définir les étiquette linguistiques, Galindo utilise 2 nouvelles commandes du langage FSQL : CREATE LABEL (pour les attributs FTYPE1 et FTYPE2) et CREATE NEARNESS (pour les attributs FTYPE3). Les valeurs de ces étiquettes sont entreposées dans la BMCF. Nous présentons tout d'abord le script décrit en FSQL pour la définition de ces étiquettes, ensuite, leur modélisation dans la BMCF :

La table EMPLOYÉ

CREATE LABEL JEUNE ON EMPLOYÉ.ÂGE VALUES 18,22,30,35 ;
CREATE LABEL ADULTE ON EMPLOYÉ.ÂGE VALUES 25,32,45,50 ;
CREATE LABEL VIEUX ON EMPLOYÉ.ÂGE VALUES 50,55,62,70 ;
CREATE LABEL PETITE ON EMPLOYÉ.EXPÉRIENCE VALUES 2,3,5,6 ;
CREATE LABEL BONNE ON EMPLOYÉ.EXPÉRIENCE VALUES 5,7,10,12 ;
CREATE LABEL SUFFISANTE ON EMPLOYÉ.EXPÉRIENCE VALUES 7,8,15,20 ;
CREATE LABEL BAS ON EMPLOYÉ.SALAIRE VALUES 50,80,120,180 ;
CREATE LABEL MOYEN ON EMPLOYÉ.SALAIRE VALUES 150,300,400,550 ;
CREATE LABEL ÉLEVÉ ON EMPLOYÉ.SALAIRE VALUES 400,600,800,1000 ;

CREATE NEARNESS ON EMPLOYÉ.RENDEMENT LABELS MAUVAIS, REGULIER, BON VALUES 0.3, 0.2, 0.7 ;

La table PROJET

CREATE LABEL PETIT ON PROJET.BUDGET VALUES 100,500,1000,1800 ;
CREATE LABEL MOYEN ON PROJET.BUDGET VALUES 1500,5000,10000,15000 ;
CREATE LABEL GROS ON PROJET.BUDGET VALUES 12000,20000,50000,70000 ;

La table POSTE

CREATE LABEL PETITE ON POSTE.EXPÉRIENCE VALUES 2,3,5,6 ;
CREATE LABEL SUFFISANTE ON POSTE.EXPÉRIENCE VALUES 5,7,10,12 ;
CREATE LABEL BONNE ON POSTE.EXPÉRIENCE VALUES 7,8,15,20 ;

CREATE NEARNESS ON POSTE.ÉTUDES LABELS SECONDAIRE, DIPLÔME, LICENCE, DOCTORAT VALUES .5, .1, .05, .6, .2, .7 ;

CREATE NEARNESS ON POSTE.PROFESSION LABELS DIRECTEUR, INGÉNIEUR, TECHNICIEN, ADMINISTRATIF, SECRÉTAIRE, VALUES .8, .5, .6,.4, .8, .2, .2, .8, .5, .7 ;

CREATE NEARNESS ON DÉPARTEMENT.LOCAL LABELS CENTRE_VILLE, BARDO, ARIANA, BEN_AROUS, NABEUL, SOUSSE VALUES .95, .7, .6, .4, .2, .8, .5, .3, .2, .5, .3, .1, .7, .4, .3 ;

La table TRAVAILLE_DANS

CREATE LABEL PETIT ON TRAVAILLE_DANS.NBRE_HEURE VALUES 0,2,7,8 ;

CREATE LABEL MOYEN ON TRAVAILLE_DANS.NBRE_HEURE VALUES 5,10,15,20 ;

CREATE LABEL ÉLEVÉ ON TRAVAILLE_DANS.NBRE_HEURE VALUES 15,20,25,30 ;

d) Remplissage de la Base de Méta Connaissances Floues

La traduction des commandes CREATE LABEL et CREATE NEARNESS se fait directement dans la BMCF. Cette traduction consiste à insérer dans la table FOL les tuples suivants pour :

La table EMPLOYÉ :

INSERT into FOL values(EMPLOYÉ,ÂGE,0,'JEUNE',0) ;
INSERT into FOL values(EMPLOYÉ,ÂGE,1,'ADULTE',0) ;
INSERT into FOL values(EMPLOYÉ,ÂGE,2,'VIEUX',0) ;
INSERT into FOL values(EMPLOYÉ,SALAIRE,0,'BAS',0) ;
INSERT into FOL values(EMPLOYÉ,SALAIRE,1,'MOYEN',0) ;
INSERT into FOL values(EMPLOYÉ,SALAIRE,2,'ÉLEVÉ',0) ;
INSERT into FOL values(EMPLOYÉ,ÉTUDES,0,'SECONDAIRE',1) ;
INSERT into FOL values(EMPLOYÉ,ÉTUDES,1,'DIPLÔME',1) ;
INSERT into FOL values(EMPLOYÉ,ÉTUDES,2,'LICENCIE',1) ;
INSERT into FOL values(EMPLOYÉ,ÉTUDES,3,'DOCTEUR',1) ;
INSERT into FOL values(EMPLOYÉ,RENDEMENT,0,'MAUVAIS',1) ;
INSERT into FOL values(EMPLOYÉ,RENDEMENT,1,'RÉGULIER',1) ;
INSERT into FOL values(EMPLOYÉ,RENDEMENT,2,'BON',1) ;
INSERT into FOL values(EMPLOYÉ,EXPÉRIENCE,0,'PETITE',0) ;
INSERT into FOL values(EMPLOYÉ,EXPÉRIENCE,1,'SUFFISANTE',0) ;
INSERT into FOL values(EMPLOYÉ,EXPÉRIENCE,2,'BONNE',0) ;

La table DÉPARTEMENT :

INSERT into FOL values(DÉPARTEMENT,LOCAL,0,'Centre_ville',0) ;
INSERT into FOL values(DÉPARTEMENT,LOCAL,1,'Bardo',1) ;
INSERT into FOL values(DÉPARTEMENT,LOCAL,2,'Ariana',1) ;
INSERT into FOL values(DÉPARTEMENT,LOCAL,3,'Ben_Arous',1) ;
INSERT into FOL values(DÉPARTEMENT,LOCAL,4,'NAbeul',1) ;
INSERT into FOL values(DÉPARTEMENT,LOCAL,5,'Sousse',1) ;

La table TRAVAILLE_DANS :

INSERT into FOL values(TRAVAILLE_DANS,NBRE_HEURE,0,'PETIT',0) ;
INSERT into FOL values(TRAVAILLE_DANS,NBRE_HEURE,1,'MOYEN',0) ;
INSERT into FOL values(TRAVAILLE_DANS,NBRE_HEURE,2,'ÉLEVÉ',0) ;

La table PROJET :

INSERT into FOL values(PROJET,BUDGET,0,'PETIT',0) ;
INSERT into FOL values(PROJET,BUDGET,1,'MOYEN',0) ;
INSERT into FOL values(PROJET,BUDGET,2,'GROS',0) ;

La table POSTE :

INSERT into FOL values(POSTE,PROFESSION,0,'DIRECTEUR',1) ;
INSERT into FOL values(POSTE,PROFESSION,1,'INGÉNIEUR',1) ;
INSERT into FOL values(POSTE,PROFESSION,2,'TECHNICIEN',1) ;
INSERT into FOL values(POSTE,PROFESSION,3,'ADMINISTRATIF',1) ;
INSERT into FOL values(POSTE,PROFESSION,4,'SECRÉTAIRE',1) ;
INSERT into FOL values(POSTE,ÉTUDES,0,'SECONDAIRE',1) ;
INSERT into FOL values(POSTE,ÉTUDES,1,'DIPLÔME',1) ;
INSERT into FOL values(POSTE,ÉTUDES,2,'LICENCE',1) ;
INSERT into FOL values(POSTE,ÉTUDES,3,'DOCTORAT',1) ;
INSERT into FOL values(POSTE,EXPÉRIENCE,0,'PETITE',0) ;
INSERT into FOL values(POSTE,EXPÉRIENCE,1,'SUFFISANTE',0) ;
INSERT into FOL values(POSTE,EXPÉRIENCE,2,'BONNE',0) ;

La création des étiquettes linguistiques définies sur les attributs FTYPE1 et FTYPE2 s'effectue dans la table FLD, et ceci pour entreposer les valeurs de chaque étiquettes linguistique. Elle possède la forme suivante :

INSERT into FLD values(EMPLOYÉ,ÂGE,0,18,22,30,35) ;
INSERT into FLD values(EMPLOYÉ,ÂGE,1,25,32,45,50) ;
INSERT into FLD values(EMPLOYÉ,ÂGE,2,50,55,62,70) ;
INSERT into FLD values(EMPLOYÉ,SALAIRE,0,50,80,120,180) ;
INSERT into FLD values(EMPLOYÉ,SALAIRE,1,150,300,400,550) ;
INSERT into FLD values(EMPLOYÉ,SALAIRE,2,400,600,800,1000) ;
INSERT into FLD values(EMPLOYÉ,EXPÉRIENCE,1,2,3,5,6) ;
INSERT into FLD values(EMPLOYÉ,EXPÉRIENCE,2,5,7,10,12) ;
INSERT into FLD values(EMPLOYÉ,EXPÉRIENCE,3,7,8,15,20) ;

```
INSERT into FLD values(TRAVAILLE_DANS,NBRE_HEURE,0,0,2,7,8) ;
INSERT into FLD values(TRAVAILLE_DANS,NBRE_HEURE,1,5,10,15,20) ;
INSERT into FLD values(TRAVAILLE_DANS,NBRE_HEURE,2,15,20,25,30) ;
INSERT into FLD values(PROJET,BUDGET,0,100,500,400,550) ;
INSERT into FLD values(PROJET,BUDGET,1,1500,5000,10000,15000) ;
INSERT into FLD values(PROJET,BUDGET,2,12000, 20000,50000,70000) ;
INSERT into FLD values(POSTE,EXPÉRIENCE,0,2,3,5,6) ;
INSERT into FLD values(POSTE,EXPÉRIENCE,1,5,7,10,12) ;
INSERT into FLD values(POSTE,EXPÉRIENCE,2,7,8,15,20) ;
```

La création des relations de similitudes définies sur les attributs FTYPE3 s'effectue dans la table FND. Elle possède la forme suivante :

```
INSERT into FND values(EMPLOYÉ,RENDEMENT,0,1,.3) ;
INSERT into FND values(EMPLOYÉ,RENDEMENT,0,2,.2) ;
INSERT into FND values(EMPLOYÉ,RENDEMENT,1,2,.7) ;
INSERT into FND values(POSTE,ÉTUDES,0,1,.5) ;
INSERT into FND values(POSTE,ÉTUDES,0,2,.1) ;
INSERT into FND values(POSTE,ÉTUDES,0,3,.05) ;
INSERT into FND values(POSTE,ÉTUDES,1,2,.6) ;
INSERT into FND values(POSTE,ÉTUDES,1,3,.2) ;
INSERT into FND values(POSTE,ÉTUDES,2,3,.7) ;
INSERT into FND values(POSTE,PROFESSION,0,1,.8) ;
INSERT into FND values(POSTE,PROFESSION,0,2,.5) ;
INSERT into FND values(POSTE,PROFESSION,0,3,.6) ;
INSERT into FND values(POSTE,PROFESSION,0,4,.4) ;
INSERT into FND values(POSTE,PROFESSION,1,2,.8) ;
INSERT into FND values(POSTE,PROFESSION,1,3,.2) ;
INSERT into FND values(POSTE,PROFESSION,1,4,.2) ;
INSERT into FND values(POSTE,PROFESSION,2,3,.8) ;
INSERT into FND values(POSTE,PROFESSION,2,4,.5) ;
INSERT into FND values(POSTE,PROFESSION,3,4,.7) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,0,1,.95) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,0,2,.7) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,0,3,.6) ;
```

INSERT into FND values(DÉPARTEMENT,LOCAL,0,4,.4) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,0,5,.2) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,1,2,.8) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,1,3,.5) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,1,4,.3) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,1,5,.2) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,2,3,.5) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,2,4,.3) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,2,5,.1) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,3,4,.7) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,3,5,.4) ;
INSERT into FND values(DÉPARTEMENT,LOCAL,4,5,.3) ;

4.3 Processus d'implémentation d'un script FSQL

Pour expliquer les transformations décrites ci-dessus, nous prenons comme exemple la table EMPLOYÉ puisqu'elle contient les trois types d'attributs flous. Pour ceci, nous considérons uniquement le schémas suivant de la table :
EMPLOYÉ (MATRICULE, NOM, PRÉNOM, ADRESSE, ÂGE, SALAIRE, RENDEMENT). La création de la table Employé en langage FSQL sera comme suit :
CREATE TABLE **EMPLOYÉ**(
MATRICULE CHAR(4) NOT NULL,
NOM CHAR(20) NOT NULL,
PRENOM CHAR(20) NOT NULL,
ADRESSE CHAR(40) NOT NULL,
ÂGE FTYPE2(5,10) NUMBER(3) DEFAULT UNKNOWN NOT NULL,
SALAIRE FTYPE1(10,50) NUMBER(7) NOT NULL,
RENDEMENT FTYPE3(1) NOT NULL,
PRIMARY KEY (MATRICULE)) ;
Si nous donnons une représentation informelle de la table EMPLOYÉ, elle sera comme suit :

MATRICULE	NOM	PRENOM	ADRESSE	ÂGE	SALAIRE	RENDEMENT
classique	classique	classique	classique	(FTYPE2)	(FTYPE1)	(FTYPE3)

TAB. 4.5 – Extension de la table EMPLOYÉ

4.3.1 CREATE TABLE

La traduction de cette commande subit un traitement dans la BD et un traitement dans la BMCF. Comme c'est vu précédemment, CREATE TABLE se traduit dans la BD par :
CREATE TABLE **EMPLOYÉ**(
MATRICULE VARCHAR2(4) NOT NULL,
NOM VARCHAR2(20) NOT NULL,
PRENOM VARCHAR2(20) NOT NULL,
ADRESSE VARCHAR2(40) NOT NULL,
AGE**T** NUMBER(1) DEFAULT **0** NOT NULL
CHECK (AGET BETWEEN 0 AND 7),
ÂGE1 NUMBER(3),
ÂGE2 NUMBER(3),
ÂGE3 NUMBER(3),
ÂGE4 NUMBER(3),
SALAIRE NUMBER(7) NOT NULL,
RENDEMENT**T** NUMBER(1) NOT NULL
CHECK (RENDEMENTT BETWEEN 0 AND 4),
RENDEMENTP1 NUMBER(3,2),
RENDEMENT1 NUMBER(3) NOT NULL,
PRIMARY KEY (MATRICULE)) ;

La table 4.6 montre une représentation de la EMPLOYÉ avec ses attribut flous.

MATRICULE	NOM	PRENOM	ADRESSE	AGET	AGE1	AGE2

AGE3	AGE4	SALAIRE	RENDEMENTT	RENDEMENTP1	RENDEMENTP1

TAB. 4.6 – La table EMPLOYE

Nous remarquons bien que l'attribut âge s'est transformé en 5 attributs : le premier ayant le nom de l'attribut post fixé par la lettre `T` en majuscule et les quatre derniers ayant les le nom de l'attribut post fixé respectivement par les chiffres `1, 2, 3` et `4`. Dans la BMCF, CREATE TABLE se nous mène à sauvegarder les informations concernant ces attributs :

1. **Modification au niveau la table FCL**

   Nous insérons une ligne dans la table FCL, pour chaque attribut flou (attribut de type FTYPE1, FTYPE2 FTYPE3). Cette ligne (Ce tuple) précise *le nom de l'attribut, le nom de la table correspondant, un numéro indiquant son type* (1 pour FTYPE1, 2 pour FTYPE2, 3 pour FTYPE3), *Len* qui sera généralement égale à 1 sauf pour le cas des attributs de type FTYPE3 (nombre de distribution de possibilité), et *un commentaire optionnel* d'explication.

   Dans notre exemple la table FCL sera comme suit :

OBJ#	**COL#**	**F_TYPE**	**LEN**	**COM**
EMPLOYÉ	ÂGE	2	1	USER||'.EMPLOYÉ.ÂGE'
EMPLOYÉ	SALAIRE	1	1	USER||'.EMPLOYÉ.SALAIRE'
EMPLOYÉ	RENDEMENT	3	1	USER||'.EMPLOYÉ.RENDEMENT'

TAB. 4.7 – La table FUZZY_COL_LIST (FCL)

   Ceci, revient à écrire le script SQL suivant :

   INSERT into FCL values (EMPLOYÉ, ÂGE, 2, 1, USER ||'.EMPLOYÉ.ÂGE') ;

   INSERT into FCL values (EMPLOYÉ, SALAIRE, 1,1, USER ||'.EMPLOYÉ. SALAIRE') ;

   INSERT into FCL values (EMPLOYÉ,RENDEMENT, 3, 1, USER||'.EMPLOYÉ. RENDEMENT') ;

2. **Modification au niveau la table FAM**

   Cette modification ne concerne que les attributs de type FTYPE1 ou FTYPE2. Nous insérons une ligne dans la table FAM, pour chaque attribut de type FTYPE1, FTYPE2.

   Cette ligne (Ce tuple) précise, *le nom de l'attribut, le nom de la table correspondant*, la valeur *marge* et la valeur *much* correspondant à cet attribut.

   Comme précisé ci-dessus l'attribut ÂGE d'un employé a une valeur approximative d'une marge de 5. La valeur minimale pour considérer deux valeurs de cet attribut comme totalement différent est de 10. Ceci implique que la valeur de marge =5 et much =10.

   Dans notre exemple la table FAM sera comme suit :

   Ceci, revient à écrire le script SQL suivant :

   INSERT into FAM values (EMPLOYÉ, ÂGE, 5, 10) ;

   INSERT into FAM values (EMPLOYÉ, SALAIRE, 10, 50) ;

OBJ#	COL#	MARGE	MUCH
EMPLOYÉ	ÂGE	5	10
EMPLOYÉ	SALAIRE	10	50

TAB. 4.8 – La table FUZZY_APPROX_MUCH (FAM)

3. **Modification au niveau la table FCC**
   Cette modification ne concerne que les attributs de type FTYPE3 qui sont compatibles entre eux. Ceci est pour éviter la répétition d'entreposer les étiquettes linguistiques définis sur des attributs flous FTYPE3.
   Nous insérons dans la table FCC, autant de lignes que d'attributs flous de type FTYPE3. Cette ligne (Ce tuple) précise, *le nom de l'attribut, le nom de la table correspondant, le nom de l'attribut compatible* et *le nom de la table correspondant.* Dans notre exemple et puisque l'attribut RENDEMENT n'est compatible avec aucun autre attribut de son type, la table FCC reste vide.

4.3.2 CREATE LABEL

Cette commande est traitée dans la BMCF.

1. **Modification au niveau la table FOL**
   Nous insérons, dans la table FOL, pour chaque attribut flou, tant de lignes que d'étiquettes linguistiques définis sur lui. Cette ligne (Ce tuple) précise le *nom de l'attribut, le nom de la table* correspondant, *le nom de l'étiquette linguistique, un numéro interne* pour l'identifier (nous commençons toujours par zéro), ainsi que *le type de sa représentation* (= 0 pour les distribution trapézoïdales des attributs FTYPE1 et FTYPE2, 1 pour les attributs FTYPE3, etc.)
   Dans notre exemple et pour les étiquettes linguistiques définis sur les attributs SALAIRE, ÂGE et RENDEMENT, la table FOL sera comme suit :
   Le script correspondant à la création de l'étiquette jeune de l'attribut âge est :
   INSERT into FOL values (EMPLOYÉ, ÂGE, 0, 'JEUNE', 0) ;
2. **Modification de la table FLD**
   Cette modification ne concerne que les attributs de type FTYPE1 ou FTYPE2. Pour chaque étiquette linguistique définis pour un attribut, nous précisons la distribution de possibilité trapézoïdale correspondante. Nous insérons donc, dans la table FLD (table 4.10), pour chaque attribut flou, tant de lignes que d'étiquettes linguistiques définis sur cet attribut. Cette ligne (Ce tuple) précise

OBJ#	COL#	FUZZY_ID	FUZZY_NAME	FUZZY_TYPE
EMPLOYÉ	ÂGE	0	'JEUNE'	0
EMPLOYÉ	ÂGE	1	'ADULTE'	0
EMPLOYÉ	ÂGE	2	'VIEUX'	0
EMPLOYÉ	SALAIRE	0	'BAS'	0
EMPLOYÉ	SALAIRE	1	'MOYEN'	0
EMPLOYÉ	SALAIRE	2	'ELEVE'	0
EMPLOYÉ	RENDEMENT	0	'MAUVAIS'	1
EMPLOYÉ	RENDEMENT	1	'REGULIER'	1
EMPLOYÉ	RENDEMENT	2	'BON'	1

TAB. 4.9 – La table FUZZY_OBJETC_LIST (FOL)

le nom de l'attribut, le nom de la table correspondant, l'identifiant l'étiquette linguistique (défini dans la table FOL) la valeur *ALPHA, BETA, GAMMA, DELTA* correspondant aux quatre valeurs du trapèze.
Comme il a été précisé dans la description des différents attributs, l'attribut âge d'un employé possède pour l'étiquette linguistique jeune (dont le numéro interne est précisé dans la table FOL par 0) et la distributions de possibilité trapézoïdale suivante : 18, 22, 30, 35.

OBJ#	COL#	FUZZY_ID	ALFA	BETA	GAMMA	DELTA
EMPLOYÉ	ÂGE	0	18	22	30	35
EMPLOYÉ	ÂGE	1	25	32	45	50
EMPLOYÉ	ÂGE	2	50	55	62	70
EMPLOYÉ	SALAIRE	0	50	80	120	180
EMPLOYÉ	SALAIRE	1	150	300	400	550
EMPLOYÉ	SALAIRE	2	400	600	800	1000

TAB. 4.10 – La table FUZZY_LABEL_DEF (FLD)

Le script correspondant à la définition des valeurs de l'étiquette jeune de l'attribut âge est :
INSERT into FLD values (EMPLOYÉ, ÂGE, 0, 18, 22, 30, 35) ;

4.3.3 CREATE NEARNESS

Cette modification ne concerne que les attributs de type FTYPE3. Pour chaque étiquette linguistique définie sur cet attribut, nous précisons le degré de ressemblance de cette étiquette avec les autres étiquettes définies sur ce même attribut. Nous insérons donc, dans la table FND, $n^2/2 - n/2$ tuple (/ représente la division entière et n représente le nombre d'étiquette linguistique défini sur l'attribut en question). Chaque ligne précise le nom de l'attribut, le nom de la table correspondant, l'identifiant de la ième étiquette[2], l'identifiant de la jème étiquette et le degré de ressemblance défini entre elles (i varie entre 0 et $n^2/2 - n/2$ et j).

Dans notre exemple, l'attribut rendement d'un employé possède pour les étiquettes linguistiques mauvais, régulier et bon identifiées respectivement dans la table FOL par 0, 1 et 2, les degrés de ressemblance données par δ(mauvais, régulier) = 0.3[3], δ(mauvais,bon) = 0.2 et δ(régulier,bon) = 0.7.
Par suite, la table FND sera comme suit :

OBJ#	**COL#**	**FUZZY_ID1**	**FUZZY_ID2**	**DEGREE**
EMPLOYÉ	RENDEMENT	0	1	0.3
EMPLOYÉ	RENDEMENT	0	2	0.2
EMPLOYÉ	RENDEMENT	1	2	0.7

TAB. 4.11 – La table FUZZY_NEARNESS_DEF (FND)

Pour les étiquettes définies sur l'attribut rendement, nous devons écrire le script SQL suivant :
INSERT into FND values (EMPLOYÉ, RENDEMENT, 0, 1, 0.3) ;
INSERT into FND values (EMPLOYÉ, RENDEMENT, 0, 2, 0.2) ;
INSERT into FND values (EMPLOYÉ, RENDEMENT, 1, 2, 0.7) ;

La figure 4.3 résume les changements effectués au niveau de la BMCF.

[2]L'identifiant de chaque étiquette est défini dans la table FOL

[3]Pour représenter ces valeurs, l'étiquette mauvais sera remplacée par son identifiant 0, l'étiquette régulier par son identifiant 1 et par suite FUZZY_ID1 = 0, FUZZY_ID2 =1 et DEGREE = 0.3

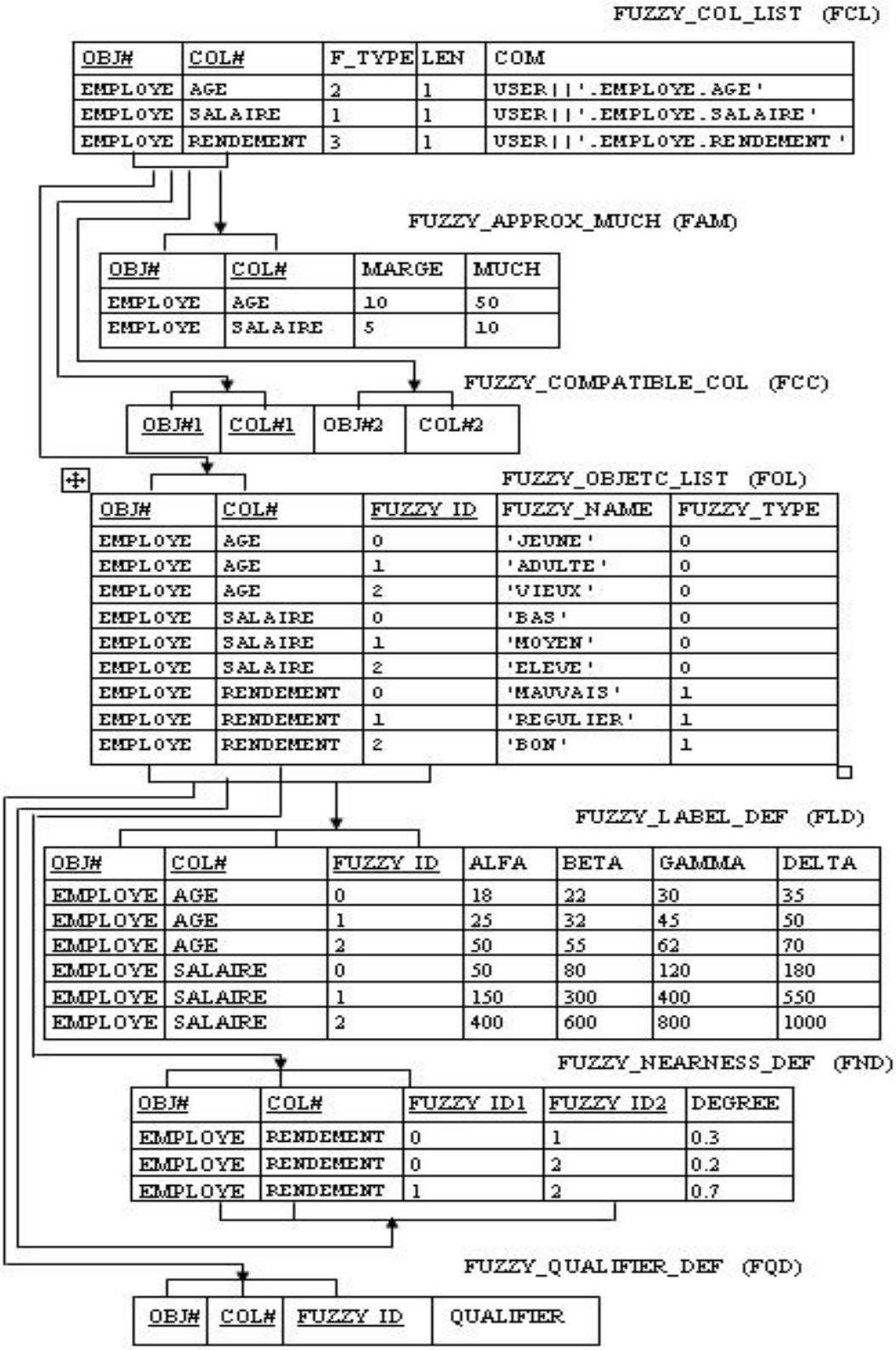

FIG. 4.3 – Mise à jour de la BMCF

4.4 Limite de FQ

Comme nous venons de le montrer, nous pouvons dire que l'étape de transformation d'une BDF décrite en FSQL en des scripts correspondants en SQL pour qu'ils puissent être gérés par un SGBD classique n'est pas évidente. Elle demande une bonne description des différentes opérations à faire au niveau de la BD et au niveau de la BMCF. Ceci est pour une simple BD. Cette opération devient de plus en plus lourde et délicate si la BD devient très grande.
A notre avis, l'utilisation du logiciel FQ sera par conséquent limitée à des exemples simples de BDF, vu que toute la transformation est être faite manuellement par le concepteur de la BD.

4.5 Conclusion

Nous avons présenté, dans ce chapitre, un exemple d'implémentation d'une BDF. Nous avons détaillé cette implémentation pour montrer les difficultés que peut engendrer une telle tâche sachant qu'elle est faite manuellement. Ceci nous a poussé à définir un outil qui gère la transformation du script FSQL à son équivalent en SQL pour qu'il soit supporté par le SGBD Oracle. Cet outil sera présenté dans le chapitre suivant.

Chapitre 5

Nouvelle approche de description et de modélisation des BDF

5.1 Introduction

Nous présentons dans ce chapitre une nouvelle approche de description et de manipulation des BDF. L'utilisateur a la possibilité, dans ce cas, de définir les différents types d'attributs floues tels que FTYPE1, FTYPE2 et FTYPE3, les mots clés introduits par FSQL tels que CREATE LABEL, CREATE NEARNESS, UNKNOWN,... Cette approche offre un outil qui génère automatiquement l'équivalent d'un script FSQL en un script SQL2 et remplit les tables nécessaires de la BMCF.
Cette nouvelle approche nous permet d'enrichir le logiciel FQ pour qu'il puisse supporter toutes les opérations de description et de manipulation des données floues. Nous présentons tout d'abord l'architecture de FIRST étendue que nous avons proposé. Ensuite, nous détaillons les étapes et les règles à suivre qui ont mené à l'implémentation de notre outil. Nous finirons par donner quelques exemples de requêtes floues modélisées dans FQ.

5.2 Architecture proposée

Nous proposons dans cette partie, une extension de l'architecture FIRST afin de supporter la création et la manipulation d'une BDF. Dans la nouvelle architecture, nous avons ajouté une couche (FSQL_TO_SQL) qui assure l'interface entre un script LDD d'une BDF modélisée en FSQL et celui correspondant en SQL avec la MAJ de la BMCF engendrés.

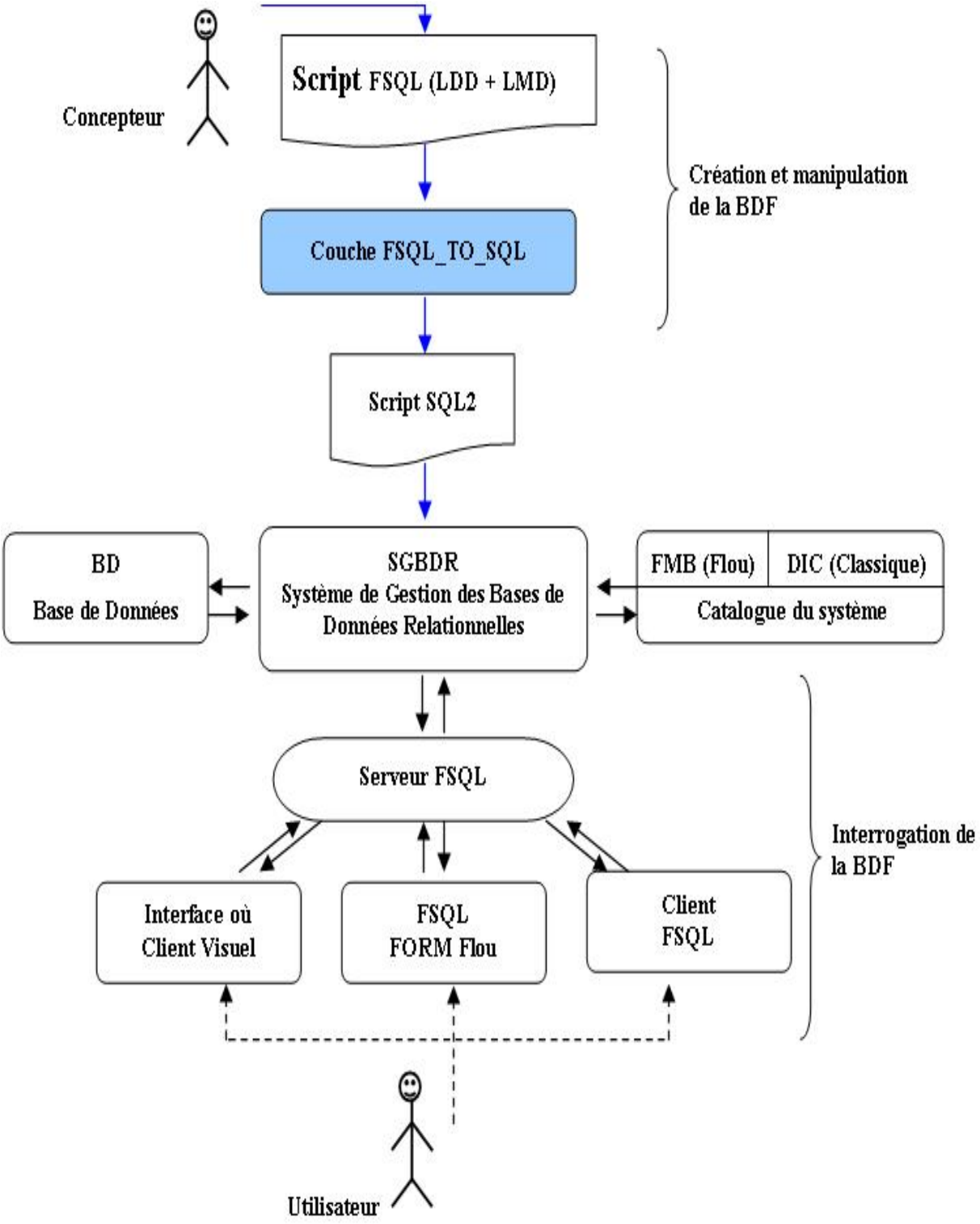

FIG. 5.1 – Architecture FIRST étendue

Ainsi, un utilisateur peut interagir avec la BDF par les deux sens : le premier consulte la BDF (simple utilisateur) et le deuxième manipule et crée la BDF (concepteur).

5.2.1 Présentation de la couche FSQL_TO_SQL

Cette couche présente un outil qui permet l'implémentation d'une BDF décrite en FSQL sous oracle 8i. Cet outil doit donner les transformations du script FSQL et les modifications qui doivent êtres faites au niveau de la BMCF.

Principe de fonctionnement

L'idée principale consiste à découper la commande décrite en FSQL en plusieurs lignes contenant chacune un attribut. En deuxième lieu nous étudions cette ligne : si elle contient un attribut classique, nous la copions dans un premier fichier (résultat1.sql). Sinon, nous spécifions un traitement bien spécifique à chaque attribut (FTYPE1, FTYPE2, FTYPE3). Ce traitement se divise en deux sous traitements, le premier consiste à traduire la commande qui concerne la BD. La commande traduite est écrite dans le fichier résultat1. Le deuxième traitement fournit un script écrit dans le fichier résultat2, contenant les informations dans la BMCF concernant les attributs flous et les objets définis sur eux.

Comme résultat de ces deux traitements, nous aurons deux fichiers : un fichier contenant la partie LDD de la BD et un deuxième contenant la modification à effectuer dans la BMCF. Ce processus de fonctionnement est illustré dans la figure 5.2. Nous pouvons aussi regrouper les deux traitement dans un même fichier puisqu'ils seront exécutés dans le SGBD.

5.2.2 Règles à suivre

Après une étude détaillée du modèle GEFRED et de l'outil présenté par Medina, nous avons pu définir un ensemble d'étapes chacune présente des règles que le logiciel doit appliquer pour réussir notre transformation. Les étapes à suivre pour effectuer la traduction des principales commandes LDD de type CREATE vers le langage SQL sont les suivantes : Pour traduire la commande CREATE TABLE de FSQL en langage SQL, nous devons :

Etape 1 : Faire appel à la commande `CREATE TABLE` classique avec une modification des champs contenant les attributs flous.

Etape 2 : `Insérer les tuples dans la BMCF` contenant les attributs flous définis sur cette table.

Etape 3 : Modéliser les différentes commandes `CREATE LABEL, CREATE NEARNESS dans la BMCF`

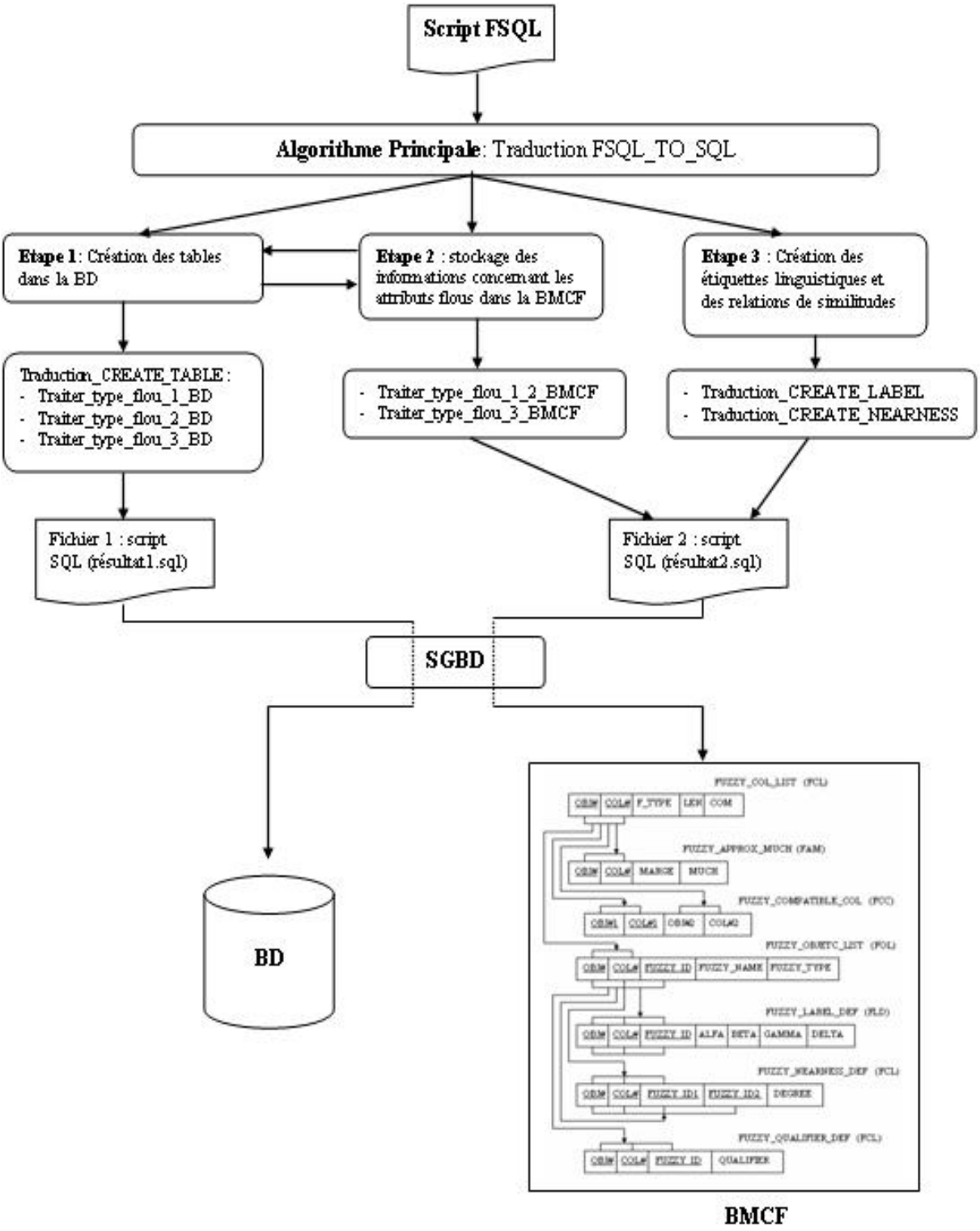

FIG. 5.2 – Fonctionnement de FSQL_TO_SQL

La commande CREATE TABLE

Cette commande permet de créer une table avec des attributs flous. Pour ceci , elle va associer à chaque type d'attribut un traitement spécifique selon les règles décrites dans les tableaux 5.1 et 5.2.

Type d'attribut	**Règles à suivre dans le traitement des attributs flous au niveau de la BD**
FTYPE1	1) supprimer **FTYPE1** de la syntaxe de la commande
FTYPE2	1) supprimer **FTYPE2** de la syntaxe de la commande 2) concaténer l'attribut avec la lettre '**T**' suivi de la restriction de ses valeurs qu'il doivent être dans l'intervalle [0,7] 3) concaténer l'attribut respectivement avec 1, 2, 3 et 4 (Attribut**1**, . . . , Attribut**4**) et par suite l'attribut se transforme en 5 attributs. 4) Ajouter dans la même ligne son type de base (type classique défini après FTYPE par défaut NUMBER) 5) traduire le mot DEFAULT selon la table de représentation 3.1 6) traduire la restriction selon la table 3.1.
FTYPE3	1) supprimer **FTYPE3** de la syntaxe de la commande 2) concaténer l'attribut avec la lettre '**T**' suivi de la restriction de ses valeurs qui doivent être dans l'intervalle [0,4] 3) concaténer l'attribut **n** fois respectivement avec Pi puis avec i, sachant que n est fourni entre parenthèse après le mot FTYPE3 et $1 \leq i \leq n$ 4) Ajouter son type de base (par défaut c'est NUMBER(3,2) pour Attribut**Pi** et NUMBER(3) pour Attribut**i** et par suite l'attribut se transforme en 2n+1 attributs. 5) traduire le mot DEFAULT selon la table 3.2 . 6) traduire la restriction selon la table 3.2 .
Classique	Rien à faire

TAB. 5.1 – CREATE TABLE dans la BD

Type d'attribut	**Règles à suivre dans le traitement des attributs flous au niveau de la BMCF**
FTYPE1, FTYPE2 FTYPE3	1) sauvegarder dans la table **FCL** les informations sur l'attribut (nom, type, longueur et un commentaire).
FTYPE1, FTYPE2	2) stocker les valeurs entre parenthèses qui suivent FTYPE1 (**n,m**) dans la table **FAM.**
FTYPE3	2) sauvegarder dans la table **FCC** les attributs compatibles avec lui
Classique	Rien à faire

TAB. 5.2 – CREATE TABLE dans la BMCF

Commande CREATE LABEL

Cette commande permet de créer une étiquette linguistique. Pour ce faire, il faut respecter les règles suivantes :

1. Insérer dans la table **FOL** l'identifiant de l'étiquette linguistique (Label), son nom et son type.
2. Insérer dans la table **FLD** les paramètres de l'étiquette linguistique $(\alpha, \beta, \gamma, \delta)$.

Commande CREATE NEARNESS

Cette commande permet de créer une relation de similitude. Pour ce faire, il faut respecter les règles suivantes :

1. Insérer dans la table **FOL** l'identifiant de l'étiquette linguistique (Label), son nom et son type.
2. Insérer dans la table **FND** la liste des étiquettes linguistiques avec leurs degrés de similitudes.

5.2.3 ALGORITHME

Nous présentons dans ce qui suit un résumé des principaux algorithmes utilisés dans notre application. Ces algorithmes permettent de créer les objets : TABLE, LABEL et NEARNESS. Les autres algorithmes sont présentés dans l'annexe.
L'algorithme principal ouvre un fichier, extrait de ce fichier les chaines de caractères, puis associe à chaque chaine, un traitement suivant la commande qu'elle contient.

A la fin de la traduction, il crée deux fichiers résultats comme il a été précisé dans la section précédente.

```
Algorihme Traduction_FSQL_à_SQL(Var s : cc[1] résultat1, résultat2, source :fichier)
Début
ouvrir_fichier (source.txt, lecture)
ouvrir_fichier (résultat1.sql, écriture)
ouvrir_fichier (résultat2.sql, écriture)
tanque (source <>fin) faire
lire_chaine(s,source)
si(s contient CREATE TABLE) alors
      Traduction_CREATE_TABLE(s, source, résultat1,résultat2)
      Passer_après_point_virgule(source)
sinon
      si (s contient CREATE LABEL) alors
         Traduction_CREATE_LABEL(s, source, résultat2, nom_label)
          Passer_après_point_virgule(source)
      sinon
         si (s contient CREATE NEARNESS) alors
           Traduction_CREATE_NEARNESS(s, source, résultat2)
           Passer_après_point_virgule(source)
         sinon
           écrire_dans_fichier(résultat1,s)
         finsi
      finsi
finsi
fintq
fermer_fichier(résultat1,résultat2,source)
connecter_à_SQLplus()
exécuter(résultat1.sql)
exécuter(résultat2.sql)
Fin
```

Procédure 1. *La procédure Traduction_ CREATE_ TABLE permet de traduire la*

[1] cc est une abréviation pour chaîne de caractère

création d'une table en associant à chaque attribut flou, deux traitements au niveau de la BD et au niveau de la BMCF.

```
Procédure Traduction_CREATE_TABLE(s :cc, source :fichier, Var résultat1,
résultat2 : fichier)
Var a1, a2, b1, b2, number : entier
nom_table, nom_attribut : cc
Début
tanque(s n'a pas atteint point virgule) faire
nom_table ← chercher_table(s)
type ← test_attribut(s)
si (type = 1) alors
      traiter_type_flou_1_BD(s, résultat1,a1,b1, nom_table, nom_attribut)
      traiter_type_flou_1_2_BMCF(résultat2,nom_table,nom_attribut,a1,b1)
sinon
      si (type = 2) alors
         traiter_type_flou_2_BD(s, résultat1,a2,b2, nom_table, nom_attribut)
          traiter_type_flou_1_2_BMCF(résultat2, nom_table, nom_attribut, a2, b2)
      sinon
         si (type=3) alors
          traiter_type_flou_3_BD(s, résultat1,number, nom_table,
          nom_attribut,domaine)
          traiter_type_flou_3_BMCF(résultat2,nom_table,nom_attribut,number,
          attribut_comp)
         sinon
          traiter_type_classique(s, résultat1)
         finsi
      finsi
finsi
fintq
Fin
```

Procédure 2. *La procédure traiter_ type_ flou_ 1_ BD permet de traduire la création d'un attribut FTYPE1 au niveau de la BD.*

```
Procédure traiter_type_flou_1_BD(s : cc, Var résultat : fichier, Var a, b : réel,
Var nom_table, nom_attribut : cc)
Début
Recherche_attribut(s, nom_attribut)
Recherche_valeur_FTYPE_1_2(s, a, b)
Supprimer_FTYPE1(s)
écrire_dans_fichier(résultat1,s)
Fin
```

Procédure 3. *La procédure traiter_type_flou_1_2_BMCF permet de stocker les informations concernant les attributs FTYPE1 et FTYPE2 au niveau de la BMCF.*

```
Procédure traiter_type_flou_1_2_BMCF(Var résultat2 : fichier, nom_table,
nom_attribut : cc, a1, b1 : réel, type_attribut :entier)
Début
Insérer_dans_FCL(résultat2,nom_table,nom_attribut,type_attribut)
Inserer_dans_FAM(résultat2,nom_table,nom_attribut,a1,a2)
Fin
```

Procédure 4. *La procédure traiter_type_flou_2_BD traduit la création d'un attribut FTYPE2 au niveau de la BD.*

```
Procédure traiter_type_flou_2_BD(s : cc, Var résultat : fichier, Var a, b : réel,
Var nom_table, nom_attribut : cc)
Var i, default : entier Restriction, attribut :cc
Début
Recherche_attribut(s, nom_attribut)
Copier(attribut, nom_attribut)
recherche_valeur_FTYPE_1_2(s, a, b)
Recherche_valeur_NUMBER(s, number)
default ← type_default(s)
Recherche_traduit_restriction(s, restriction)
Concaténer_mot(attribut, "T")
Supprimer_FTYPE2(s)
```

Concaténer_CHAINE(s, attribut)
Concaténer_chaine(s, "NUMBER(1) DEFAULT")
Concaténer_chaine(s, "default")
Concaténer_chaine(s, restriction)
concaténer_chaine(s, "CHECK (attribut BETWEEN 0 AND 7),")
Pour i de 1 à 4 faire
    concaténer_mot(nom_attribut,"i")
    concaténer_chaine(s, nom_attribut)
    concaténer_chaine(s, "NUMBER(number),")
finpour
écrire_dans_fichier(résultat1,s)
Fin

Procédure 5. *La procédure traiter_type_flou_3_BD traduit la création d'un attribut FTYPE3 au niveau de la BD.*

`Procédure` **traiter_type_flou_3_BD(s : cc, Var résultat : fichier,Var number : entier, Var nom_table, nom_attribut :cc, Var domaine : cc)**
Var i, default : entier
restriction, attribut :cc
Début
Recherche_attribut(s1, nom_attribut)
Copier(attribut, nom_attribut)
Recherche_valeur_FTYPE_3 (s, a, b)
Recherche_valeur_NUMBER(s, number)
$default \leftarrow type\_default(s)$
Recherche_valeur_restriction(s, restriction)
Recherche_valeur_DOMAIN(s, domaine)
Concaténer_mot(attribut, "T")
supprimer_FTYPE3(s1)
Concaténer_chaine(s1, "attribut")
Concaténer_chaine(s1, "NUMBER(1) DEFAULT")
Concaténer_chaine(s1, "default")
Concaténer_chaine(s1, restriction)
Concaténer_chaine(s1,"CHECK (attribut BETWEEN 0 AND 4),")

```
Pour i de 1 à number faire
    Concaténer_mot(attribut,"Pi")
    Concaténer_chaine(s1, nom_attribut)        Annuler_concat(attribut, i)
    Concaténer_chaine(s1, "NUMBER(3,2),")
    Concaténer_mot(attribut,"i")
    Concaténer_chaine(s1, nom_attribut)
    Concaténer_chaine(s1, "NUMBER(3),")
finpour
Ecrire_dans_fichier(résultat1,s1)
Fin
```

Procédure 6. *La procédure traiter_type_flou_3_BMCF permet de stocker les informations concernant les attributs FTYPE3 au niveau de la BMCF.*

```
Procédure traiter_type_flou_3_BMCF(Var résultat2 : fichier, nom_table,
nom_attribut : cc, number : entier, attribut_comp : cc)
Début
Insérer_dans_FCL(résultat2, nom_table, nom_attribut, type_attribut, number)
Inserer_dans_FCC(résultat2, nom_table, attribut_comp)
Fin
```

Procédure 7. *La procédure Traduction_CREATE_LABEL sauvegarde les paramètres d'une étiquette linguistique (LABEL) dans la BMCF.*

```
Procédure Traduction_CREATE_LABEL(s :cc, source :fichier,Var résultat2 :fi-
chier)
Début
si (s contient from) alors
    parametres_label2(s, nom_label2,nom_table2, attribut, a, b, c, d)
sinon
    parametres_label(s, nom_label, nom_table, attribut, a, b, c, d)
finsi
Inserer_dans_FOL(s, résultat2, nom_table, attribut, id_label, nom_label, type_label)
Inserer_dans_FLD(s, résultat2, id_label, nom_table, a,b,c,d)
Fin
```

Procédure 8. *La procédure Traduction_ CREATE_ NEARNESS sauvegarde les paramètres d'une relation de similitude (NEARNESS) dans la BMCF.*

```
Procédure Traduction_CREATE_NEARNESS(s :cc, source :fichier, Var résultat2 :fichier)
Début
parametres_Nearness(s, liste_label, nom_table, attribute, liste_degré,nombre_label)
pour i de 0 à nombre_label faire
        Inserer_dans_FOL(s,résultat2,nom_table,attribut,id_label,nom_label,type_label)
finpour
k=0
pour i de o à nombre_label2̂
      div 2 - nombre_label div 2 faire
      pour j de i à nombre_label - 1
         Inserer_dans_FND(s, résultat2, nom_table, attribut, id_label j, id :_label j+1,
          degré k)
          k ← k + 1
      finpour
finpour
Fin
```

5.2.4 Présentation de l'outil FSQL_TO_SQL

Nous avons programmé l'outil FSQL_TO_SQL avec le langage C++ sous Microsoft Visual C++ 6.0. Il supporte les systèmes d'exploitation Windows 2000/NT/XP.
Pour respecter les règles du génie logiciel et plus précisément le concept d'interface Homme/Machine, nous avons programmé FSQL_TO_SQL d'une façon à avoir une interface très simple et facile à utiliser (figure 5.3). Cette interface est similaire à celles utilisées dans les interfaces d'applications Microsoft.
Pour utiliser FSQL_TO_SQL, l'utilisateur doit, tout d'abord, saisir le script de sa BDF modélisé en FSQL en ouvrant une nouvelle page ou en donnant le chemin du fichier source contenant ce script (.fsql).
En cliquant sur le bouton traduire qui se trouve dans barre de menu sous Action, notre logiciel génère son équivalent en SQL (figure 5.4).

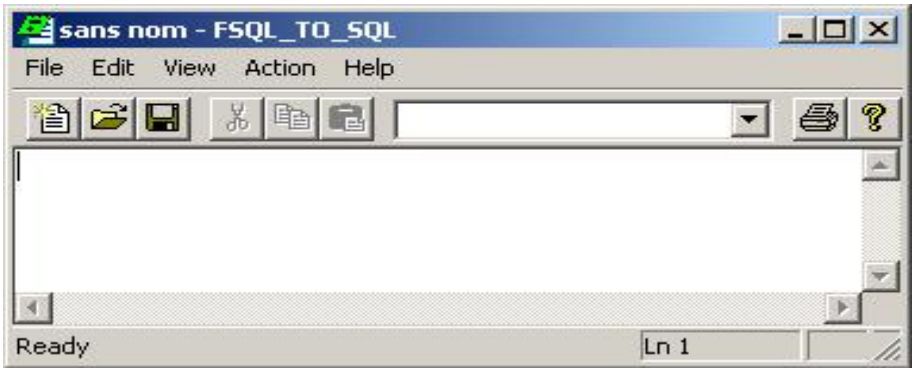

FIG. 5.3 – Interface de FSQL_TO_SQL

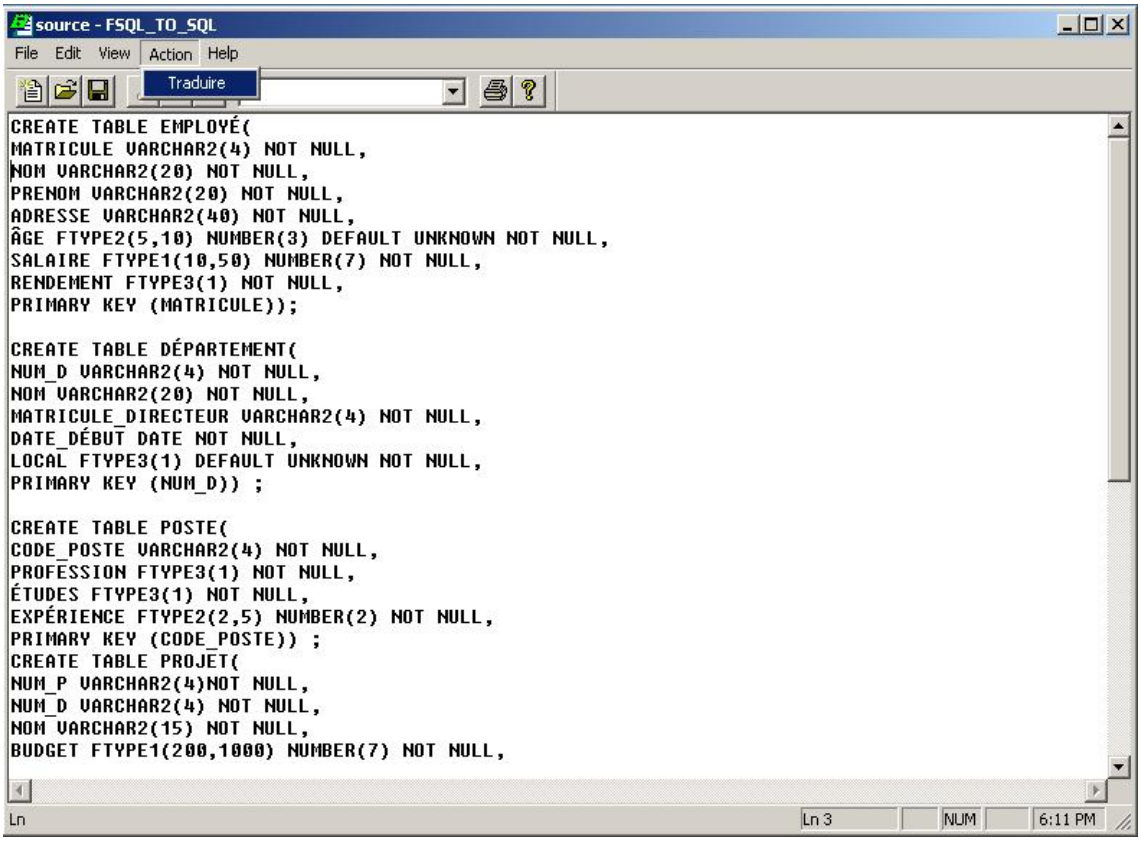

FIG. 5.4 – Édition d'un script FSQL sous FSQL_TO_SQL_DDL

Nous signalons à ce niveau les erreurs qui peuvent être détectées par notre outil. Si l'erreur est au niveau des nouveaux objets de FSQL, notre outil affiche cette erreur, la cause la plus éventuelle de son apparition et la ligne où elle se trouve (figure 5.5). Sinon (c'est à dire l'erreur s'est produite au niveau des commandes SQL), un rapport concernant cet erreur sera généré par le SGBD en question.

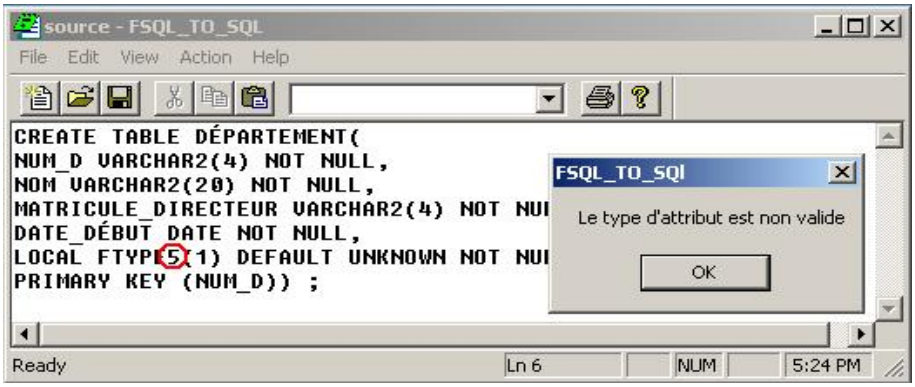

FIG. 5.5 – Traitement des erreurs dans FSQL_TO_SQL

Si la traduction a réussi (pas d'erreur), nous pouvons enregistrer le fichier résultat contenant le script de modélisation de la BDF décrit en SQL. Généralement, l'enregistrement s'effectue sous le répertoire $C : \backslash Oracle \backslash Ora81 \backslash BIN$ pour qu'il soit exécuté directement dans SQL*Plus.

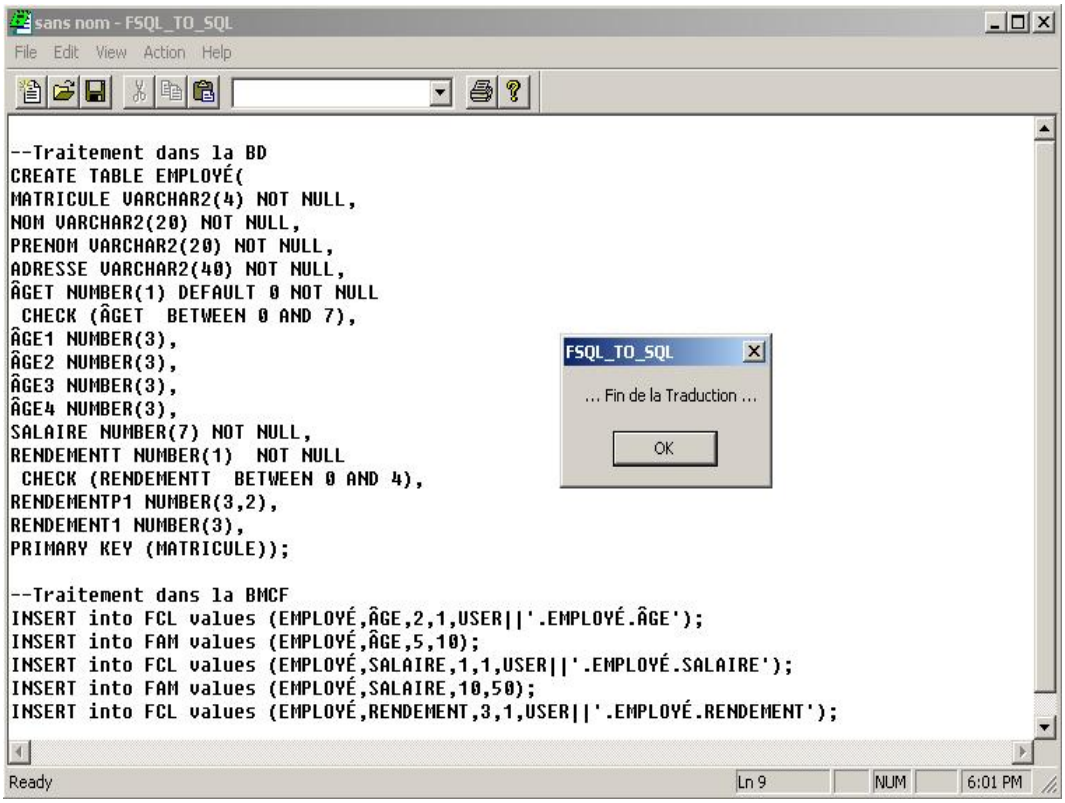

FIG. 5.6 – Traduction d'une requête FSQL sous FSQL_TO_SQL_DDL

5.3 Interrogation de la BDF

Maintenant que nous avons implémenté notre BDF, et pour bénéficier de son apport dans les requêtes flexibles, nous présentons quelques exemples de ces requêtes modélisées en langage FSQL en utilisant le logiciel FQ. Pour ceci, nous prenons comme exemple les requêtes qui peuvent être définis dans la table EMPLOYÉ de la figure 5.7.

Nº Fila	MATRICULE	NUM_D	CODE_POSTE	NOM	PRENOM	SEXE	ADRESSE	AGE	SALAIRE	ETUDES	RENDEMENT	EXPERIENCE	MATRICULE_SUP
1	E0	D1	P5	BEN SAID	AHMED	M	BARDO	17	600	LICENCE	REGULIER	TRES_PETITE	E10
2	E1	D1	P3	BEN HSSINE	MOHAMED	M	BARDO	JEUNE	800	LICENCE	BON	TRES_PETITE	E11
3	E2	D3	P1	GUESMI	RAMZI	M	LE KEF	27	800	LICENCE	BON	PETITE	E10
4	E3	D4	P3	BEN CHEIKH	IMEN	F	SOUSSE	[25,30]	700	LICENCE	BON	3	E10
5	E4	D0	P5	DAGHBAGI	BRAHIM	M	GABES	[40,45,48,55]	120	SECONDAIRE	MAUVAIS	20	E16
6	E5	D2	P5	BEN HSSINE	SARRA	F	ZAIGHOUAN	30±5	60	SECONDAIRE	BON	23	E1
7	E6	D0	P4	BEN SALEH	ZIED	M	EZZOUHOUR	23	600	LICENCE	REGULIER	BONNE	E1
8	E7	D0	P0	AZEIZ	ZIED	M	ARIANA	ADULTE	800	LICENCE	BON	5	E11
9	E8	D2	P5	HABBOUBI	SOFIENE	M	DENDEN	45	800	LICENCE	EXCELLENT	1	E1
10	E9	D1	P3	EL MEDDEB	ACHREF	M	BARDO	AGE	400	DIPLOME	BON	2±2	E11
11	E10	D3	P4	OUNELLI	HBIB	M	EL MOUROUGE	23	1200	DOCTORAT	EXCELLENT	BONNE	E16
12	E11	D1	P0	NAHDI	TAREK	M	EZZOUHOUR	35±5	800	LICENCE	REGULIER	BONNE	E11
13	E12	D3	P1	MGHIRBI	MAJDI	M	MANNOUBA	JEUNE	620	LICENCE	BON	20	E16
14	E13	D4	P0	MALLOULI	MOUNA	F	EL MOUROUGE	25	900	DOCTORAT	BON	15	E16
15	E14	D3	P4	EL ABDEN	ABDALLAH	M	LA MARSA	31	350	DIPLOME	MAUVAIS	3±2	E16
16	E15	D2	P1	GALINDO	JOSE	M	EL MANAR	JEUNE	750	LICENCE	BON	[2,6,13,17]	E11
17	E16	D1	P3	GRISSA	AMEL	F	EL MENZEH	27	1500	DOCTORAT	EXCELLENT	BONNE	E11
18	E17	D3	P1	NAHDI	HAMDI	M	EZZAHROUNI	48	800	LICENCE	REGULIER	7	E16
19	E18	D2	P2	SAADANI	MOUNIR	M	SLIMEN	VIEUX	800	LICENCE	BON	GRANDE	E16
20	E19	D2	P5	BEN SALEM	ILYESS	M	MANNOUBA	ADULTE	800	LICENCE	MAUVAIS	BONNE	E16
21	E20	D0	P2	JBEBLI	ILYESS	M	RADES	55	80	SECONDAIRE	BON	[15,20]	E10
22	E21	D3	P2	ALOUI	FATMA	F	EL MOUROUGE	[40,45,48,55]	800	DIPLOME	BON	SUFFISANTE	E10
23	E22	D1	P4	AMIN	SANA	F	FOUCHANA	4	800	LICENCE	REGULIER	15±2	E10

FIG. 5.7 – Extension de la relation EMPLOYÉ

Requête 1 :Considérons la requête qui consiste à trouver les employés **jeunes** (**avec un degré $\geq$ 0.5**)".
la formulation de cette réponse en FSQL est comme suit :
SELECT NOM,PRENOM, **CDEG(ÂGE)**
FROM EMPLOYÉ where ÂGE **FEQ $JEUNE THOLD 0.5**

Résultats retournés par la requête 1 : Nous remarquons que les tuples retournés correspondent à des employés qui appartiennent à l'étiquette jeune avec un degré >= 0.5 même s'ils appartiennent a d'autres étiquettes (l'employé AZEIZ figure dans la requête bien qu'il qu'il appartient à l'étiquette ADULTE, mais puisqu'il y a une intersection entre les deux fonctions trapézoïdales des étiquettes ADULTE et JEUNE, il figure dans ces tuples avec un degré de 0.8.
Requête 2 :Considérons la requête qui consiste à trouver les employés **jeunes** ayant

Nº Fila	NOM	PRENOM	CDEG(AGE)
1	BEN	MOHAMED	1
2	GUESMI	RAMZI	1
3	BEN	IMEN	1
4	BEN	SARRA	1
5	BEN SALEH	ZIED	1
6	AZEIZ	ZIED	0,8
7	OUNELLI	HBIB	1
8	NAHDI	TAREK	0,5
9	MGHIRBI	MAJDI	1
10	MALLOULI	MOUNA	1
11	EL ABDEN	ABDALLAH	0,8
12	GALINDO	JOSE	1
13	GRISSA	AMEL	1
14	BEN SALEM	ILYESS	0,83

FIG. 5.8 – Les tuples retournés par la requête 1

un salaire qui excède 800 dinars.
la modélisation de cette requête est comme suit :
SELECT NOM, PRENOM, **CDEG(ÂGE)**, SALAIRE FROM EMPLOYÉ
WHERE ÂGE **FEQ $JEUNE** AND SALAIRE >= 800
Cette requête montre que nous pouvons utiliser des comparateurs flous et classiques (exactes) ensembles.

Nº Fila	NOM	PRENOM	CDEG(AGE)	SALAIRE
1	BEN	MOHAMED	1	800
2	GUESMI	RAMZI	1	800
3	OUNELLI	HBIB	1	1200
4	MALLOULI	MOUNA	1	900
5	GRISSA	AMEL	1	1500

FIG. 5.9 – Les tuples retournés par la requête 2

Si nous voulons demander un salaire **approximativement plus grand que** 750 dinars. La requête sera :
SELECT NOM, PRENOM, **CDEG(ÂGE)**, SALAIRE FROM EMPLOYÉ
WHERE ÂGE **FEQ $JEUNE** AND SALAIRE **FGT #750**
La phrase **approximativement** est transformée en # qui est une constante floue indiquant `une valeur approximative`, alors que la phrase plus grand que s'est transformée en **FGT** qui est un `comparateur flou` ayant le sens de possiblement plus grand que. Cette requête renvoie les mêmes tuples affichés précédemment.
Cependant lorsque nous remplaçons le comparateur FGT (Fuzzy Greater Then) par le comparateur MGT (Mutch Greater Then), nous remarquons que certains tuples

ne s'affichent pas. Ceci est dû à la valeur indiquée lors de la création de l'attribut SALAIRE qui indique une marge et une valeur limite pour considérer deux valeurs différentes. En fait la marge ici est de 10 et la valeur limite est 50 et par suite les employés ayant un salaire dans l'intervalle [750, 800] ne s'affichent pas.
La requête sera modélisée comme suit :
SELECT NOM, PRENOM, **CDEG(ÂGE)**, SALAIRE FROM EMPLOYÉ
WHERE ÂGE **FEQ $JEUNE** AND SALAIRE **MGT** #**750**

les tuples retournés par cette requêtes sont :

Nº Fila	NOM	PRENOM	CDEG(AGE)	SALAIRE
1	OUNELLI	HBIB	1	1200
2	MALLOULI	MOUNA	1	900
3	GRISSA	AMEL	1	1500

FIG. 5.10 – Les tuples retournés par la requête 3

5.4 Limites de la modélisation d'une BDF

Avec l'outil FSQL_TO_SQL, nous pouvons maintenant exploiter une BDF dès sa création jusqu'à son interrogation, mais il nous reste encore un problème. A la description d'une BDRF, plusieurs questions peuvent se poser concernant le choix du type de l'attribut :

- "Sur quels critères choisir le type de l'attribut ?"
- "Pourquoi, un attribut peut être dans des cas FTYPE1 et dans d'autres cas FTYPE2."
- Dans la requête "donner les employés jeunes qui travaillent dans un département à gros budget avec un degré minimum égal à 0.5", ce degré est-il de possibilité, d'incertitude ou d'appartenance et comment choisir son type ?

Ces problèmes sont illustrés dans les points suivants :

Importance de l'attribut dans le Système d'Information

Tout attribut doit être qualifié dans le SI ou dans la BD ou il sera hébergé. Cette qualification discute son importance dans la BD. Si nous prenons à titre d'exemple l'attribut âge dans une BD d'annonces publicitaires, l'âge du client n'est pas une donnée assez importante dans cette BD, il peut alors avoir des valeurs floues ou

approximatives et par suite il sera flou de type 2 (FTYPE2). Si nous changeons maintenant le domaine de la BD de cet attribut vers une BD de la caisse de sécurité sociale qui s'occupe des employés assurés. Un employé qui atteint les 60 ans reçoit automatiquement un mandat. Nous remarquons bien que l'âge d'un employé ne peut pas être approximatif ou imprécis, vu son importance dans le système d'information associé, et par suite, sa précision est très demandée.

Possibilité de stockage de données imprécises

Un autre critère pour choisir le type d'attribut est sa possibilité de stocker des données floues, un attribut tel que "rendement" ne peut pas avoir des valeurs tels que 10 ou 50 (exactes) mais par contre, il peut avoir des valeurs sous forme d'étiquettes linguistiques tel que régulier ou mauvais et par suite il doit être de type 3 (FTYPE3). La matricule d'un employé ne doit avoir ni des données floues ni des constantes floues dans la requête et donc elle doit être de type classique.

Interrogation de l'attribut

Si nous parlons maintenant de l'interrogation de la BD, si un attribut va subir une interrogation flexible, il devra être l'un des trois attributs FTYPE1, FTYPE2 ou FTYPE3. Néanmoins, d'autres critères d'interrogation participent à la décision du choix d'attribut. On peut citer les comparateurs flous (le comparateur FGT ne peut pas être appliqué sur un attribut FTYPE3), les constantes floues (la constante "Approximative" n'est pas applicable pour un attribut FTYPE3),...

Degrés associés à l'attribut

Dans [GAL 04] il a été introduit les degrés sous forme d'attribut (quatre types) chacun ayant un sens. Ce qui nous intéresse ici est l'importance de ces degrés dans la description du monde réel. En fait, il faut préciser une stratégie pour choisir le sens des degrés dans notre BDF. Dans la requête "trouver les employés ayant un rendement régulier avec un degré minimum 0.5", le degré peut avoir le sens de possibilité que l'employé X a un rendement régulier que de réalisation ou d'appartenance. Par contre dans la requête "trouver les employés jeunes avec un degré minimum 0.7", le degré peut avoir le sens d'appartenance de l'employé à jeune ou de possibilité d'être jeune que d'importance.
Tous ces problèmes sont flous en terme de complexité. Ils seront posés comme perspectives dans des travaux ultérieurs.

5.5 Conclusion

Dans ce chapitre, les difficultés supplémentaires apportées par le processus de création d'une BDF supportant le modèle GEFRED ont été mises en évidence. Pour se faire, et tout en respectant le modèle GEFRED, nous avons étendu l'architecture FIRST afin d'autoriser l'utilisateur d'interagir avec une BDF. Ensuite, et par analogie à FQ, nous avons introduit un outil, qui en se basant sur des règles bien définies, permet de traduire un script FSQL à son équivalent SQL2. Ces règles sont basées sur l'architecture FIRST étendue et les relations dans la BMCF. L'apport essentiel de cet outil est de faciliter la tâche du concepteur de la BDF et d'automatiser le processus de son implémentation

Conclusion et Perspectives

Habituellement, pour stocker de grandes quantités de données, les bases de données ont des mécanismes incorporés seulement pour des relations exactes. Comme la technologie est appliquée pour atteindre des domaines très variés, qui supportent des données vagues, imprécises et/ou des préférences dans les attributs, parfois les relations usuelles de SQL ne peuvent pas complètement satisfaire ou modéliser les relations et les attributs de ces domaines. L'interrogation flexible constitue alors une alternative à l'interrogation booléenne pour certains types d'applications.

Nous avons tenté, dans ce mémoire, d'étudier ce type d'interrogation dans les BD floues. Pour ce faire, nous avons commencé par faire un tour d'horizon des publications faites sur le sujet. Ce tour d'horizon a permis de constater que la flexibilité dans l'interrogation des BD est utilisée dans les systèmes relationnels. Le développement de la théorie des sous-ensembles flous a constitué un cadre général pour supporter cette flexibilité, notamment, dans les SGBD relationnels. Nous avons alors présenté les deux BDF qui existent. Puis nous nous sommes concentrés sur modèle GEFRED pour introduire notre travail. Dans une deuxième étape de ce mémoire, nous avons proposé une approche pour automatiser le processus de création d'une BDF. Cette approche étend l'architecture actuelle (FIRST) proposé par Medina en ajoutant une couche qui permet à l'utilisateur la description et la manipulation de sa BDF. Pour ceci, nous avons implémenté un outil, appelé FSQL_TO_SQL.
Cet outil permet de transformer un script décrit en langage FSQL à son équivalent décrit en SQL.

Ainsi, notre approche permet de rendre transparent la création d'une BDF avec le langage FSQL. Le concepteur n'est plus obligé de dominer la base de méta connaissances du modèle GEFERD ni son processus d'implémentation sous le SGBD Oracle. Les BDF peuvent être utilisés dans plusieurs domaines à savoir le domaine de traite-

ment d'image et de reconnaissances de paroles. En effet lorsque nous cherchons une image dans une base de données, il est intéressant de la poser en fixant un certain degré d'incertitude. Par exemple chercher l'image qui est à 80% identique à l'image en question. D'après nous, Ceci sera une grande contribution dans l'évolution de ce domaine vu que, généralement, nous ne pouvons pas avoir une image exacte (nette sans bruit). Ceci est de même dans le domaine de reconnaissance de parole, il serait intéressant de chercher les voix qui sont proches à 90% d'une voie fixée. Plusieurs perspectives futures s'imposent. Nous en citons les suivantes :

- Utilisation des BDF dans la recherche d'images et la reconnaissance de voie et de forme.
- Concevoir un système expert qui permet d'aider un utilisateur à choisir les types d'attributs flous de sa BDF (FTYPE1, FTYPE2, FTYPE3).
- Introduire le concept flou dans les bases de données avancées à savoir les BD déductives, BD Objet Relationnelles et les BD Orientées Objet

Annexe

Algorithmes de Traduction FSQL_TO_SQL

Comme continuation aux algorithmes présentés dans la section 5.1.2, nous exposons dans ce qui suit les autres sous programmes nécessaires aux fonctionnement de l'outil FSQL_TO_SQL.

Procédure 9. *La procédure Traduction_ALTER_TABLE traduit la modification d'une table*

Procédure Traduction_ALTER_TABLE(s :cc, Var source :fichier, Var résultat1 : fichier, Var résultat2)
Début tanque(s n'a pas atteint point virgule) faire
si (type_alter = "ADD") alors
    $type \leftarrow test_attribut(s)$
si(type=1) alors
    traiter_type_flou_1_BD(s, résultat1,a1,b1, nom_table)
    traiter_type_flou_1_2_BMCF(s, résultat2,nom_table,a1,b1)
sinon
    si (type = 2) alors
      traiter_type_flou_2_BD(s, résultat1,a2,b2, nom_table)
      traiter_type_flou_1_2_BMCF(s, résultat2,nom_table,a2,b2)
    sinon
      si (type=3) alors
        traiter_type_flou_3(s, résultat1,number, nom_table,domaine)
        traiter_type_flou_3_BMCF(s, résultat2, nom_table,number,domaine)
      sinon
        traiter_type_classique(s, résultat1)
      finsi

```
        finsi
finsi
sinon
            modifier_dans_FMB(s,résultat2)
finsi
fintq
Fin
```

Procédure 10. *La procédure Traduction_ ALTER_ LABEL traduit la modification d'une étiquette linguistique.*

```
Traduction_ALTER_LABEL(s :cc, source :fichier,Var résultat2 :fichier)
Début
parametres_nouveau_label(s, nom_label1, id_label, nom_table attribut, a2, b2, c2, d2)
parametres_nouveau_label(s, nom_label2, nom_table, attribut, a, b, c, d)
Inserer_dans_FOL(s, résultat2, nom_label2, id_label, nom_table, attribut)
Inserer_dans_FLD(s, résultat2, id_label, nom_table, a2,b2,c2,d2)
Fin
```

Procédure 11. *La procédure Traduction_ ALTER_ NEARNESS traduit la modification d'une relation de similitude.*

```
Traduction_ALTER_NEARNESS(s :cc, source :fichier, Var résultat2 :fichier)
Début
parametres_nouveau_Nearness(s, num_label, id_label, nom_table, attribut)
si ( s ne contient pas CDEG) alors
        remplacer_ALTER_CREATE(s)
        Traduction_CREATE_NEARNESS(s :cc, source :fichier, Var résultat2 :fichier)
sinon
pour i de 0 à num_label-2 faire
        nouveau_degré(s, nouveau_degré,id_label1,id_label2)
update_dans_FND(s,résultat2,nom_table,attribut,id_label1,id_label_2,nouveau_degré)
finpour
finsi
Fin
```

Procédure 12. *La procédure Traduction_DROP_TABLE traduit la suppression d'une table.*

Procédure Traduction_DROP_TABLE(s :cc, Var source :fichier, Var résultat1 : fichier)
Début
supprimer_table(s, résultat1) si (s contient FTYPE1 ou FTYPE2) alors
parametres_FTYPE_1_2(a,b, domaine, attribut, nom_table)
supprimer_de_FCL(s, résultat2, nom_table)
supprimer_de_FAM(s, résultat2, a,b, nom_table)
sinon
si(s contient FTYPE3) alors
parametres_FTYPE3(number, domaine, attribut, nom_table)
supprimer_de_FCL(s, résultat2, nom_table)
supprimer_de_FCC(s, résultat2, nom_table)
finsi
finsi
Fin

Procédure 13. *La procédure Traduction_DROP_LABEL traduit la suppression d'une étiquette linguistique.*

Procédure TRADUCTION_DROP_LABEL (s :cc, Var résultat2 : fichier)
Début
parametres_label(s, nom_label, id_label, nom_table, attribut, a, b, c, d)
supprimer_de_FOL(s, résultat2, id_label, nom_table, attribut)
supprimer_de _FLD(s, résultat2, id_label, nom_table, attribut)

Procédure 14. *La procédure Traduction_DROP_NEARNESS traduit la suppression d'une relation de similitude.*

Procédure DROP_NEARNESS(s :cc, Var résultat2 :fichier)

Début

parametres_Nearness(s, num_label, nom_table, attribut)

si (s contient l'étiquette) alors

supprimer_de_FOL(s,résultat2, nom_table, attribut, id_label_1)

pour i de 0 à num_label-2 faire

supprimer_de_FND(s, résultat2,nom_table, attribut, id_label, id_label_i, degré)

finpour

sinon

pour i de 0 à num_label-2 faire

supprimer_de_FOL(s, résultat2, nom_table, attribut, id_label_i)

supprimer_de_FND(s, résultat2, nom_table, attribut, id_label_i)

finpour

finsi

Fin

Bibliographie

[BEZ 93] J.C. Bezdek "Fuzzy Models -What Are They, and Why ?". *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 1, February 1993 - Edited by P.D.

[BOS 92] P. Bosc , O. Pivert, "Some Approches for relational Database Flexible Querying". *International Journal of Intelligent Information Systems*,1,pp. 323-354, 1992.

[BOS 93] P. Bosc, O. Pivert, "On the Evaluation of Simple Fuzzy Relational Queries : Principles and Measures". *Fuzzy Logic : State of the Art, Kluwer Acadimic Publishers*, pp. 355-36, 1993.

[BOS 95] P. Bosc, O. Pivert, "SQLf : A relational database language for fuzzy quering", *IEEE transactions on Fuzzy Systems*, 3, pp. 1-17, 1995.

[BOS 97] P. Bosc, D. Dubois, O. Pivert, H. Prade, "Flexible queries in relational databases : The example of the division operator". *Theoretical Computer Science*, 171, pp. 281-302, 1997.

[BOS 98] P. Bosc, L. Liétard, O. Pivert, "Bases de données et Flexibilité : Les requêtes Graduelles". *Techniques et Sciences informatiques*, 17(3), pp. 355-378, 1998.

[BOS 00] P. Bosc, O. Pivert, "SQLf query functionality on top of a regular relational database management system", *Knowledge Management in Fuzzy Databases Heidelberg*, 2000.

[BOS 02] P. Bosc, L. Cholvy, D. Dubois, N. Mouaddib, O. Pivert,H. Prade, G. Raschia, M. C. Rousset, "Les informations incomplètes dans les bases de données et en intelligence artificielle". *Actes des deuxièmes assises nationalesdu GdR I3*, 2002.

[BOS 03] P. Bosc, L. Liétard, O. Pivert, D. Rocacher, "Bases de données multimédia et interrogation souple". *Actes des deuxièmes assises nationales du GdR I3*, 2003.

[BUC 82] B.P. Buckles , F.E. Petry, "A Fuzzy Representation of Data for Relational

Databases". *Fuzzy Sets and Systems*, 7, pp. 213-226, 1982.

[CAS 96] E. C. Casanova, J. M. D. Victoria, "Bases de Datos Borrosas". Cours de Bases de Données Floues, 1996.

[CHA 82] C.L. Chan "Decision Support in an Imperfect World". Research Report, pp. 100-102, 1982.

[COD 70] E.F. Codd, "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM*, 13, N^0 6, pp. 377-387, 1970.

[GAL 98a] J. Galindo, J.M. Medina, O. Pons, J. C. Cubero, "A Server for Fuzzy SQL Queries". *Flexible Query Answering Systems*,eds. T. Andreasen, H. Christiansen and H.L. Larsen, *Lecture Notes in Artifcial Intelligence (LNAI)* 1495, pp.164-174, 1998. Ed. Springer, 1998. International Conference on Flexible Query Answering Systems, FQAS'98, Roskilde (Denmark), May 1998.

[GAL 98b] J. Galindo, J.M. Medina, A. Vila, J. C CUBERO, "Fuzzy Comparators for Flexible Queries to Databases", *Proc. of IBERAMIA'98*. Lisbona (Portugal), 1998.

[GAL 98c] J. Galindo, J.M. Medina, O. Pons, M.A. Vila, J.C. Cubero, "A Prototype for a Fuzzy Relational Database". *Demo Session in the 6th International Conference on Extending Database Technology, EDBT'98*, Valencia (Spain), March 1998.

[GAL 99a] J. Galindo, "Tratamiento de la Imprecisión en Bases de Datos Relacionales : Extensión y Adaptación de los SGBD Actuales". Tesis Doctoral. Universidad de Granada, 1999.

[GAL 00] J. Galindo, J. M. Medina, J. C. Cubero, M. T. Garcia, "Fuzzy Quantifiers in Fuzzy Domain Calculus" *8-th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU'2000*, Madrid, juillet 2000.

[GAL 02] J. Galindo, M.C. Aranda, J. L. Caro, A. Guevara, A. Aguayo "Applying Fuzzy Databases and FSQL to the management of rural accommodation". *Tourist Management Journal*, pp. 623-629, 2002.

[GAL 04] J. Galindo, A. Urrutia, M. Piattini, "Representation of fuzzy knowledge in relational databases". *Database and Expert Systems Applications, Proceedings. 15th International Workshop.* pp. 917-921, 2004.

[GRI 00] R. Grin, "Introduction aux bases de données, modèle relationnel". Version 3.3, 2002.

[GRI 05] R. Grin, "Langage SQL et objet-relationnel". Version 5.1, 2005.

[HAC 02] N. Hachani, "Conception et implémentation d'une approche de relaxation d'une requête SQL". Mémoire de mstère, *Institut Supérieur de Gestion de Tunis*, 2001.

[HEL 01] M. Hellmann, "Fuzzy Logic Introduction". 2001.

[ICH 86] T. Ichikawa, M. Hirakawa, "ARES : A Relational Database with the Capability performing Flexible Interpretation of Queries". *IEEE transactions on Software Engineering*, pp. 624-634, 1986.

[KAC 94] J. Kacprzyk, S. Zadrozny, "Fuzzy Quering for Microsoft Access". *Proceeding of Third IEEE International conference on Fuzzy Systems*, 1, pp. 167-171 (1994).

[KAC 86] J. Kacprzyk, A. Ziolkowski,"database Queries with Fuzzy Linguistic Quantifiers". *IEE Transaction systems*, 16(3), pp. 474-478, 1986.

[LAC 87] M. Lacroix, P. Lavency "preferences : Putting More Knowledge into Queries" 13th VLDR Conference, 217-225, 1987.

[LAM 00] Lambalgen "Fuzzy logic". ILLC University of Amesterdam, 2000.

[MED 94a] J.M. Medina, O. Pons, M.A. Vila, "GEFRED : A Generalized Model for Fuzzy Relational Databases". *Information Sciences*, 77 (6), pp. 87-109, 1994.

[MED 94b] J.M. Medina, O. Pons, M. A. Vila, "An elemental processor of fuzzy SQL". *Mathware and Soft Computing*, 1 (3), pp. 285-295, 1994.

[MED 95] J.M. Medina, O. Pons, M.A. Vila, "FIRST. A Fuzzy Interface for Relational SysTems". *VI International Fuzzy Systems Association World Congress (IFSA 1995)*. Sao Paulo (Brasil), 1995.

[MOT 82] A. MOTRO "VAGUE : A User Interface to Relational Databases that Permits Vague Queries". *ACM Transaction off Information Systems*, 6(3), pp. 187-214, 1988.

[PON 95] O. Pons, "Representacion Logica de Bases de Datos Difusas Fundamentos Teoricos e Implementacion" thèse de doctorat 1995.

[PRA 82] H. Prade, C. Testemale, "Fuzzy Relational Databases : Representational issues and Reduction Using Similarity Measures". I*nformation Sciences*, 38(2), pp. 118-126, 1987.

[RAB 90] F. Rabatti, "Retrieval of Multimedia Documents by Imprecise Query Specification" *Lecture Notes in Computer Science*, 416, pp. 202-218, 1990.

[RIG 03] P. Rigaux, "Cours de bases de données ". 2003

[TAH 77] V. Tahani, "A conceptual Framework for Fuzzy Query Processing : A step Toward Very Intelligent Database Systems". *Information Processing and Management*, 13, pp. 289-303, 1977.

[UMA 80] M. Umano, S. Fukami, M. Mizumoto, K. Tanaka, "Retrieval Processing from Fuzzy Databases". *Technical Reports of IECE of Japan*, Vol. 80, N^0 204 (on Automata and Languages), pp. 45-54, AL80-50, 1980.

[YOS 93] T. Yoshikane, "Fuzzy Database Query Languages and Their Relational Completeness Theorem". *IEEE Transaction On Knowledge and Data Engineering*, vol 5, Nř 1, February 1993.

[ZAD 65] L.A. Zadeh, "Fuzzy Sets". *Information and Control*, Vol. 8, pp. 338-352, 1965.

[ZEM 85] M. Zemankova-Leech , A. Kandel , "Implementing Imprecision in Information Systems". *Information Sciences*, 37, pp. 107-141, 1985.

Netographie

[BEN 02] Benjamin F., Xavier N. et Florent S. "T.E.R : LA LOGIQUE FLOUE", 2001, (http ://www.cindy.ensmp.fr/%7Ecoueque/TER/logiqueFloue/ter.html)

[CAM 01] C. Campioni, "Cours de bases de données relationnelles". (www.cmi.univ-mrs.fr/ campioni/documents/BD/BD-relationnelles.pdf).

[GAB 01] GABRIEL P., "Introduction générale à la logique floue et à la théorie des sous-ensembles floues". 2001,(http ://elap.montefiore.ulg.ac.be/cours/documents/fuzzy/Figure1/la_logique_floue.htm)

[GAL 99b] J. Galindo, "le serveur FSQL + FQ", 1999. (http ://www.lcc.uma.es/personal/ppgg/FSQL.html)

[GAM 03] A. Gamache, "Modèles, Architectures et Langages de données" (http ://www.ift.ulaval.ca/%7Eagamache/developpez/transitCompteurDev.html).

[MAR 01] H. MARTIN, "Base de Données et Système de Gestion de Base de Données ", (http ://www-lsr.imag.fr/Les.Personnes/Herve.Martin/HTML/FenetrePrincipale.htm).

[STE 03] S. Stefano, "Base de données". 2003, (http ://lbdwww.epfl.ch/f/teaching/courses/poly1/).

[DAE 01] Daehne, P. Leutert, "Modèles relationnel et objet", Novembre 2001. (www.hesge.ch/heg/prestations_recherche/doc/pd_isnet15.pdf)

RÉSUMÉ

Les Systèmes de Gestion de Bases de Données Relationnelles (SGBDR) sont devenus, sans conteste, le noyau de tout système informatique. Cependant, la diversification des applications des bases de données a montré les limites des SGBDR notamment sur le plan de modélisation des données imprécises et de l'interrogation flexible. Ainsi, plusieurs extensions du modèle relationnel et du langage SQL ont été proposées pour introduire une certaine incertitude dans la modélisation des données et une certaine flexibilité dans l'interrogation des BD. Les bases de données floues offrent ces extensions et représentent un cadre adéquat pour les manipuler.
Le présent mastère est une contribution à l'étude des BDF. Dans ce contexte, nous avons étendu l'architecture FIRST qui se base sur le modèle GEFRED, avec une couche FSQL_TO_SQL permettant la transformation et la traduction d'un script de modélisation d'une BDF écrit en FSQL à son équivalent SQL. Avec cette extension, un utilisateur peut modéliser sa BDF sans avoir le souci de manipuler la complexité de sa création.

Mots clés : Requête flexible, Modèle GEFRED, Algèbre Relationnelle Etendue, Algèbre Relationnelle Floue Généralisée, SQL, FSQL, SQLf, FIRST.

ABSTRACT

The Database Management Systems (DBMS) became, without dispute, the core of all computer system. However, the diversification of the applications of the data bases showed the limits of the DBMS notably as regards to modelling of the imprecise data and the flexible querying. Thus, several extensions of the relational model and the SQL language have been proposed to introduce both, a quite uncertainty in the modelling of the data and a flexibility in DB queries. The fuzzy databases (FDB) offer these extensions and represent an adequate setting to manipulate them.
The present master is a contribution to the survey of the FBD. In this context, we extend the FIRST architecture that is based on the GEFRED model, with a FSQL_TO_SQL layer allowing the transformation and the translation of a modelling script of a written FBD in FSQL to the equivalent SQL. With this extension, the user can describe his FBD without worrying about the manipulation of the creation's complexity.

Key words : Flexible queries, GEFRED Model, Extended Relational Algebra, Generalized Fuzzy Relational Algebra, SQL, FSQL, SQLf, FIRST.