Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Centre Universitaire Abbas LAGHROUR Khenchela

École Doctorale (E.S.I)

Sciences et Technologies de l'Information et de la Communication (STIC)

Mémoire

Pour l'obtention du diplôme de magistère en informatique

Option : Systèmes d'**I**nformations et de **C**onnaissances (SIC)

Intégration de la logique floue dans la gestion des bases de données Application à PostgreSQL

Présenté par **Kamel SEMASSEL**

Soutenu publiquement le 21 avril 2012 devant le jury composé de :

Mr. BALLA Amar	Professeur à l'ESI	Président
Mr. REDJIMI Mohamed	Maître de Conférences à l'Université de Skikda	Examinateur
Mr. MAZOUZI Smaine	Maître de Conférences à l'Université de Skikda	Examinateur
Mr. HIDOUCI Walid-Khaled	Maître de Conférences à l'ESI	Directeur de mémoire
Mr. LOUDINI Malik	Maître de Conférences à l'ESI	Invité

Remerciements

Je tiens à remercier Messieurs ***HIDOUCI Walid-Khaled*** *et* ***LOUDINI Malik,*** *Maitres de conférences à l'Ecole Nationale Supérieure en Informatique (Ex. INI), pour le sujet proposé, leurs directives précieuses et leur disponibilité tout au long de ce travail.*

Je tiens aussi à exprimer ma gratitude au ***Prof. BALLA Amar*** *de l'Ecole Nationale Supérieure en Informatique (ESI), pour m'avoir fait l'honneur en acceptant de présider le jury.*

Que Messieurs ***REDJIMI Mohamed*** *et* ***MAZOUZI Smaine****, Maîtres de conférences à l'Université de Skikda, trouvent içi l'expression de ma reconnaissance pour avoir accepté d'examiner et juger mon travail.*

Merci à ceux qui ont participé de près ou de loin à l'élaboration de ce mémoire.

Résumé

Les systèmes de gestion de base de données les plus répandus actuellement sont les systèmes relationnels. Cependant, ils sont incapables de représenter ou de manipuler des données imprécises ou incertaines. Différentes approches d'intégration des données imparfaites (nulles, imprécises, incertaines…) ont conclu sur l'utilisation de la logique floue qui constitue un cadre très adapté et unifié pour la représentation de ces types de données. Le modèle GEFRED constitue le modèle le plus général des bases de données floue, mais par rapport à la complexité et à la variété des informations du monde réel, sa représentation reste rigide et limitée.

Dans ce mémoire, Nous proposons une extension du modèle GEFRED , afin de lui permettre une représentation multidimensionnelle des concepts floues avec un module d'adaptabilité floue aux différentes situations de l'utilisateur, ainsi le système peut attribuer aux concepts flous (comme les étiquettes linguistiques par exemple) plusieurs significations floues répondants aux différents situations trouvés dans le monde réel , ce qui permet d'obtenir un modèle multidimensionnel de représentation des données imprécises et de préférences des utilisateurs. AFSQL (Adaptive FSQL) est notre extension pour traiter les requêtes floues adaptatives. Nous proposons également une nouvelle méthode d'implémentation du modèle sous le SGBD open source PostgreSQL.

Mots clés: bases de données floues, requêtes flexibles, logique floue, ensembles flous, données imprécises, GEFRED, FSQL.

Abstract

The most widespread DBMS currently are the relational systems, however, they are unable to represent imprecise or uncertain data , several research tasks were established to deal with this problem, with the appearance of the fuzzy logic which constitutes a very adapted framework to the representation of vague knowledge, the GEFRED model constitutes the most general model of fuzzy relational database, but compared to the complexity and the variety of information of the real world, its representation remains rigid and limited.

In this work, We propose an extension of GEFRED model, in order to allow a multidimensional representation of fuzzy data with a module of fuzzy adaptability to the various situations of the user, thus the system can assign to the fuzzy concepts (like the linguistic labels for example), several fuzzy significances related to different situations and contexts met in the real world, which makes possible to obtain a multidimensional model of fuzzy data representation and user preferences. AFSQL (Adaptive FSQL) is our extension to treat the adaptive fuzzy query. We also propose a new method of implementation of the model under an open source DBMS which is PostgreSQL.

Key words: fuzzy databases, flexible query, fuzzy logic, fuzzy set, imprecise data, GEFRED, FSQL.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les bases de données sont actuellement au cœur de tout système d'information dans n'importe quel domaine, on les utilise pour représenter les informations que l'on peut avoir sur le monde réel ou tout au moins, sur une partie du monde réel. Cette représentation doit être la plus fidèle possible afin que l'interrogation de ce système permette à son utilisateur de se faire l'idée la plus exacte possible et de prendre des décisions adéquates. Cependant, notre connaissance du monde réel est souvent imparfaite et imprécise, ceci fait défier notre capacité de créer des bases de données efficaces et performantes.

Il y a deux solutions envisageables dans les situations où la connaissance du monde réel est imparfaite, La première solution est de limiter le modèle à cette partie du monde réel dont l'information parfaite est disponible. Dans le modèle de données relationnel, ceci implique qu'un tuple qui décrit un détail de l'employé par exemple sera exclu tout à fait de la base de données si un de ses attributs (âge ou salaire ou autre) est manquant ou est suspect d'imprécision. La base de données résultante modélisera seulement les employés pour lesquels les informations parfaites sont disponibles.

La deuxième solution est de développer des modèles de données qui permettent la représentation des informations imparfaites. Supposant que les informations disponibles sur l'âge d'un employé particulier sont imparfaites ; par exemple, la seule information connue sur l'âge d'une personne est son intervalle, si le modèle de données est doté des outils de spécification et de manipulation des intervalles, alors l'information imparfaite pourrait être stockée dans une base de données gérant ce type d'information et pourra être ensuite exploitée.

Puisque la deuxième solution nécessite souvent des outils supplémentaires, la plupart des systèmes de gestion de base de données adhèrent aux modèles de données qui n'incluent que quelques outils standards pour gérer l'information imparfaite, la possibilité de stocker des valeurs *nulles* étant l'outil le plus généralement disponible.

Les bases de données relationnelles sont reconnues comme un modèle de base de données très utile dans les systèmes d'information et ont trouvé des applications larges avec succès. Les données traitables dans ce modèle doivent être de valeurs précises. Une extension de ce modèle a permis de

traiter les données *NULLES*. Ainsi, la valeur NULLE est utilisée pour tous les types imprécis tels que *inconnu* et *inapplicable*.

En ce qui concerne l'interrogation classique d'une base de données relationnelle, 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 base de données, Plusieurs travaux ont été menés pour pallier ce problème. Le deuxième problème est que l'interrogation booléenne ne permet pas à l'utilisateur ni d'utiliser des termes linguistiques vagues ou 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. D'autre part, l'utilisateur souhaite que ses préférences soient considérées selon l'ordre décroissant, ainsi, les données retournées par le SGBD doivent être ordonnées et présentées à l'utilisateur selon ses préférences. Sans cette flexibilité, l'utilisateur doit affiner les critères de recherche jusqu'à obtenir éventuellement des résultats satisfaisants car il n'a pas des connaissances à priori sur les données qu'il consulte.

Les modèles classiques de base de données se trouvent donc incapables de représenter et de manipuler de l'information imprécise. Depuis le début des années 80, La logique floue introduit par Zadeh dans les années 1965 [ZAD 65] a été employée pour étendre divers modèles de base de données afin de remédier au problème de représentation et de traitement des données imprécises et ceci selon différentes approches.

Le modèle GEFRED constitue le modèle le plus général des bases de données relationnelles floues, il permet la représentation des concepts flous de différents types. Malgré les extensions qui ont été réalisées sur ce modèle [GAL 06][GTB 09], sa représentation s'avère rigide et ne peut pas satisfaire le besoin d'une représentation élargie et flexible comme dans le monde réel.

L'objectif de ce mémoire est de proposer une extension du modèle GEFRED sur les deux niveaux : niveau de représentation des données floues et niveau d'interrogation des données floues. L'extension au premier niveau permet de donner aux concepts flous modélisés plusieurs significations dépendamment des dimensions contextuels pouvant influer sur le sens, l'extension au deuxième niveau permet, dans la phase de traitement des requêtes, d'utiliser ces significations avec

une adaptation floue par rapport aux paramètres contextuels. Ainsi, nous pouvons représenter et manipuler des termes linguistiques du langage naturel d'une manière élargie et flexible.
Ce mémoire est organisé selon le plan suivant : dans le premier chapitre, nous présenterons tout d'abord les concepts fondamentaux des ensembles flous et des bases de données, ensuite nous présenterons les principales approches d'intégration et de manipulation des données imprécises dans les bases de données sans l'utilisation de la logique floue.

Dans le deuxième chapitre, on commencera par une classification des différents types d'information floue, puis nous présenterons les différents modèles de bases de données dites "floues" visant à intégrer la logique floue dans la représentation et l'interrogation des données. Nous présenterons également les différentes algèbres relationnelles floues et les langages d'interrogation des bases de données floues.

Dans le troisième chapitre nous détaillerons le modèle de représentation de l'information floue dans les bases de données relationnelles.

Le quatrième chapitre est consacré à notre extension du modèle GEFRED qui permet une représentation multidimensionnelle des données floues avec une couche d'adaptation par contexte et par profil ainsi que son implémentation sous un SGBDR libre qui est PostgreSQL, un cas d'application est aussi présenté, Nous terminons par une conclusion générale et les perspectives futures de notre travail.

Chapitre 1

La logique floue et les données imprécises

1. 1 introduction

Une base de données est un ensemble structuré de données, géré à l'aide d'un ordinateur. 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 stocker et qui, d'autre part offre un ensemble d'opérateurs destinés à manipuler ces données. Plusieurs modèles de bases de données ont été définis, mais aucun d'entre eux n'a pu détrôner le modèle relationnel ni se faire une place notable sur le marché.

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 des concepts généraux sur les bases de données relationnelles et leurs limites en termes d'interrogation flexible. Ensuite nous étudierons les différentes approches de représentation et d'interrogation des données imprécises.

1.2. Logique floue et théorie des sous ensembles flous

1.2.1 Historique:

Les prémisses de la logique floue sont apparues avant les années 1940, avec les premières approches, par des chercheurs américains, du concept d'incertitude. Il a fallu attendre jusqu'à 1965, pour que le concept de sous ensemble floue soit proposé par L. A. Zadeh, automaticien de réputation internationale, professeur à l'université de Berkeley en Californie, qui a contribué à la modélisation de phénomène sous forme floue, en vue de pallier les limitations liés aux incertitudes des modèles classiques à équation différentielle. En 1974, M. Mamdani expérimentait la théorie énoncée par Zadeh sur une chaudière à vapeur, matériel dont on connaît la complexité, introduisant ainsi la commande floue dans la régulation d'un processus industriel [SAM 07]. Plusieurs applications ont

alors vu le jour en Europe, pour des systèmes parfois très complexes, telle la régulation de fours de cimenterie réalisée par la société F. L. Smidt-Fuller.

Grâce au chercheur japonais M. Sugneo, la logique floue était introduite au Japon dès 1985. Les sociétés japonaises profitent de l'avantage à la fois technique et commercial de la logique floue:

- Facilité d'implantation;
- Solution de problèmes multi variables complexes;
- Robustesse vis à vis des incertitudes;
- Possibilité d'intégration du savoir de l'expert.

La théorie des sous ensembles flous est une extension de la théorie ensembliste qu'elle généralise pour pouvoir représenter les imprécisions des ensembles manipulés [RAG 03]. Nous ne donnons ici que les bases nécessaires à notre étude. Le lecteur intéressé pourra se reporter aux nombreux ouvrages sur le domaine [BMM 03][ZAD 05].

1.2.2. Ensembles, sous ensembles et sous ensembles flous :

Dans la théorie ensembliste classique un sous ensembles d'un ensemble de référence est constitué des éléments qui possèdent une ou plusieurs propriétés communes caractéristiques de ce sous ensemble. Chaque élément *e* de l'univers de référence E est ainsi caractérisé par son appartenance ou son non appartenance aux sous ensembles E_i définis [RAG 03].
Cela s'exprime par la fonction caractéristique $\varphi E_i : E \to \{0,1\}$:

$$\varphi E_i(e) = \begin{cases} 1 & si\ e \in E_i \\ 0 & \text{sinon} \end{cases} \qquad \textbf{(1.1)}$$

Par exemple si E est l'ensemble des tailles des personnes vivant en Algérie, le sous ensemble E_{grand} correspondant aux grandes tailles est défini par :

$$\varphi E_{grand}(e) = \begin{cases} 1 & si\ Taille\,(e) \geq 180cm \\ 0 & \text{sinon} \end{cases} \qquad \textbf{(1.2)}$$

Ce concept se traduit par le graphe de la figure 1.1

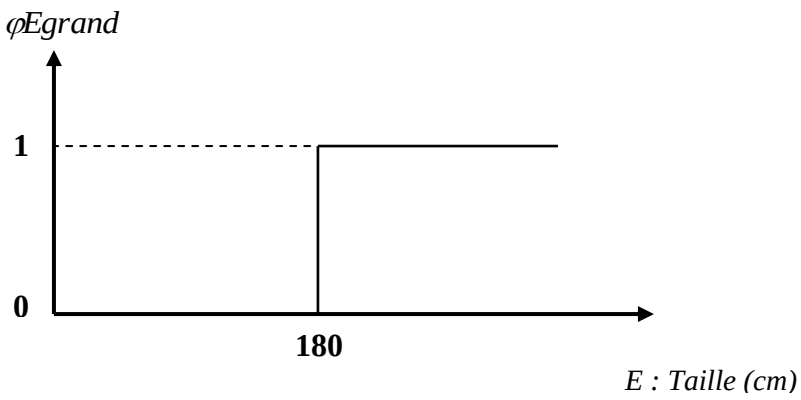


Figure 1.1 : fonction caractéristique de E_{grand}

Les limites de cette représentation apparaissent ici très clairement. La grande taille d'une personne est un concept que l'être humain est capable de formuler et d'utiliser pour raisonner. Sa définition se fonde essentiellement sur des observations et l'expérience acquise dans son environnement. Sa représentation stricte formulée par la fonction caractéristique ci-dessus n'est donc qu'une restriction du concept initial.

La formulation par les sous ensembles flous essaie de pallier les contraintes imposées par le mode de représentation précédent. Ainsi, un sous ensemble flou E_i^f étend la notion de sous ensemble en généralisant la fonction caractéristique par une fonction d'appartenance $\mu E_i^f : E \rightarrow [0,1]$ qui permet de rendre compte de l'appartenance partielle (ou encore graduelle) d'un élément au sous ensemble flou. Les fonctions d'appartenance peuvent être définies de différentes manières (cf. figure 1.2) mais leur principal objectif est de décrire le concept associé au sous ensemble flou [RAG 03]. Il existe donc une relation sémantique forte entre les deux qui rend leur extraction automatique souvent difficile.

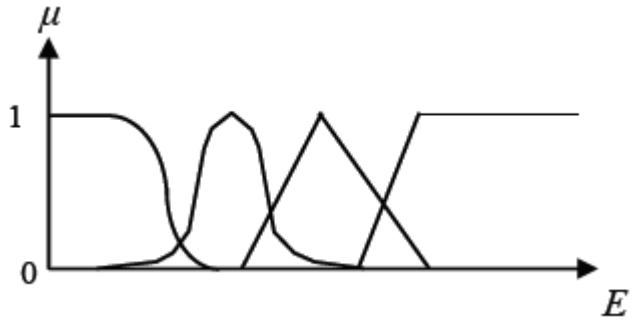


Figure 1.2 : Exemples de fonctions d'appartenance

En reprenant l'exemple sur les personnes de grande taille, la figure 1.3 illustre le graphe d'une fonction d'appartenance pour le sous ensemble flou E^{f}_{grand} :

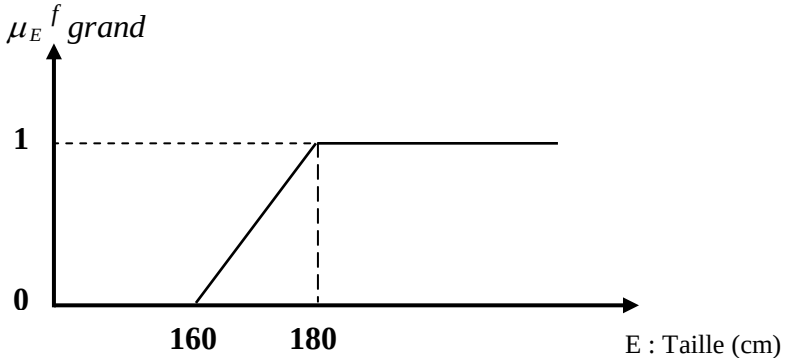


Figure 1.3 : Fonctions d'appartenance E^{f}_{grand}

La forme du sous ensemble flou défini indique que les personnes qui ont plus de 180 cm sont considérées comme étant grandes et que celles dont la taille se trouve entre 160 cm et 180 cm le sont plus ou moins.

On remarquera à ce stade de la présentation qu'en dehors de la prise en compte des imprécisions, les sous ensembles flous permettent de représenter des concepts de façon intuitive, ce qui facilite leur compréhension. De plus, par leur granularités, les ensembles flous sont parfaitement adaptés pour représenter des données de façon synthétique et compacte, notamment lorsque l'univers de référence est numérique [RAG 03]. Par exemple si un échantillon de 10000 personnes est utilisé pour étudier la taille d'une population, celui ci peut être synthétisé par trois (voir plus) sous ensembles flous (petit, moyen et grand), évitant ainsi de stocker ou de manipuler l'ensemble des données.

1.2.3. Éléments caractéristiques des sous ensembles flous

Bien que la fonction d'appartenance μ_A d'un sous ensemble flou A permet de le définir entièrement, plusieurs éléments caractéristiques sont couramment utilisés pour les manipuler, Nous les énumérons ici de manière rapide en considérant E comme étant l'univers de référence [RAG 03].

- Le noyau

Le noyau Noy (A) d'un sous ensemble flou A est l'ensemble des éléments de E dont le degré d'appartenance à A vaut 1 :

$$Noy(A) = \{e \in E : \mu_A(e) = 1\} \qquad \textbf{(1.3)}$$

- Le support

Le support Supp (A) d'un sous ensemble flou A est l'ensemble des éléments de E dont le degré d'appartenance à A est non nul :

$$Supp(A) = \{e \in E : \mu_A(e) > 0\} \qquad \textbf{(1.4)}$$

- La hauteur

La hauteur h(A) d'un sous ensemble flou A est la valeur maximale de la fonction d'appartenance μ_A :

$$h(A) = \sup_{e \in E} \mu_A(e) \qquad \textbf{(1.5)}$$

Si *h(A) =1*, on dit que le sous ensemble flou A est normalisé.

- La cardinalité

La cardinalité $|A|$ d'un sous ensemble flou A est la quantité (floue) d'éléments de E qui appartiennent à A :

$$|A| = \sum_{e \in E} \mu_A(e) \qquad \textbf{(1.6)}$$

- Les α-coupes

une α-coupe A_α d'un sous ensemble flou A est l'ensemble des éléments de E qui appartiennent à A avec un degré au moins égale à **α :**

$$A_\alpha = \{e \in E : \mu_A(e) \geq \alpha\} \qquad \textbf{(1.7)}$$

α-coupe permet à partir des sous ensembles flous de revenir à la notion des sous ensembles classiques. Ainsi A_α est un sous ensemble décrit par sa fonction caractéristique comme définie par l'équation 1.1.

La figure 1.4 illustre les principales notions définies ci-dessus [RAG 03].

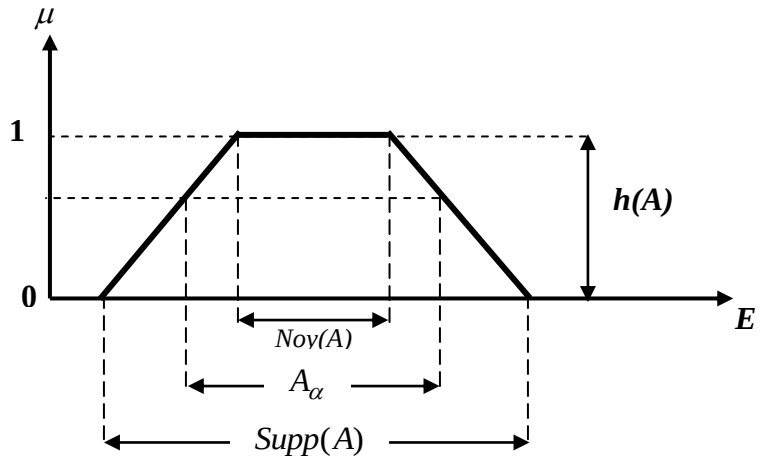


Figure 1.4 : Principaux éléments caractéristiques d'un sous ensemble flou A

1.2.4. Opérations sur les sous ensembles flous :

La théorie des sous ensembles flous étant une généralisation de la théorie ensembliste. La plupart des opérations ensemblistes ordinaires sont aussi généralisées. Nous présentons ici les principales opérations utilisées pour la manipulation de deux sous ensembles flous A et B [RAG 03].

- L'égalité :

Deux sous ensembles flous A et B sont égaux si leurs fonctions d'appartenance sont égales en tout point de E :

$$A = B \Leftrightarrow \forall e \in E, \mu_A(e) = \mu_B(e) \qquad \textbf{(1.8)}$$

- L'inclusion :

Le sous ensemble flou A est inclut dans le sous ensemble flou B si tout élément de E appartient à A avec un degré d'appartenance inférieur ou égal à celui à B.

$$A \subseteq B \Leftrightarrow \forall e \in E, \mu_A(e) \leq \mu_B(e) \qquad \textbf{(1.9)}$$

- L'union

L'union de deux sous ensembles flous A et B est un sous ensemble flou C constitué des éléments de E associés à un degré d'appartenance à C qui correspond à la plus grande valeur entre l'appartenance à A et l'appartenance à B.

$$\forall e \in E, \mu_C(e) = \mu_{A \cup B}(e) = \max(\mu_A(e), \mu_B(e)) \qquad \textbf{(1.10)}$$

Un exemple de résultat de l'union entre deux sous ensembles flous est présenté sur la figure 1.5.

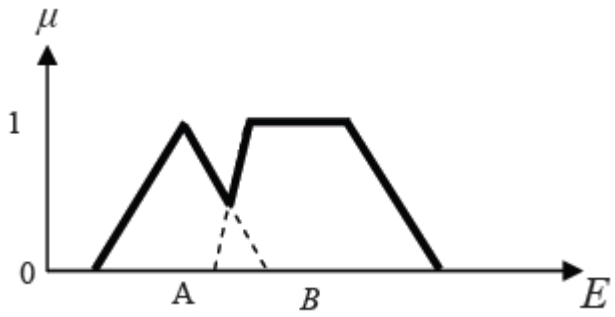


Figure 1.5 : Union de deux sous ensemble flou A et B

- L'intersection

L'intersection de deux sous ensembles flous A et B est un sous ensemble flou C constitué des éléments de E associés à un degré d'appartenance à C qui correspond à la plus petite valeur entre l'appartenance à A et l'appartenance à B.

$$\forall e \in E, \mu_C(e) = \mu_{A \cap B}(e) = \min(\mu_A(e), \mu_B(e)) \qquad \textbf{(1.11)}$$

Un exemple de résultat de l'intersection entre deux sous ensembles flous est présenté sur la figure 1.6.

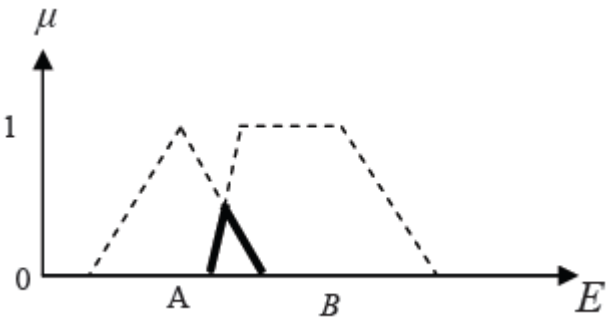


Figure 1.6 : Intersection de deux sous ensembles flous A et B

- Le complément :

Le complément $\overline{A}$ du sous ensemble flou A est un sous ensemble flou constitué des éléments de E associés à un degré d'appartenance à A qui est d'autant plus grand que le degré d'appartenance à A est faible :

$$\forall e \in E, \mu_{\overline{A}}(e) = 1 - \mu_A(e) \qquad \textbf{(1.12)}$$

- Le produit cartésien :

Les problèmes considérés sont souvent décrits dans plusieurs univers de référence $E_{1,}\ldots$, E_n et il est intéressant de pouvoir raisonner dans un univers de référence E global composé de chacun des univers initiaux. E correspond donc au produit cartésien de $E_{1,}...$, En : $E_1 \times \ldots \times E_n$ et ses éléments *e* sont des n uplets : $e= (e_{1,}..., e_n)$. Le produit cartésien de *n* sous ensembles flous $A_{1,}..., A_n$ définis respectivement sur les univers de référence $E_{1,}..., E_n$ est un sous ensemble flou *A* défini sur *E* par sa fonction d'appartenance [RAG 03]:

$$\forall e = (e_1,\ldots,e_n) \in E,\ \mu_A(e) = \min(\mu_{A_1}(e_1),\ldots,\mu_{A_n}(e_n)) \qquad \textbf{(1.13)}$$

- La projection

De façon analogue, si E est un univers de référence complexe résultant du produit cartésien d'autres univers de référence $E_{1,}..., En$, il peut être intéressant d'obtenir des informations sur un univers plus simple $F_n = F_a \times \ldots \times F_k$ avec $\{a,..., k\} \subset \{1,..., n\}$. La projection d'un sous ensemble flou A, défini sur E, sur l'univers F est un sous ensemble flou B défini sur F par [RAG 03] :

$$\forall f = (f_a,..., f_k) \in F,\ \mu_B(f) = \sup_{\{i\,/\,1\le i\le n,\, i\ne a,\ldots,\, i\ne k\}} \mu_A((...,e_i,...)) \qquad \textbf{(1.14)}$$

- Relations floues et compositions :

Une relation floue R permet de représenter un lien entre deux univers E_1 et E_2 par un sous ensemble flou défini dans $E_1 \times E_2$ par sa fonction d'appartenance $\mu_{R:}\ E_1 \times E_2 \to [0,1]$.

La composition $B = AoR$ est alors l'opération qui permet à partir de la relation *R* et d'un sous ensemble flou *A* de E_1 de déduire le sous ensemble flou *B* de E_2 défini par [RAG 03]:

$$\forall e_2 \in E_2,\ \mu_B(e_2) = \sup_{e_1} \min(\mu_A(e_1), \mu_R(e_1, e_2)) \qquad \textbf{(1.15)}$$

L'extension à des relations floues *n* aires et la composition de plusieurs relations floues sont bien entendu possibles.

1.2.5. Normes et co-normes triangulaires :

Dans la partie précédente, les opérations d'union et d'intersection ont été réalisées grâce aux fonctions min et max. Il ne s'agit en fait que de deux fonctions particulières qui appartiennent à une famille plus vaste : *les normes triangulaires* (t_normes) et les *conormes triangulaires* (t_conormes) respectivement. D'une façon générale, les t-normes et t-conormes permettent de généraliser les opérations ensemblistes d'union et d'intersection (et par extension la disjonction et la conjonction de propositions) sur les sous ensembles flous. Pour que cette généralisation soit correcte, un certain nombre de propriétés doivent être vérifiées [RAG 03] :

- T-norme

La norme triangulaire est une fonction $T:[0,1]\times[0,1]\rightarrow[0,1]$ qui pour tout élément *x,y,z, t*$\in$ *[0,1]* possède les propriétés suivantes :

- Commutativité : $T(x,y)=T(y,x)$
- Associativité : $T(x, T(y,z))=T(T(x,y),z)$
- Monotonie : $T(x,y) \leq T(z,t)$ *si* $x \leq z$ *et* $y \leq t$
- 1 est élément neutre : $T(x,1)=x$

De plus, elle assure que $\forall x, y \in [0,1]$, $T(x,y) \leq x$ *et* $T(x,y) \leq y$. Le tableau 1.1 regroupe les t-normes les plus courantes.

Nom	***Fonction***
Zadeh	$min(x,y)$
Probabiliste	x,y
Lukasiewicz	$max(x+y-1,0)$

Tableau 1.1 : Exemples de normes triangulaires

- T-conorme :

La conorme triangulaire est une fonction $\perp:[0,1]\times[0,1]\rightarrow[0,1]$ qui pour tout élément x,y,z,t $\in$ [0,1] possède les propriétés suivantes :

- Commutativité : $\perp(x,y)=\perp(y,x)$
- Associativité : $\perp(x, \perp(y,z))=\perp(\perp(x,y),z)$
- Monotonie : $\perp(x,y) \leq \perp(z,t)$ *si* $x \leq z$ *et* $y \leq t$
- 0 est élément neutre : $\perp(x,0)=x$

Elle assure aussi que $\forall x, y \in [0,1]$, $\perp(x,y) \geq x$ *et que* $\perp(x,y) \geq y$. Le tableau 1.2 regroupe les t-conormes les plus courantes.

Nom	***Fonction***
Zadeh	*max(x,y)*
probabiliste	*x+y-x.y*
Lukasiewicz	*min(x+y,1)*

Tableau 1.2 : Exemples de co-normes triangulaires

L'existence des fonctions différentes pour réaliser les normes et co-normes triangulaires n'est pas anodine. Elles ont été définies pour obtenir des comportements particuliers en fonction du type d'opération à réaliser et du contexte d'utilisation. Le choix de ces fonctions dans un système revêt donc une importance toute particulière [RAG 03].

1.2.6 Variable linguistique

En logique classique, les concepts manipulés sont décrits par des attributs qui prennent leurs valeurs sur un univers de référence. En logique floue ces concepts sont représentés par des *variables linguistiques*. Une variable linguistique est définie par un triplet (A,E_A, F_A) où A est un attribut (nom de variable), E_A son univers de référence et $F_A = \{A_1, A_2, \ldots\}$ est un ensemble de sous ensembles flous décrivant A. Les sous ensembles flous $A_1, A_2, \ldots$ sont souvent désignés par un terme linguistique (ou valeur) précisant leur rôle [RAG 03]. La figure 1.7 montre un exemple de variable linguistique associée au concept de la taille d'une personne : (taille, [0,300], {petit, moyen, grand}).

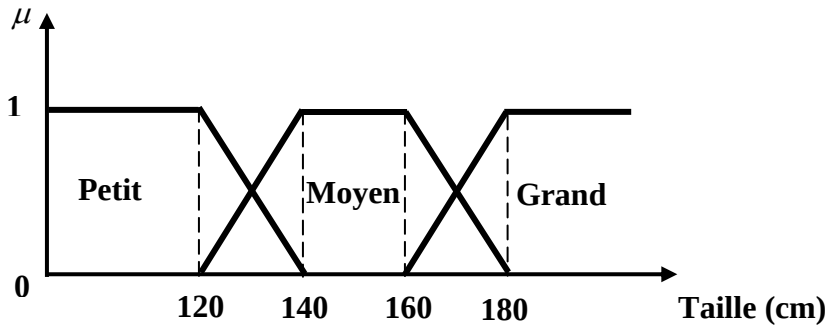


Figure 1.7 : Variable linguistique associée à la taille d'une personne

Une variable linguistique permet donc d'une part de synthétiser l'information manipulée grâce aux sous ensembles flous (modélisation qualitative) et d'autre part de représenter des concepts imprécis tels que l'homme en manipule quotidiennement. La détermination de la forme et de la position de ces sous ensembles flous est un élément essentiel pour pouvoir effectuer des raisonnements valides, robustes et compréhensibles.

C'est pour cette raison que dans beaucoup de systèmes comme ceux destinés à l'aide à la décision, les sous ensembles flous sont définis a priori par des experts du domaine afin qu'ils représentent exactement leurs connaissances. Cependant il n'est pas toujours possible d'obtenir une telle expertise, que ce soit à cause de la complexité du problème ou bien parce que les experts sont trop rares voir inexistants. Dans ces conditions, des algorithmes peuvent être mis en œuvre pour extraire automatiquement les sous ensembles flous [RAG 03]. Une expertise du résultat peut éventuellement être faite par la suite afin de déterminer la signification (le terme linguistique) des sous ensembles flous obtenus. En général, cette expertise ne sera réellement possible que si les algorithmes utilisés limitent la quantité de sous ensembles flous produits ainsi que leur chevauchements [GYP 99].

1.2.7 Distributions de possibilité

Considérons le sous-ensemble flou taille de la figure 1.7 et une variable, la taille d'un joueur dans une équipe (on note val_taille sa valeur) :

- Si l'on connait val_taille, alors, u_{Taille} (val_taille) mesure le degré d'appartenance de la taille du joueur au concept de *taille d'une personne* : *Taille* est utilisé avec la sémantique d'un sous-ensemble flou;
- Si en revanche la taille du joueur est mal connue et que l'on sait seulement qu'il s'agit d'une *taille d'une personne*, alors, u_{Taille} définit le sous-ensemble des valeurs possibles pour val_taille : c'est la sémantique d'une distribution de possibilité.

Il est important de noter cette nature commune entre sous-ensembles flous et distributions de possibilité. Bien que la sémantique sous-jacente soit différente, les deux formalismes seront traités Indifféremment par la suite [THO 03]. En particulier, on pourra comparer un critère de sélection, défini par un sous-ensemble flou, avec une donnée imprécise, caractérisée par une distribution de possibilité. Soulignons que, dans le cas classique (non flou), où le critère de sélection et la donnée

ont une valeur précise ou représentée par un intervalle, cette différence de sémantique entre le critère de sélection et la donnée existe déjà: le premier exprime la ou les valeurs recherchées, tandis que la seconde exprime la ou les valeurs existantes ou potentielles.
L'utilisation de sous-ensembles flous et de distributions de possibilité n'est qu'un assouplissement du cas classique.

Définition 1.1 Une distribution de possibilité d'une variable V définie sur un domaine X est une fonction π_V de X dans [0, 1] qui associe à chaque élément x de X le degré avec lequel il est possible que V prenne la valeur x.

π_V (x) = 0 signifie que V = x est impossible. π_V (x) = 1 signifie que V = x est complètement possible. Si pour tout x de X, π_V (x) = 1, alors toutes les valeurs sont complètement possibles pour V, ce qui représente l'ignorance totale de la valeur de V.
Comme pour un sous-ensemble flou, une distribution de possibilité est dite normalisée si sa hauteur vaut 1. Il existe alors au moins une valeur complètement possible pour V, ce qui traduit la consistance de V [THO 03].

Mesure de nécessité

La mesure de nécessité, notée N, est définie comme $N(A)=1-\Pi(\neg A)$ qu'il faut lire comme la nécessité d'un événement est égale à 1 moins la possibilité de l'événement contraire. Cette définition contient le fait que d'une manière générale, la mesure de nécessité, ou certitude, de l'événement A sur un domaine U, notée N(A), est une fonction de U dans l'intervalle réel [0,1]. Si N(A)=0 alors l'événement A est complètement incertain, si $\Pi(A)=1$ alors l'événement A est complètement certain, et si $0<N(A)<1$ alors l'événement A est plus ou moins certain [JCL 05].

1.3 Concepts fondamentaux des bases de données

L'évolution des systèmes de base de données a été initialement limitée par les conditions du traitement de données traditionnelles. Des modèles de données hiérarchiques et réseaux ont été adoptés par les SGBD comme modèles de base de données dans les années 60 et les années 70. Cependant, ces modèles ont leurs inconvénients, par contre, le modèle relationnel de base de

données, proposé par E.F. Codd en 1970 [COD 70], a une structure simple et une base mathématique solide, Il a rapidement remplacé les modèles de bases de données hiérarchiques et réseaux et est devenu le modèle de base de données dominant pour les systèmes de gestion des bases de données commerciaux et libres.

1.3.1 Le modèle relationnel

Les systèmes de gestion de base de données les plus répandus actuellement sont les systèmes relationnels, basés sur la théorie des ensembles. Cette approche ensembliste permet de donner un cadre formel pour le stockage et l'interrogation des données. Dans ces systèmes, les données sont encodées sous forme de relations, qui peuvent être vues comme des tables, elles-mêmes constituées de n-uplets, c'est-à-dire d'ensembles d'attributs, les colonnes des tables. Ces attributs sont définis sur un domaine donné, prédéfini lors de la modélisation de la relation. Une relation est donc un sous-ensemble de toutes les combinaisons possibles entre les valeurs des domaines des différents attributs. Afin de représenter le cas où la valeur d'un attribut est inconnue, une valeur nulle a été introduite. Elle représente l'ignorance totale ou l'absence de valeur, permettant ainsi de rendre certains attributs optionnels [LOI 04].

- Les clés

Les différents enregistrements de la relation (les lignes de la table) sont identifiés de manière unique par un sous-ensemble de leurs attributs, appelé clé ou identifiant. En effet, le modèle ensembliste sous-jacent implique que tous les n-uplets d'une relation doivent être distincts, et donc différer deux à deux par la valeur d'au moins un attribut, afin de pouvoir garantir un accès unique à chaque enregistrement. Il est donc possible de trouver un groupe minimal d'attributs caractéristiques de chaque n-uplet .

1.3.2 Algèbre relationnelle et manipulation des données

Afin de pouvoir manipuler ces relations, et en particulier reconstruire les données originales à partir des différentes relations et des dépendances définies dans le schéma relationnel, l'algèbre relationnelle a été élaborée. Elle utilise donc deux aspects du schéma tel qu'il a été défini : le point de vue ensembliste des relations et leur structure [LOI 04].

- **Opérations ensemblistes**

Les relations étant basées sur des ensembles, les opérateurs classiques des ensembles s'appliquent. Il est ainsi possible d'effectuer des unions, intersections et différences de relations, dans la mesure où ces opérations s'appliquent à un sous-ensemble commun des attributs des différentes relations mises en jeu. Le produit cartésien des relations est aussi possible. Ces relations peuvent être des relations réelles du schéma ou des ensembles de n-uplets résultant d'autres opérations antérieures, comme une jointure par exemple.

- **Opérations relationnelles**

La tâche principale de manipulation des données, concernant leur consultation, est de reconstruire les n-uplets désirés à partir des relations et des dépendances fonctionnelles, et de sélectionner ceux contenants les valeurs intéressantes. Différents opérateurs relationnels ont cette fonction :

Sélection : La sélection consiste à ne considérer que les n-uplets ayant une valeur particulière pour un ou plusieurs de ses attributs. Le critère de sélection est une fonction sur le domaine de l'attribut, généralement une comparaison avec une valeur x fixée.

Projection : La projection ne conserve qu'un sous-ensemble des attributs de la relation.

Jointure : La jointure est l'opérateur clé de l'algèbre relationnelle. En effet, c'est elle qui permet d'exploiter les dépendances fonctionnelles pour reconstruire les n-uplets précédemment décomposés. Concrètement, la jointure de deux relations r et s sur les attributs A et B par l'opérateur de comparaison θ est le sous-ensemble de l'union des n-uplets de r et s pour lesquels les valeurs de A et B respectent $A \theta B$. Considérons l'exemple de la commande décomposée en trois relations *commande* , *client* et *produit,* il est ainsi possible de retrouver la relation initiale complète en effectuant une jointure entre *commande* et *produit* sur l'égalité des identifiants de produits, puis une jointure entre ce résultat et *client* sur l'égalité des identifiants de clients.

Division : Le but de la division est de trouver les n-uplet d'une relation pour lesquels la valeur d'un attribut apparaît dans une deuxième relation. Dans le cadre de notre exemple, une division pourrait se formuler par une requête de type *trouver les clients ayant commandé plus de 100 pièces de tous les produits valant plus de 1500 DA*. Elle peut se décrire à l'aide des opérateurs déjà mentionnés.

- **Les requêtes**

Les requêtes sur ces systèmes sont définies à l'aide des opérations précédentes, combinées successivement. Les critères utilisés dans la sélection et la jointure peuvent en outre être agrégés par les opérateurs booléens classiques. Généralement, les systèmes autorisent aussi des opérations de regroupement des n-uplets en fonction de la valeur d'un ou plusieurs attributs, accompagnées de fonctions s'appliquant sur des ensembles d'éléments, comme le calcul de la moyenne de la valeur d'un attribut par exemple.
Ainsi, des requêtes élaborées peuvent être construites par l'utilisateur afin d'extraire des données de la base, voir obtenir de nouvelles données par le calcul [LOI 04].

1.3.3 Le langage SQL

Le langage SQL (Structured Query Language) peut être considéré comme le langage d'accès normalisé aux bases de données [HMA 11]. Il est aujourd'hui supporté par la plupart des produits commerciaux ou libres. Il a fait l'objet de plusieurs normes ANSI/ISO dont la plus répandue aujourd'hui est la norme SQL2 qui a été définie en 1992. Nous décrivons ici les principaux aspects de cette norme.

Une requête SQL se présente généralement sous la forme " SELECT ... FROM ... WHERE " :

- la clause **SELECT** exprime le résultat attendu sous la forme d'une liste d'attributs auxquels il est possible d'appliquer différents opérateurs et fonctions.
- la clause **FROM** liste les relations utilisées pour évaluer les requêtes;
- la clause **WHERE** qui est facultative énonce une condition que doivent respecter les enregistrements sélectionnés.

Le langage SQL comporte :

- Un langage de définition des données (LDD) qui permet de définir des relations, des vues externes et des contraintes d'intégrité;
- Un langage de manipulation des données (LMD) qui permet d'interroger une base de données sous forme déclarative sans se préoccuper de l'organisation physique des données;

- Un langage de contrôle des données (LCD) qui permet de contrôler la sécurité et les accès aux données.

Le succès du langage SQL est dû essentiellement à sa simplicité et au fait qu'il s'appuie sur le schéma conceptuel pour énoncer des requêtes en laissant le SGBD responsable de la stratégie d'exécution [HMA 11].

1.4 Traitement de l'imprécision dans les bases de données

Le modèle relationnel, avec les outils qu'il dispose, permet une modélisation puissante des cas concrets des données. Cependant, il ne permet de gérer que les données atomiques et parfaitement connues. De plus, la recherche est basée sur des opérateurs ensemblistes stricts, et ne permet que la formulation de requêtes booléennes.

Dans cette section, nous allons aborder les approches permettant le traitement des données imprécises dans le domaine des bases de données afin d'étendre le modèle relationnel classique et permettre ainsi plus de flexibilité et ceci sans utiliser la théorie des ensembles flous.

1.4.1 Valeurs Nulles

La première représentation des données imprécises dans les bases de données a été proposée par [COD 79], et traite des valeurs nulles. Codd définit une logique trivaluée introduisant la valeur nulle ($\perp$). Ici, cette valeur nulle a le sens d'inconnue.

Il introduit donc les prédicats suivants [LOI 04] :

1. $x\ \theta\ y \equiv \perp$ *si* x *ou* y *sont nuls et* θ *est* $\prec, \leq, =, \neq, \geq, \succ,$

2. $x \in S \equiv \perp$ si x est nul pour tout sous-ensemble S du domaine, c'est-à-dire si on ne sait pas si x appartient ou non à S.

3. $T \subseteq S \equiv \perp$ *si* $x \in T$ et x est nul, pour tout sous-ensemble S.

Ainsi que les extensions aux opérateurs logiques :

- $\perp \vee T = T, \perp \vee F = \perp, \perp \vee \perp = \perp$
- $\perp \wedge T = \perp, \perp \wedge F = F, \perp \wedge \perp = \perp$
- $\neg \perp = \perp$

Dans [GES 91], cette approche est étendue pour gérer plusieurs interprétations des valeurs nulles. Des valeurs par défaut sont attribuées aux données manquantes, et un degré de vérité leur est associe. Une logique quadrivaluée est utilisée, de façon à ce que le statut logique des données par défaut, représenté par leur valeur de vérité, puisse être interprété comme vrai et faux, mais aussi inconnu et inapplicable [LOI 04].

Un calcul complètement compositionnel par rapport à tous les connecteurs n'est cependant pas compatible avec une modélisation satisfaisante de l'information manquante. Voir par exemple à ce sujet [DET 04]. Une nouvelle approche de modélisation des valeurs nulles basée sur le traitement sémantique par rapport au contexte est présentée dans [MGT 10].

1.4.2 Intervalles

Le problème des intervalles de valeur est abordé dans [GRA 80]. Pour les valeurs numériques, trois types sont gérés : une valeur simple si l'information est complète, une paire représentant l'intervalle des valeurs possibles, et une valeur nulle si l'information est manquante. Le prédicat *peut-être* est introduit à des fins de comparaisons, et signifie « vrai ou peut-être vrai ». il s'applique lorsque la valeur recherchée est dans l'intervalle défini pour la valeur de l'élément.

Ceci illustre un aspect problématique des bases de données obtenues. En effet, les n-uplets des relations de la base de données sont normalement uniques, mais dans ce cas, deux n-uplets ayant les mêmes intervalles de valeurs peuvent en fait avoir des valeurs réelles différentes, et donc être distincts. Ainsi, on considère qu'il peut y avoir autant de n-uplets que de valeurs possibles dans l'intervalle sans que la contrainte d'unicité ne soit violée [LOI 04].

1.4.3 Approche basée sur la logique modale

En définissant $\|Q\| \; et \; \|T\|$ comme les ensembles des objets du monde réel pouvant être représentés par respectivement la requête Q et le n-uplet T, [LIP 79] définit trois catégories dans lesquelles les n-uplets sont classée en fonction de la requête :

1. $T \in \{certain\} si \|T\| \subset \|Q\|$
2. $T \in \{possible\} si \|T\| \cap \|Q\| \neq \phi$
3. $T \in \{rejeté\} si \|T\| \cap \|Q\| = \phi$

Par exemple, avec une relation *HOTEL (ID,Prix)*, le n-uplets T=[1,40-50] peut représenter onze objets du monde réel (en supposant les prix entiers). Ainsi, la requête *HOTEL [prix<45]* place T dans l'ensemble des résultats possibles, et la requête *HOTEL [prix > 30]* $\wedge$ *HOTEL [prix < 60]* le place dans l'ensemble des résultats certains [LOI 04].

1.4.4 Bases de données statistiques et probabilistes

L'utilisation d'un modèle de données probabilistes est une manière naturelle de traiter les données incertaines [LOI 04]. Le modèle développé dans [BGP 92] est un des plus aboutis. Chaque attribut stochastique est considéré comme une fonction discrète de distribution de probabilité, et les valeurs des attributs du n-uplet sont normalisées[1]. Le concept de probabilité manquante est introduit pour palier la difficulté de déterminer une probabilité pour toutes les valeurs possibles du domaine. Elles permettent ainsi de représenter l'incertitude de la valeur des données.

[PCK 03] présente une autre extension du modèle relationnel pour traiter les données incertaines, en attachant a chaque tuple dans la BDD une probabilité, seulement, au lieu d'avoir une probabilité par élément, on associe une probabilité totale. Dans [RSG 05], on présente une extension du cadre probabiliste pour faire des calculs d'agrégation sur des données probabilistes.

Cependant, l'approche probabiliste est peu adéquate pour représenter de l'information imprécise et incomplète, car en probabilité, l'absence complète de certitude qu'un énonce soit vrai implique nécessairement la certitude que l'énoncé contraire est vrai, ce qui est discutable et d'ailleurs faux dans l'approche précédente, comme dans l'approche possibiliste [LOI 04].

1.5. Requêtes flexibles

Une requête flexible spécifie des préférences au sein des conditions élémentaires, et des priorités entre ces conditions, permettant une transition plus douce entre la satisfaction totale de la requête et le rejet de l'élément. Ainsi, le résultat de la requête n'est plus un ensemble d'éléments sélectionnés, mais un ensemble d'éléments discriminé en fonction de leur satisfaction globale [LOI 04].

[1] Une probabilité est dite normalisée si La somme des probabilités de toutes les valeurs probables est égale à 1

Le problème de l'expression des préférences utilisateurs dans les requêtes a reçu beaucoup d'attention ces dernières années [BOS 08,10b] [LIR 09a,09b] [HKP 08] [VIA 06] [KOL 04].

En général, il est possible de distinguer deux méthodes d'expression des préférences. Une méthode qualitative où les préférences sont spécifiées par des relations binaires de préférence comme les formules logiques et les constructeurs de préférences spécifiques [KIE 02] [CHO 03]. Une méthode quantitative où les préférences sont spécifiées en assignant des scores numériques définissent des degrés d'intérêt d'un utilisateur à ces objets [AWI 00] [KOL 04].

1.6 Approches de modélisation des requêtes flexibles

Les travaux visant à exprimer des préférences dans les requêtes flexibles peuvent être regroupés en quatre catégories [BUC 06] :

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.

1.6.1 Critère complémentaire de classement

La première extension proposée consiste à ajouter des critères supplémentaires aux critères de sélection afin de permettre de réaliser une classification quantitative des n-uplets résultats de sélection. De cette façon une requête comporte deux parties, une condition principale qui vise à sélectionner un ensemble de n-uplets de la base de données et une partie relative à l'explication des préférences, toutes les deux basées sur des expressions booléennes [ALA 07].

Un système comme PreferenceSQL [KIK 02] met en œuvre cette technique. Des préférences sur les critères sont définies lors de leur combinaison. Leur juxtaposition traduit qu'ils ont tous la même importance, alors qu' une combinaison en cascade leur attribue des importances relatives de plus en plus faibles. Cela a pour conséquence de produire un ordre partiel sur les résultats qui sont regroupés en différentes classes non comparables entre elles. Par contraste, dans

des systèmes comme Deduce2 [CHA 82] ou Preferences [LAC 87], le critère secondaire exprimant les préférences produit un ordre total sur les résultats d'une requête.

1.6.2 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 ≈) étend l'égalité stricte et la condition élémentaire, A≈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 [MAB 05].

Plusieurs systèmes, comme Ares [ICH 86] ou Vague [MOT 89], sont basés sur de tels opérateurs de similarité. Ces systèmes posent des problèmes du fait du caractère ad hoc du calcul associé à l'évaluation de conjonctions ou de disjonctions au sein de requêtes complexes.

1.6.3 Préférences avec des termes linguistiques

Dans cette approche, l'idée est d'associer des termes de préférences aux conditions booléennes figurant dans la requête [MAB 05]. Parmi les systèmes qui se basent sur cette approche, nous pouvons citer MULTOS [RAB 90].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 : $\begin{cases} P_{k,j} \times i_k \; si \; X_k = v_{k,j} \\ 0 \qquad sinon \end{cases}$

Pour tout critère de recherche de la forme :

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

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 [MAB 05].

1.6.4 Approche basée sur les ensembles flous

Dans cette approche, Les préférences sont prises en compte au travers de l'expression de prédicats flous commensurables. De tels prédicats, basés sur un référentiel commun d'interprétation, peuvent être combinés au moyen d'une riche palette d'opérateurs de la logique floue dont certains, traduisant par exemple des effets de compensations ou d' importances relatives entre critères, n'ont pas de correspondant dans la logique booléenne. La sélection des données et l'exploitation de préférences pour les ordonner sont donc fusionnées et les résultats sont totalement ordonnés. Il a été montré [BOS 94] qu' une telle approche englobe les capacités d' expression de Deduce2, Preferences, Ares et Vague. L'approche PreferenceSQL peut également être considérée dans le cadre des ensembles flous moyennant l'introduction dans le langage d'opérateurs d'ordonnancement lexicographique de type leximin ou discrimin [YAW 05].

La théorie des ensembles flous a été utilisée pour exprimer des préférences dans les conditions des requêtes flexibles aussi bien que pour exprimer des préférences entre les conditions [GAL 08b]. Tahani [TAH 77] a introduit le concept de relation floue dans les SGBDs. 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 représentés par des sous-ensembles flous. Cette approche a été étendue pour permettre l'interrogation floue dans le SGBD Microsoft Access [KAC 94].

Bosc et Pivert [BOS 95] ont présenté une extension du langage relationnel SQL nommée SQLf pour les différents types de requêtes SQL (requêtes imbriquées, requêtes de division), le langage SQLF a fait l'objet de plusieurs travaux d'extensions [BOS 04, 10a, 11].

Galindo a proposé un langage appelé FSQL (Fuzzy Structured Query Language) [GAL 99, 06] qui permet la représentation et l'interrogation des données dans une base de données relationnelle floue basée sur le modèle GEFRED (GEneralized model for Fuzzy RElational Databases). Ce langage étend SQL pour introduire des quantificateurs flous, des constantes floues et des comparateurs flous.

Les deux derniers travaux seront détaillés dans le chapitre suivant.

1 .7 Conclusion

Le modèle relationnel permet une modélisation puissante des cas concrets des données. Cependant, il ne permet de gérer que les données atomiques et parfaitement connues. Les données mal connues, imprécises et entachées d'erreurs sont donc difficiles à représenter directement dans le cadre de ce modèle.

La logique floue présentée dans ce chapitre s'avère un cadre très approprié pour le traitement de l'imprécision, dans le chapitre suivant nous allons présenter deux concepts fondamentaux qui ont été introduit afin d'étendre le modèle relationnel classique et de permettre ainsi plus de flexibilité, à savoir les bases de données floues et les langages d'interrogation des bases de données floues.

Chapitre 2

Les bases de données floues

2.1 Introduction

Quand la seule information disponible est imprécise, incertaine ou même vague, il semble plutôt nécessaire d'essayer de la représenter telle qu'elle est, et de le stocker dans une base de données afin de l'exploiter pour répondre à des requêtes de besoin.

La nécessité de modéliser l'imprécision dans les bases de données peut être vue clairement dans le traitement des requêtes flexibles, des requêtes qui ne sont pas précises. Il peut y avoir plusieurs raisons de telles requêtes. En premier lieu, l'ambigüité dans la façon avec laquelle la requête est énoncée comme l'exemple d'une requête exprimée en langage naturel, la connaissance contextuelle et pragmatique peut alors aider à choisir la traduction la plus plausible. Les requêtes de langage naturel peuvent également impliquer des mots ou des expressions avec des significations vagues. Ceci peut être motivé par le besoin d'expression des préférences progressives entre les valeurs admissibles. L'utilisateur peut également être incertain de ce qu'il recherche. Dans tous ces cas, le traitement des requêtes n'est pas trivial.

La théorie des ensembles flous et de possibilité [ZAD 65, 78] offrent un cadre unifié qui permet de fournir une solution à la manipulation des requêtes flexibles et à la gestion d'information imprécise et incertaine. Un système de requête floue est une interface aux utilisateurs pour obtenir l'information de la base de données en utilisant des phrases du langage (quasi) naturel [HUM 09].

Dans ce chapitre, nous présenteront les principaux modèles des bases de données dites "floues" visant à intégrer les concepts de la logique floue dans la représentation des données. Ainsi que les différentes algèbres relationnelles floues et les langages d'interrogation des bases de données floues.

2.2 Type d'informations floues

Lorsqu'on parle de l'information floue, plusieurs termes sont utilisés, nous distinguons quatre types principaux d'imperfections qui peuvent toucher l'information, à savoir : l'incertitude, l'imprécision, le vague et l'inconsistance. Nous allons détailler ces différents concepts.

- **Incertitude**

L'incertitude est relative à l'information ou à la source [MAR 05]. Elle est liée au manque d'informations disponibles pour déterminer si un état classique[1] est réellement vrai ou faux [BOS 97]. Dans ce cas, la meilleure solution est d'essayer d'estimer la tendance de la déclaration pour être vrai ou pour être faux, plusieurs cadres sont possibles :

- les approches numériques telles que la théorie des probabilités, la théorie de possibilité, les fonctions de croyance, et autres méthodes.
- des méthodes de déduction purement symboliques qui utilise les mécanismes non-classiques pour produire des conclusions plausibles malgré le manque partiel d'information

- **Imprécision**

Une information est dite imprécise si elle s'avère insuffisante pour l'usage qu'on voudrait en faire et notamment pour répondre à une question donnée de manière claire. C'est en général le contenu d'une information, ce à quoi elle réfère, qui est imprécis. Elle se traduit soit par une hésitation face à plusieurs choix entre lesquels on ne peut pas trancher, soit par des valeurs mutuellement exclusives mais dont aucune n'est exclue.

Cette imprécision peut provenir par exemple :

[1] Un état classique peut être soit Vrai ou Faux

- d'une mauvaise connaissance des informations numériques qui peut être la conséquence d'une insuffisance des instruments d'observation (4000 à 5000 touristes), d'erreurs de mesures (poids à 1% près) ou encore d'une connaissance approximative (le prix d'un bon ordinateur familial est environ entre 40000 et 50000 DA),
- d'un contenu informationnel représenté par des connaissances approximatives et vagues en termes de langage naturel.

Cependant, l'imprécision est une notion qui dépend du contexte, ceci se traduit par le fait qu'une information qui est précise dans un contexte, peut ne plus l'être dans un autre contexte. Cela dépend de la granularité de description du référentiel des situations considérées [DUP 94].

L'incomplétude de l'information est aussi considérée comme un aspect de l'imprécision. Elle traduit l'indisponibilité et le manque d'information à un instant t s'exprimant via :

- un niveau de connaissance limité sur les alternatives qui peut aller jusqu'à l'ignorance totale en l'absence de toute évaluation sur une alternative,
- un processus d'acquisition de l'information dont on ne contrôle pas la dynamique.

Remarque : Il est possible de trouver l'imprécision et l'incertitude du second ordre si la valeur de degré de vérité ou d'une mesure d'incertitude (probabilité, possibilité, etc.) soit sans précision, vaguement énoncée, ou incertaine.

▪ Aspect vague

Une déclaration vague contient des attributs vagues ou graduels, elle peut également inclure des quantificateurs vagues [BOS 97].

En fait, la signification d'un attribut vague dépend du contexte : un grand papillon est plus petit qu'un petit éléphant ; le contexte peut même dépendre de l'utilisateur. Ainsi, la signification du mot "Jeune" par exemple, peut changer en fonction du contexte.

Une déclaration telle que " Ali est jeune " n'est pas nécessairement vrai ou faux, elle peut être utilisée dans deux situations complètement différentes : L'âge est connu avec précision, par exemple il a 30 ans, et la phrase alors reçoit un degré de vérité qui estime dans quelle mesure la valeur "30" correspond à la représentation des " jeunes " dans le contexte considéré ; ou le fait que "Ali est jeune" est la seule information disponible sur l'âge de cette personne et la phrase représente ainsi une

contrainte flexible sur les valeurs acceptables pour son âge. Dans les deux cas, une relation d'ordre entre les valeurs compatibles avec le concept "jeune" est définie.

- **Inconsistance**

On parle de l'inconsistance de l'information lorsque plusieurs informations sont en conflit en présence de redondance. Elle traduit le fait que les informations sont contradictoires et peuvent ainsi induire une incohérence dans la conclusion. Elle s'explique par le fait que [DEA 07]:

- Les informations proviennent d'une multitude de sources pouvant entrainer une divergence des points de vue,
- Des informations récentes et d'autres moins récentes coexistent dans la même base de données.

Dans [ZML 08], on parle d'un autre type qui est l'ambigüité, qui signifie que le manque de quelques éléments nécessaires pour une interprétation complète mène à plusieurs interprétations possibles.

2.3 Modèles de bases de données floues

Les modèles classiques de base de données souffrent souvent de leur incapacité de représenter et de manipuler des informations imparfaites qui peuvent se produire dans beaucoup d'applications du monde réel. Le problème de représentation et de traitement de l'information imparfaite n'est pas trivial, parce qu'il exige une modification au niveau de la structure des relations, ainsi qu'au niveau des opérations. Un grand objectif des recherches menées dans le domaine des bases de données a été l'association de sémantique supplémentaire dans le modèle de base de données [ZMA 06].

Pour résoudre le problème des données imprécises ou incertaines, et pour permettre d'élaborer des requêtes plus flexibles, le monde des bases de données s'est tourné vers les techniques issues de la logique floue. L'utilisation de ces techniques amène ainsi à ce qu'on appelle les modèles de bases de données floues. Ces modèles peuvent être classifiés selon trois approches [RZK 06], à savoir : l'approche basée sur les relations de similitude comme le modèle de Buckles-Petry, l'approche basée sur la théorie des possibilités comme le modèle de Prade-Testemale, le modèle d'Umano-Fukami et

le modèle de Zemankova-Kandel et l'approche hybride comme le modèle GEFRED. Nous allons présenter les principes de ces modèles.

2.3.1 Modèle de base

Le modèle de base des bases de données relationnelles floue est considéré le modèle le plus simple, il consiste à ajouter un degré dans l'intervalle [0, 1] à chaque instance (ou tuple). Ceci permet de maintenir l'homogénéité de données de la base de données. Néanmoins, la sémantique associée à ce degré déterminera son utilité, et cette signification sera utilisée dans le processus d'interrogation.

Ce degré peut avoir la signification du degré d'appartenance de chaque tuple à la relation comme il peut avoir d'autres significations comme le niveau de dépendance entre deux attributs ou le degré de réalisation d'une condition [GAL 06].

Le problème principal de ces modèles flous est qu'ils ne tiennent pas compte de la représentation d'informations imprécises sur un certain attribut d'une entité spécifié (comme par exemple les valeurs ''grande'' ou ''courte '' pour un attribut de taille). En outre, le caractère flou est assigné globalement à chaque instance (tuple), ce qui rendre impossible de déterminer la contribution floue spécifique de chaque élément constituant un attribut [GAL 06].

2.3.2 Modèles basés sur les relations de similitude

Le modèle de Boucles-Petry [BUP 82][PET 96] est le premier modèle qui utilise des relations de similitude dans le modèle relationnel. Dans ce modèle, une relation floue est définie comme un sous-ensemble du produit cartésien suivant : $P(D_1) \times \ldots \times P(D_M)$, où $P(D_i)$ représente les sous domaines d'un domaine D_i, y compris tous les sous-ensembles qui pourraient être considérés dans le domaine D_i. Les types de données autorisés dans ce modèle sont les suivants:

- Ensemble fini de scalaires (étiquettes).
- Ensemble fini de nombres.
- Ensemble de nombres flous.

La signification de ces ensembles est disjonctive, c.-à-d., la valeur réelle est une valeur appartenant à l'ensemble. Les types d'équivalence sur un domaine sont construits d'une relation de similitude, dans laquelle l'utilisateur fournit des valeurs prises par une telle relation. Typiquement,

ces valeurs de similitude sont normalisées dans l'intervalle [0.1], où 0 correspond à "complètement différent" et 1 correspond à "complètement similaire".

Un seuil de similitude peut être établi, avec une valeur entre 0 et 1, afin d'obtenir les valeurs dont la similitude est plus grande que le seuil ou de considérer ces valeurs non distinguables [GAL 06].

Dans [BEL 05,06a, 06b], Belohlavek & Vychodil introduits le concept des relations ordonnées sur un domaine avec similitude, cette approche permet à la fois de saisir des seuils de similitude et des validités approximatives des dépendances fonctionnelles. Cette approche est souple et a une plus grande capacité expressive. Les seuils de similitude n'ont pas besoin d'être fixés, en plus, un ensemble de connecteurs logiques peuvent être utilisés, plutôt qu'un seul connecteur.

L'utilisation des relations de similitude fournit un outil approprié et intuitif pour représenter l'imprécision des données. Cependant, ce modèle a plusieurs inconvénients [MAB 05] :

- 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 base de donnée.
- Il peut donner lieu à des résultats confus lors de l'application des opérateurs relationnels aux classes d'équivalences.
- Il présente un résultat qui peut avoir plusieurs interprétations.

2.3.3 Modèles possibilistes

Dans le cadre des bases de données relationnelles, le modèle possibiliste offre un cadre unifié pour représenter les valeurs précises et les valeurs imprécises telles que les intervalles usuels ou flous et les valeurs nulles. On considère que la valeur d'un attribut d'un n-uplet peut être décrite par une distribution de possibilité qui représente les valeurs plus ou moins préférées que l'attribut puisse prendre.

L'un des intérêts de ce modèle est sa compatibilité avec l'expression de requêtes floues puisque la sémantique de préférence utilisée pour les valeurs candidates des données mal connues et pour les critères de sélection est la même. Cet aspect a été exploité dans les travaux de Prade et Testemale [PRT 84] qui sont à l'origine de l'application de l'approche possibiliste aux bases de données.

Dans [BOS 05d, 09], d'autres modèles possibilistes ont été présentés dans lesquels la théorie de possibilité est utilisée pour représenter les valeurs qui sont plus ou moins certaines au lieu de ceux qui sont plus ou moins possibles en utilisant un niveau de certitude à chaque partie de données.

Nous allons présenter les principaux modèles qui sont à l'origine de l'application de l'approche possibiliste aux bases de données.

▪ Modèle de Prade et Testemale

Prade et Testemale ont proposé un modèle de base de données relationnelle floue qui permet l'intégration des données incomplètes ou incertaines en utilisant la théorie de possibilité [PRT 84].
Ce modèle 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.
- Ensemble fini de nombre.
- Ensemble de nombres ou étiquettes flous.

Ces domaines peuvent prendre les valeurs suivantes :

- Une valeur unique $d \in D$ parfaitement connue. Exemple âge=19. Par contre, ce modèle ne considère pas le cas de différentes valeurs parfaitement connues (attributs multi-valués) comme par exemple langue_parlé = français et anglais.
- Une valeur nulle qui n'inclut pas la valeur inconnue et non applicable.
- Une distribution de possibilité sur le domaine de l'attribut. Par exemple, Hauteur = "pas très haut", où *pas très haut* est 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}

La représentation des différents types de données dans ce modèle est résumée dans le tableau 2.1.

▪ Modèle de Umano-Fukami

Cette proposition [UMA 82 ,94] utilise également les distributions de possibilité afin de modéliser les informations incomplètes. Les valeurs utilisées dans ce modèle sont :

- les distributions de possibilité
- les valeurs indéterminées, inconnues et nulles 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}

Information	**Modèle de Prade-Testemale**	**Modèle d'Umano-Fukami**
La donnée précise est connue et exacte : c	$\pi_{A(x)}(e) = 0$ $\pi_{A(x)}(c) = 1$ $\pi_{A(x)}(d) = 0, \quad \forall d \in D, d \neq c$	$\pi_{A(x)}(d) = \{1 / c\}$
Inconnue mais applicable	$\pi_{A(x)}(e) = 0$ $\pi_{A(x)}(d) = 1, \quad \forall d \in D$	Inconnue
inapplicable	$\pi_{A(x)}(e) = 1$ $\pi_{A(x)}(d) = 0, \quad \forall d \in D$	indéfinie
Ignorance totale	$\pi_{A(x)}(d) = 1, \quad \forall d \in D \cup \{e\}$	Nulle
Intervalle [m, n]	$\pi_{A(x)}(e) = 0$ $\pi_{A(x)}(d) = 1 \quad if \ d \in [m, n] \subseteq D$ $\pi_{A(x)}(d) = 0 \quad in \ other \ case$	$\pi_{A(x)}(d) = 1 \quad si \ d \in [m, n] \subseteq D$ $\pi_{A(x)}(d) = 0 \quad sinon$
L'information disponible est une distribution de possibilité μ_a	$\pi_{A(x)}(e) = 0$ $\pi_{A(x)}(d) = \mu a(d) \quad \forall d \in D$	$\pi_{A(x)}(d) = \mu_a(d) \quad \forall d \in D$
La possibilité d'inapplica--bilité est λ, Si applicable la donnée est μa	$\pi_{A(x)}(e) = \lambda$ $\pi_{A(x)}(d) = \mu a(d) \quad \forall d \in D$	Sans représentation

Table 2.1 : Représentation de l'information dans les modèles Prade-Testemale et Umano-Fukami

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.

Concernant les requêtes, ce modèle permet l'expression des requêtes en termes exactes aussi bien que flous, le modèle résout le problème de requête en divisant l'ensemble d'instances impliquées dans la relation en trois sous-ensembles [GAL 06] :

- 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.
La représentation des différents types de données dans ce modèle est résumée dans le tableau 2.1.

▪ Modèle de Zemankova-Kandel

Le modèle de Zemankova-Kandel [ZEK 84] se compose de trois parties :

- Une base de données de valeurs, dans laquelle des données sont organisées dans une forme similaire aux modèles possibilistes.
- Une base de données explicative, dans laquelle les sous-ensembles floues et la relation floue utilisée sont stockées.
- Un ensemble de règles de traduction pour la manipulation des adjectifs et des modificateurs.

La requête est posée d'une façon similaire à celle dans le modèle de Prade-Testemale, sauf pour la mesure de possibilité utilisée pour trouver la compatibilité du sous-ensemble floue F de la condition, avec une valeur de l'attribut A pour chaque tuple dans la relation, est donnée par l'équation suivante :

$$P_A(F) = sup_{u \in D} \{\mu_F(u).\ \pi(u)\} \qquad \textbf{(2.6)}$$

La mesure de certitude est donnée par l'équation suivante:

$$C_A(F) = max_{u \in D} \{0,\ inf \{\mu_F(u).\pi_A(u)\}\} \qquad \textbf{(2.7)}$$

La certitude est utilisée à la place de la nécessité dans le modèle de Prade-Testemale. Cependant, l'interprétation du degré de certitude est peu claire et aucune relation entre la possibilité et la certitude, par contre une relation existe entre la possibilité et la nécessité : N (X) = 1 - P (¬X).

Le résultat d'une requête est présenté en tant que relations floues contenant deux champs, dans lesquelles les valeurs de possibilité et de certitude pour chaque instance sont incluses pour une requête donnée. Des seuils minimaux peuvent être établis pour ces relations [GAL 06].
Sur des conditions imposées en cas de sélection, les points de départ sont des relations de similitude définies sur D x D, à partir duquels n'importe quelle autre relation comparative est établie. Néanmoins, ce modèle a quelques limitations importantes, qui le font un modèle incomplète.

2.3.4 Modèles possibilistes étendus

Comme alternative aux mesures de possibilité et de nécessité, les valeurs de vérité possibilistes étendues (EPTVs) peuvent être utilisées pour mesurer l'incertitude [DET 02]. Le concept d'un EPTV est une extension du concept d'une valeur de vérité possibiliste (PTV), L'idée de base derrière les EPTVs est qu'on considère un univers I∗ de trois valeurs de vérité : (T) vrai, (F) faux et Indéfinie (⊥), ce dernier étant une pseudo-description pour les cas où l'information est inapplicable et une valeur de vérité booléenne ne pourraient pas être utilisés sans perte d'information [TRC 06].

Sur la base des valeurs de vérité possibilistes étendues, [DET 03] propose une approche possibiliste étendue. Dans [TRC 06], L'utilité d'un cadre logique basé sur EPTVs pour supporter la modélisation et l'interrogation flexible des bases de données est discutée.

Ce cadre facilite le traitement de l'information inapplicable, des contraintes d'intégrité inapplicables et des critères des requêtes inapplicables. Il permet une meilleure prise en compte de l'information imparfaite ou manquante, ainsi qu'une meilleure modélisation de l'information inconnue par rapport aux approches classiques basées sur la logique trivaluée ou quadrivaluée.

2.3.5 Modèles hybrides

L'approche hybride constitue un cadre général qui utilise les avantages des approches précédentes. Takahashi [TAK 93] a proposé un modèle de base de données relationnelle floue qui

utilise les distributions de possibilité pour représenter les valeurs des attributs, et les ensembles floues comme valeurs de vérité des tuples.

Medina et al. [MED 94a] ont proposé un modèle de base de données floue nommée GEFRED qui a été étendu [GAL 06] [GTB 09]. Ce modèle définit une extension des relations du modèle relationnel appelées relations floues généralisées pour permettre le stockage des informations imprécises avec un degré de compatibilité lié à chaque valeur d'attribut. Ils ont également défini une algèbre appelée algèbre relationnelle floue généralisée, Nous présenterons dans ce qui suit les concepts de base de ce modèle.

▪ Modèle GEFRED de Médina et al.

Le modèle GEFRED (GEneralised model for Fuzzy RElationnel Database) a été proposé en 1994 par Medina et al. [MED 94a] et étendu par Galindo et al. [GAL 99, 00, 05, 06][GTB 09]. Il constitue une synthèse éclectique des différents modèles publiés pour traiter le problème de représentation et d'interrogation des informations floues au moyen des BDRs. 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 [MAB 05]. 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.

Définition 2.1

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, INDEFINI et NULL (type 8, 9 et 10 de la table 2.2). 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 2.2 [MAB 05].

Définition 2.2

Une Relation Floue Généralisée, R, est donnée 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" où le dernier est optionnel,

$$H = \{(A_1 : D_1[,C_1]), (A_2 : D_2[,C_2]), ..., (A_n : D_n[,C_n])\} \quad \textbf{(2.8)}$$

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].

- Le Corps comporte un ensemble de tuples, appelés "tuples flous généralisés", où chaque tuple est composé d'un ensemble de triple "attribut-valeur-degré" où le degré est optionnel,

$$B = \{A_1 : \tilde{d}_{i1}[,c_{i1}]), A2:\tilde{d}_{i2}[,c_{i2}]),...,(An:\tilde{d}_{in}[,c_{in}]) \} \quad \textbf{(2.9)}$$

Avec i = 1,2,..., m, et m est le nombre de tuples dans la relation, d_{ij} représente la valeur du domaine pour le tuple i et l'attribut A_j, c_{ij} est le degré de la compatibilité associé à cette valeur.
Le degré de compatibilité est utilisé pour savoir le degré avec lequel une valeur d'un attribut a satisfait la condition de la requête.

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}.

Tableau 2.2 : Types de données dans GEFRED

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 [MAB 05].

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 étendus de sorte que chacun d'eux opèrent sur une (des) relation(s) graduelle(s) et retourne une telle relation.

L'algèbre relationnelle étendue a été introduite par Bosc pour étendre l'algèbre relationnelle classique [BOS 98,04]. 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 cartésien : $\mu_{R \times S}(xy) = max(\mu_R(x), 1 - \mu_S(y))$

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

- **Sélection** : $\mu_{\sigma\phi(R)}(x) = min(\mu_R(x), \mu_\phi(x))$ 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** : $\mu_{RA\theta BS}(xy) = min(\mu_R(x), \mu_S(y), \mu_\theta(x.A, y.B)))$ où A (resp. B) est un sous-ensemble de X (resp. Y), a et b sont compatibles (c-à-d 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 [BOS 98,05a-b-c,10b], parmi ces définitions :

- 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 :

$x \in R_{A \div B} S \Leftrightarrow [x \in \Pi_x(R) \text{ et } (\forall a, a \in \Pi_y(S) \Rightarrow (a, x) \in R)]$

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 implication floue et interpréter le quantificateur universel comme une conjonction généralisée, soit :

$\mu R_{A \div B} S(x) = min(\mu_{\Pi x(support(R))}(x), mins\ \mu_S(s) \rightarrow \mu_R(s.B, x))$

ou encore $\forall x \in \Pi x(support(R)),\ \mu R_{A \div B} S(x) = mins\ \ \mu_S(s) \rightarrow \mu_R(s.B, x))$

- Une autre forme 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∩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.

2.4.2 Algèbre Relationnelle Généralisée

Le modèle GEFRED décrit une Algèbre Relationnelle Floue Généralisée pour manipuler les relations floues généralisées [MED 94a] [GAL 06]. Les opérations classiques, Union, Intersection, Différence, Produit Cartésien, Projection, Jointure et Sélection, sont étendues pour leur permettre d'opérer d'une manière cohérente [MAB 05].

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 2.3

Soit U le domaine du discours considéré, le Comparateur Étendu, θ, se défini comme toute relation floue sur U qui peut être exprimée selon la façon suivante :

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

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

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 [GTB 06].

Avec cette définition, le Comparateur Étendu, θ, nous permet de modéliser, dans un chemin logique, les opérateurs de la comparaison suivants :

- Comparateurs classiques de l'Algèbre Relationnelle tel que : $=,=, >, \geq, <, \leq$.

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 $d_i = d_j$.

- Comparateurs flous tel que "approximativement égal", "beaucoup plus grand que", Ce genre de comparateur sera exprimé par des fonctions d'appartenances.

Par exemple, l'opérateur "approximativement égal" pourrait être modélisé par le comparateur étendu correspondant $\cong$ avec la fonction d'appartenance suivante : $\mu : (d_i, d_j) = e^{-\beta|di-dj|}$ *avec* $\beta > 0$.

- Comparateurs de Similitude qui opèrent sur des données scalaires avec une relation de similitude établie entre eux.

Définition 2.4

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 :

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

$$\Theta^{\theta}\ (d_1, d_2) \rightarrow [0, 1]$$

Θ^{θ} est dit Comparateur Flou Généralisé sur D abouti par le comparateur étendu, s'il vérifie :

$$\Theta^{\theta}(\ d_1', d_2') = \theta_e(d_1, d_2)\ \forall d_1, d_2\ \in U.$$

Où d_1', d_2' représentent les distributions de possibilité ,1/d1, 1/d2, induits respectivement par les valeurs d_1, d_2.

2.5 Langages de manipulation des bases de données floues

La manipulation de toute base de donnée nécessite obligatoirement de définir un langage permettant, à partir des commandes qu'il offre, de répondre à toutes les informations demandées. Les principaux langages de manipulation des bases de données floues sont : SQLf et FSQL [MAB 05].

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, 04, 10, 11]. Le langage SQLF fait l'objet de plusieurs travaux d'extension par l'équipe de recherche IRISA/Badins (Bases de Données et Interrogation Souple) dans le projet iSQLF [BAS 06] [BAG 07].

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 autorisées. 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. Pour la clause group by, SQLf introduit une version floue de cette clause qui a été notamment décrite dans [BOS 11].

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 le 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_{CR} *and* f_{CS} *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_{CR} (resp. f_{CS}) étant une expression de sélection sur R (resp. S). Un couple de valeur (a,b) de la relation résultante R_f a le degré :

$$\mu_{Rf}(a, b) = max_{r \in R \text{ et } r.A=a \text{ et } s \in S \text{ et } s.B=s} min(\mu_{fCR}(r), \mu_{fCS}(s), \mu_{\theta}(r.C, s.D))$$

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 : $\mu_{AinSR}(a) = \max_{b \in support(SR) \wedge b=a} \mu_{SR}(b)$ où SR est l'ensemble flou retourné par la sous requête. La non-appartenance est définie par :

$\mu_{A \text{ not in } SR}(a) = 1 - \mu_{A \text{ in } SR}(a) = 1 - \max_{b \in support(SR) \wedge b=a} \mu_{SR}(b)$

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 (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 :

$$\mu_{exists}(SR) = max_{x \in support(SR)} \mu_{SR}(x).$$

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 [MAB 05].

Expression de la division étendue

L'expression de la division dans SQLf est exprimée comme suit :

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.

Une extension de la division relationnelle, appelée division approchée, correspond au cas où le quantificateur universel sous-jacent à la division est lui-même relaxé. Cette relaxation peut par exemple passer par l' utilisation d' un quantificateur linguistique comme "la plupart" [BOS 05a-b-c].

2.5.2 FSQL

Le langage FSQL [MED 94b][GAL 07] 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 [MAB 05].

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 BDD. Le langage FSQL incorpore également des nouveautés pour autoriser le traitement des informations imprécises. 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 cet attribut, précédées par le symbole $ pour les distinguer facilement. Nous pouvons distinguer deux types d'étiquettes :

1. Etiquettes 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 $\alpha = 16$, $\beta = 22$, $\gamma = 28$ et $\delta = 32$ (en nombre d'année).

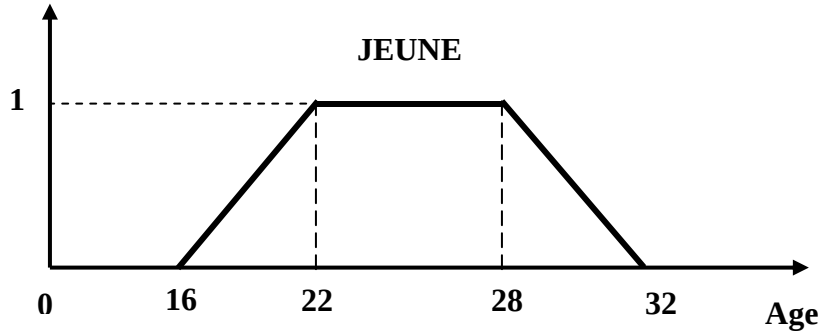


Figure 2.1 : Exemple d'étiquette linguistique pour l'attribut Âge

2. Étiquettes pour des attributs dans un domaine non ordonné (type 1,3 et 5 de la table 2.2). 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.), FSQL inclut les comparateurs flous présentés dans le Tableau 2.3. 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 de possibilité mesurent à quelle valeur (degré) c'est possible que la condition soit exécutée.

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)
MGI	NMGT	Beaucoup Plus Grand que (Much Greater Than)
MLI	NMLT	Beaucoup Plus Petit que (Much Greater Than)

Tableau 2.3 : Les comparateurs flous de FSQL

- **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 $\geq$.

En plus du seuil de réalisation, il ya des extensions du FSQL, comme PFSQL [TAL 08][SKT 08], qui utilise d'autre paramètres comme le poids et les priorités entre les conditions. Les poids peuvent être de deux types : statique ou dynamique [GAL 08b]. Ce qui permet plus de personnalisation dans la formulation de la requête et un meilleur raffinage des résultats retournés.

- **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 le tableau 2.4, bien que l'utilisateur puisse changer facilement cette fonction.

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

Tableau 2.4 : La fonction CDEG de FSQL

- **Les constantes floues**: FSQL a introduit de nouvelles constantes basées sur le concept flou comme il est montré dans le tableau 2.5.

Constantes floues	**Signification**
UNKNOWN	Valeur inconnue mais l'attribut est applicable (type 8 de la table 2.2)
UNDEFINED	L'attribut n'est pas applicable ou n'a pas de sens (type 9 de la table 2.2)
NULL	Une ignorance total (type 10 de la table 2.2)
$[a,b,c,d]	Trapèze flou (a≤b≤c≤d)
$label	Etiquette linguistique : trapézoïdale ou scalaire
$[n,m]	Intervalle "entre n et m" (a = b = n et c = d = m)
#n	Valeur floue "Approximative" (b = c = n et n-a = d-n = marge)

Tableau 2.5 : 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 2.5.

<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 SGBD.

- 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[ρ] (condition_floue) THOLDτ

$Quantificateur FUZZY [ρ] (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 soient pas tenus dans l'analyse et dans l'exécution.

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.

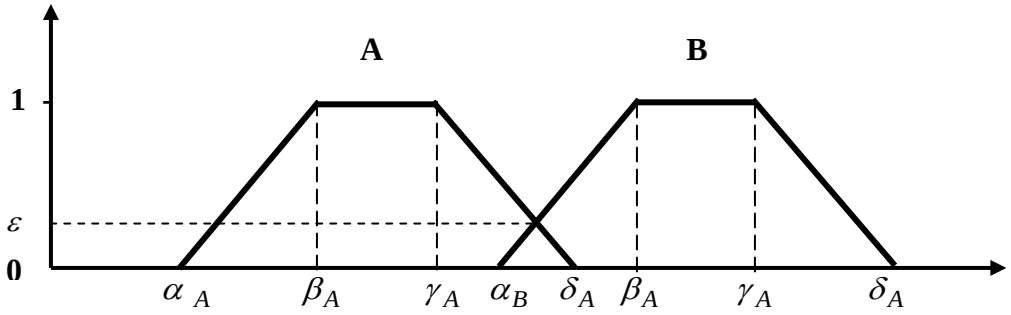


Figure 2.2 : Comparaison de deux distributions de possibilités trapézoïdales

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 2.6.

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 requêtes sur le nombre total de tuples résultant d'une certaine consultation, en disant si ce nombre est "grand", "petit", "beaucoup", "un peu", etc. Dans ce cas, le quantificateur dépend d'une seule quantité.

Les quantificateurs relatifs peuvent résoudre des requêtes 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", etc.

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 :

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

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

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 , 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".

Dans [GAL 06], Les quantificateurs flous (absolu ou relatif) peuvent être écrits avec un (ou deux) seuil(s) de réalisation.

- Les commandes de FSQL

Le langage FSQL étend déjà les commandes existantes dans SQL, mais il incorpore aussi d'autres nouvelles commandes. Le langage FSQL distingue dans la définition des données trois types d'attributs. Cette classification est inspirée du modèle GEFRED.

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 1a 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 1a distribution A.*
FDIF/NFDW	*1 CDEG(A NFEQ B)*	*1 CDEG(A FEQ B)*
FGT/NFGT	$\begin{cases} 1 & si\ \gamma_A \geq \delta_B \\ \dfrac{\delta_A - \gamma_B}{(\delta_B - \gamma_B) - (\gamma_A - \delta_A)} & si\ \gamma_A \prec \delta_B\ et\ \delta_A \succ \gamma_B \\ 0 & \text{sinon} \end{cases}$	$\begin{cases} 1 & si\ \alpha_A \geq \delta_B \\ \dfrac{\beta_A - \gamma_B}{(\delta_B - \gamma_B) - (\alpha_A - \beta_A)} & si\ \alpha_A \prec \delta_B\ et\ \beta_A \succ \gamma_B \\ 0 & \text{sinon} \end{cases}$
FGEQ/NFGEQ	$\begin{cases} 1 & si\ \gamma_A \geq \delta_B \\ \dfrac{\delta_A - \alpha_B}{(\beta_B - \alpha_B) - (\gamma_A - \delta_A)} & si\ \gamma_A \prec \delta_B\ et\ \delta_A \succ \alpha_B \\ 0 & \text{sinon} \end{cases}$	$\begin{cases} 1 & si\ \alpha_A \geq \beta_B \\ \dfrac{\beta_A - \alpha_B}{(\beta_B - \alpha_B) - (\alpha_A - \beta_A)} & si\ \alpha_A \prec \beta_B\ et\ \beta_A \succ \alpha_B \\ 0 & \text{sinon} \end{cases}$
FLT / NFLT	$\begin{cases} 1 & si\ \beta_A \leq \alpha_B \\ \dfrac{\alpha_A - \beta_B}{(\alpha_B - \beta_B) - (\beta_A - \alpha_A)} & si\ \beta_A \succ \alpha_B\ et\ \alpha_A \prec \beta_B \\ 0 & \text{sinon} \end{cases}$	$\begin{cases} 1 & si\ \delta_A \leq \alpha_B \\ \dfrac{\gamma_A - \beta_B}{(\alpha_B - \beta_B) - (\delta_A - \gamma_A)} & si\ \delta_A \succ \alpha_B\ et\ \gamma_A \prec \beta_B \\ 0 & \text{sinon} \end{cases}$
FLEQ/NFLEQ	$\begin{cases} 1 & si\ \beta_A \leq \gamma_B \\ \dfrac{\delta_B - \alpha_A}{(\beta_A - \alpha_A) - (\gamma_B - \delta_B)} & si\ \beta_A \succ \gamma_B\ et\ \alpha_A \prec \delta_B \\ 0 & \text{sinon} \end{cases}$	$\begin{cases} 1 & si\ \alpha_A \leq \gamma_B \\ \dfrac{\gamma_A - \delta_B}{(\gamma_B - \delta_B) - (\delta_A - \gamma_A)} & si\ \delta_A \succ \gamma_B\ et\ \gamma_A \prec \delta_B \\ 0 & \text{sinon} \end{cases}$
MGT/NMGT	$\begin{cases} 1 & si\ \gamma_A \geq \delta_B + M \\ \dfrac{\gamma_B + M - \delta_A}{(\beta_A - \alpha_A) - (\gamma_B - \delta_B)} & si\ \gamma_A \prec \gamma_B + M\ et\ \delta_A \succ \gamma_B + M \\ 0 & \text{sinon} \end{cases}$	$\begin{cases} 1 & si\ \gamma_A \geq \delta_B + M \\ \dfrac{\gamma_B + M - \delta_A}{(\alpha_A - \beta_A) - (\delta_B - \gamma_B)} & si\ \alpha_A \prec \gamma_B + M\ et\ \beta_A \succ \gamma_B + M \\ 0 & \text{sinon} \end{cases}$
MLT /NMLT	$\begin{cases} 1 & si\ \beta_A \leq \alpha_B - M \\ \dfrac{\beta_B - M - \alpha_A}{(\beta_A - \alpha_A) - (\alpha_B - \beta_B)} & si\ \beta_A \succ \alpha_B - M\ et\ \alpha_A \prec \beta_B - M \\ 0 & \text{sinon} \end{cases}$	$\begin{cases} 1 & si\ \delta_A \leq \alpha_B - M \\ \dfrac{\beta_B - M - \gamma_A}{(\delta_A - \gamma_A) - (\alpha_B - \beta_B)} & si\ \delta_A \succ \alpha_B - M\ et\ \gamma_A \prec \beta_B - M \\ 0 & \text{sinon} \end{cases}$

Tableau 2.6 : Définition des comparateurs flous dans la famille de Possibilité/Nécessité

Les principales commandes LDD de FSQL sont :

CREATE TABLE : C'est la phrase qui inclut plus de nouvelles :

1. Type de Données Floues : Il peut être l'un des 4 attributs flous type1 (FTYPE1 ou CRISP), type2 (FTYPE2 ou POSSIBILISTIC) ,type3 (FTYPE3 ou SCALAR) et type4 (FTYPE4 ou SCALAR)
2. Sous consultation, constantes, conditions et expressions floues.
3. Restrictions de colonne : Dans cette partie, plusieurs restrictions peuvent être imposées, 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).

CREATE NEARNESS : C'est une nouvelle commande, spécifique de FSQL, qui sert à créer des étiquettes pour les attributs flous Type3. 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 ;

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 définies 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.

2.6 Conclusion

Nous avons présenté dans ce chapitre les différentes approches qui visent à utiliser les techniques de la logique floue (sous-ensembles flous, distributions de possibilités) pour pallier le problème de représentation et de manipulation des données imprécises dans les bases de données classiques.

La représentation des données sous forme d'ensembles flous permet de coder l'information de façon plus complète et plus fidèle. Au niveau du langage de manipulation des bases de données, SQL a été étendu à FSQL et SQLf selon l'algèbre utilisée en adjoignant des concepts flous. Les requêtes sont ainsi plus flexibles et plus faciles à exprimer.

Nous allons se baser dans ce qui suit, sur le modèle GEFRED et également sur le langage FSQL pour présenter les extensions nécessaires pour permettre la représentation des données imparfaites dans un SGBD relationnel.

Chapitre 3

Représentation de l'information floue dans les bases de données relationnelles

3.1 Introduction

L'information floue peut être intégrée dans les SGBDRs à deux niveaux : Le premier niveau considère la possibilité de faire des requêtes floues aux bases de données classiques. La seconde est liée au problème d'ajout de l'information floue au système. Dans les deux cas, la logique floue fournit un outil puissant pour représenter l'information [MED 95b]. Le traitement du problème au deuxième niveau donne lieu aux modèles de base de données relationnelle floue dont GEFRED constitue le modèle le plus général, les autres modèles peuvent être considérés comme des cas particuliers.

Le modèle GEFRED a été proposé dans le sens d'établir un prototype de SGBDR flou qui met en application un modèle flou concret de base de données relationnelle. En plus il inclut la capacité de représentation et de traitement de l'information floue dans un SGBDR classique. Ce modèle incorpore de nouvelles définitions dans la structure et le traitement de données , ce qui permet d'intégrer, dans le même cadre, les modèles relationnels flous précédents [MED 95b].

Dans ce chapitre nous présentons les différents aspects liés au modèle théorique utilisé ainsi que les structures de données et les traitements nécessaires pour la représentation et le traitement des données floues dans les SGBDRs.

3.2. Le modèle théorique utilisé

Les caractéristiques principales du modèle utilisé sont les suivantes [MED 95b] :

- Traitement de l'information imprécise de différents types.

- Une organisation différente de l'information. La même structure de relation est utilisée pour représenter l'information initiale, l'information résultante des opérations algébriques et les résultats finaux.
- Un certain contrôle peut être fait sur la précision avec laquelle n'importe quelle condition simple impliquée dans une requête est satisfaite.

3.2.1. Structure de données

L'information traitée par le modèle est organisée comme suit :

• Le domaine DG qui englobe tous les attributs d'une relation contenant des données du tableau 2.2

• Les données sont structurées par un modèle de relation, R_{FG}, donné par :

$$R_{FG} \in (D_{G1}, C_1) \times \ldots \times (D_{GN}, C_n),$$

où chaque D_{Gj} est un domaine du type indiqué précédemment, C_j est une "attribut de compatibilité" qui prend ses valeurs dans [0, 1]. Chaque attribut est associé à un attribut de compatibilité. Dans les relations de base, l'attribut de compatibilité n'apparaît pas. Cette relation représente l'information initiale aussi bien que celle qui résulte des opérations de l'algèbre floue.

La manipulation de ces relations par l'algèbre relationnelle floue peut modifier, pour chaque tuple, les valeurs d'attribut de compatibilité.

3.2.2 Traitement des données

L'algèbre floue utilisée dans ce modèle est une extension de l'algèbre classique ; dans cette extension les opérateurs de comparaison spécifiques sont utilisés afin de traiter l'information floue. La requête floue reçoit un traitement spécial, basé sur les points suivants [MED 95b] :

- On appelle *une sélection atomique*, une requête sur une relation type R_{rG}, dans laquelle on recherche la satisfaction d'une simple condition.
- Quand un attribut, un opérateur et un constant flou sont impliqués dans une sélection atomique, une telle condition sera satisfaite à un degré pour chaque valeur d'attribut. Un tel degré prend une valeur dans [0, 1]

- Dans une sélection atomique, nous pouvons établir un seuil pour le degré de satisfaction d'une condition. Grâce à ce seuil, nous pouvons éliminer les tuples qui ne remplissent pas la condition à un degré supérieur ou égal au seuil.
- Le résultat d'une sélection atomique avec un seuil pour le degré de satisfaction est, de nouveau, une relation, dans laquelle le degré de satisfaction d'une condition pour chaque valeur de l'attribut impliqué apparaît dans l'attribut de compatibilité.

Les conditions composées sont ceux obtenues en combinant des conditions simples avec des connecteurs de la logique (inversion, conjonction et disjonction).
Les conditions composées sont résolues comme suit :
- À partir de chaque condition simple nous obtenons la relation résultante en appliquant une sélection atomique avec un seuil donné.
- Pour les conditions simples liées avec l'opérateur conjonctif, nous faisons l'intersection des relations obtenues à partir de chaque condition. Après, les valeurs des attributs de compatibilité associés à chaque attribut impliqué dans les conditions simples sont calculées.
Ce calcul consiste à donner à l'attribut de compatibilité de chaque tuple d'intersection une valeur qui est égale au minimum de ceux dans les conditions simples initiales.
- Pour des conditions simples liées à l'opérateur disjonctif, on fait l'union des relations obtenues pour chaque condition et on met à jour l'attribut de compatibilité avec la valeur maximum.
- Pour un état simple réalisant une inversion, on met à jour la valeur d'attribut de compatibilité avec le complément à 1 de la valeur actuelle dans chaque tuple.

3.3 Représentation de l'information floue

Les éléments liés au traitement des données floues peuvent avoir différentes représentations. Une distribution normalisée de possibilité, par exemple, peut être représentée par différents types de fonctions, mais nous utiliserons une représentation trapézoïdale pour notre cas. La même chose pour la modélisation des opérateurs flous ainsi que pour le reste des éléments flous utilisés dans le système. Les outils de représentation présentés ne sont pas exclusifs pour une représentation concrète mais représente la base sur laquelle le système est établi selon le modèle conçu pour un

SGBDRF. Par conséquent, elle constitue une étape entre la formulation d'un modèle de gestion de base de données floue et l'implémentation efficace d'un système basé sur ce dernier [MED 95b].
D'une manière générale, Les données peuvent être soit précises ou imprécises, pour les données précises, la représentation fournie par le serveur SGBDR sera utilisée. Par contre, les données imprécises nécessitent la création des nouveaux types de données.

3.3.1. Types de données imprécises

Le modèle considère deux groupes différents avec deux représentations différentes pour les données imprécises [MED 95b] :

1- Données imprécises dans un domaine ordonné : Ce groupe de données contient des distributions de possibilité définies sur des domaines continus ou discrets mais ordonnés. Chaque donnée de ce type est associée à une fonction d'appartenance. Pour la simplicité dans la représentation et l'efficacité du calcul, la représentation présentée dans le tableau 3.1 sera adoptée. 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 [α, β, γ, δ], comme le montre le tableau 3.1.

-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 comme le montre le tableau 3.1.

-Valeurs approximatives : Elles représentent le concept imprécis "*approximativement n*" au moyen d'une valeur, appelée "*marge*". Elles utilisent la distribution de possibilité triangulaire du tableau 3.1 comme fonction d'appartenance.

-Intervalles de possibilité : Ce sont des cas particuliers des distributions de possibilité trapézoïdales dans lesquelles les inclinaisons des deux côtés du trapèze sont infinies et par conséquent toutes les valeurs entre les deux extrémités sont les seules qui sont complètement possibles (possibilité 1), comme c'est montré dans le tableau 3.1.

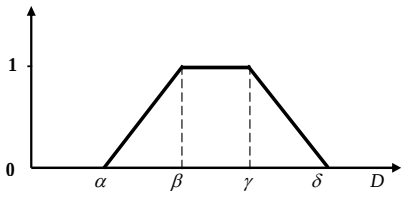
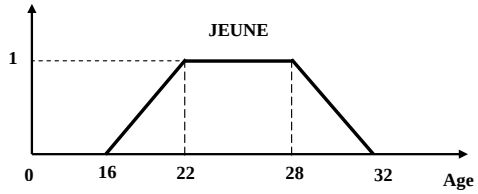
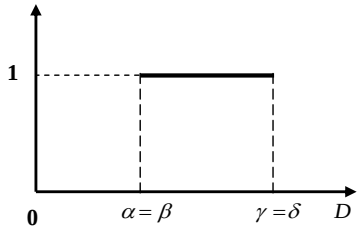
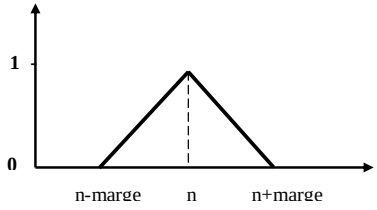
Type de donnée	Représentation
Distribution de possibilité trapézoïdale	
Etiquette linguistique	
Intervalle	
Valeur approximative	

Tableau 3.1 : Représentation des données imprécises sur un domaine ordonné

2- Données imprécises dans un référentiel non ordonné : ce groupe de données est établi sur un domaine discret sur lequel il ya des " relations de proximité " définies entre ses valeurs. Dans ce cas, nous devrons stocker la représentation des données aussi bien que la représentation des relations de proximité définies sur les valeurs de domaine.

Les différentes données que nous pouvons représenter dans ce groupe sont les suivants :

-Scalaires simples. Ces données sont représentées en utilisant le mode de représentation du serveur SGBDR. Nous devons seulement fournir au système l'information nécessaire pour les traiter avec la relation de proximité définie sur le domaine fondamental.

-Distribution de possibilité sur un domaine discret : des données imprécises de ce type sont associées à une représentation dans laquelle les valeurs de domaine qui la constituent sont décrites ensemble avec les valeurs respectives de possibilité pour chacune d'elles. $((P_1, D_l),\ldots, (P_n, D_n,))$.

3- Type de données « UNKNOWN ». Les données de ce type expriment l'ignorance concernant la valeur qu'un attribut puisse prendre, on sait, en fait, qu'il est possible que l'attribut prenne une des valeurs de domaine. Par conséquent, nous représentons le type INCONNU par la distribution de possibilité $\{1/u : u \in U\}$ où U est le domaine fondamental. La figure 3.1 montre cette distribution de possibilité.

4- Type de données « UNDEFINED » Quand un attribut prend la valeur UNDEFINED, il reflète le fait qu'aucune des valeurs de son domaine n'est permise. Ceci signifie qu'aucune des valeurs n'est possible. Par conséquent, la distribution de possibilité associée est $\{0/u : u \in U\}$, où U est le domaine fondamental. La figure 3.1 montre cette distribution de possibilité.

5- Type de données « NULL ». Quand un attribut prend la valeur NULL, cela signifie qu'on n'a aucune information, soit parce qu'elle est inconnue (UNKNOWN) ou parce qu'aucune valeur du domaine n'est possible (INDEFINED). La distribution de possibilité dans ce cas est {1/UNKNOWN, 1/UNDEFINED}.

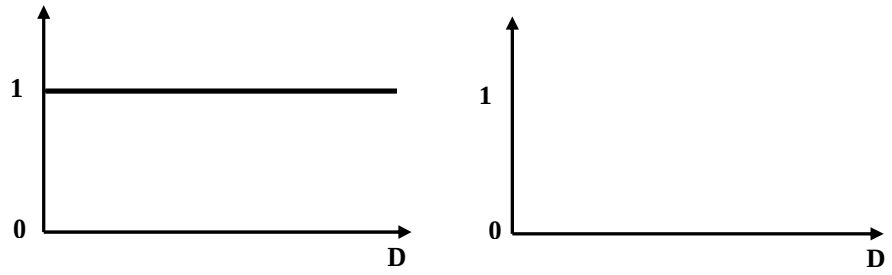


Figure 3.1 : Représentation des types UNKNOWN et UNDEFINED

3.3.2 Opérateurs relationnels floues:

Les différents opérateurs de comparaison utilisés pour associer des relations de base de données sont les opérateurs relationnels. Pour traiter l'information imprécise, ces opérateurs ont été étendus. La représentation adoptée dans ce modèle pour les différents opérateurs de relation est comme suit [MED 95b] :

- **Égal à** : Cet opérateur modélise le concept d'égalité pour des données imprécises. Formellement, il peut être exprimé par la fonction d'appartenance donné par :

$$\mu_{\text{equal}_{\grave{a}}}(d,d') = \sup_{(d,d')\in DxD} \min\left(p(d,d'),\Pi(\text{d}),\Pi(\text{d}')\right)$$

où p (d, d') est une relation de proximité.

$\Pi(\text{d}),\Pi(\text{d}')$ Sont les distributions de possibilité respectives définies sur le domaine de discours D.

- Pour les données imprécises définies sur un domaine ordonné, $\text{p}(\text{d},\text{d}') = \delta(\text{d},\text{d}')$ où δ est un delta de Diracs. Tenant compte de la représentation utilisée pour ce type de données, le résultat de l'opération *égal à* peut être obtenue géométriquement (figure 3.2).
- Pour des données avec similitude sur un domaine discret, p (d, d') est la représentation matricielle de la « relation de proximité » qui est définie sur le domaine de discours D.

- **Approximativement égal** : Cet opérateur fournit le degré avec lequel deux valeurs numériques exactes sont approximativement égaux. On le calcule selon l'expression suivante :

$$\mu_{approx\text{-}egal}(x,y) = \begin{cases} 0 & si \quad |\text{x - y}| > \text{marge} \\ 1 - |\text{x - y}| / \text{marge} & si \quad |\text{x - y}| \le \text{marge} \end{cases}$$

La figure 3.2 montre comment on le calcule. Le paramètre marge correspond à l'opérateur du domaine sur lequel il est défini.

- **Supérieur ou égal** : Il est défini sur un des domaines ordonnés. Sa fonction d'appartenance est donnée par la relation floue :

$$p_{\geq}(A,B) = \sup_{x,y \in X \times Y} \min(\geq(x,y), \pi_A(x), \pi_B(y)),$$

où A et B sont des données imprécises sur un domaine ordonné ou des données numériques "exactes", $\pi_A(x), \pi_B(y)$ leur représentations de possibilité respectives et $\geq$ l'opérateur supérieur ou égal classique donné par : $\geq(x,y) = \begin{cases} 0 & si \quad x < y \\ 1 & si \quad x \geq y \end{cases}$

Cet opérateur peut résoudre les comparaisons suivantes :

- Degré auquel un nombre " exacte " est " supérieur ou égal " à une distribution de possibilité.
- Degré auquel une distribution de possibilité est " supérieure ou égale " à un nombre "exacte ".
- Degré auquel une distribution de possibilité est " supérieure ou égale " à une autre distribution de possibilité.

- **Inférieur ou égal** : Il est défini sur des domaines ordonnés. La fonction d'appartenance de cet opérateur est donnée par la relation floue

$$p_{\leq}(A,B) = \sup_{x,y \in X \times Y} \min(\leq(x,y), \pi_A(x), \pi_B(y)),$$

où A et B sont des données imprécises sur un domaine ordonné ou des données numériques " exactes ", $\pi_A(x), \pi_B(y)$ leur représentations de possibilité respectives et $\leq$ l'opérateur inférieur ou égal classique donné par : $\leq(x,y) = \begin{cases} 0 & si \quad x \succ y \\ 1 & si \quad x \leq y \end{cases}$

Cet opérateur peut résoudre la même comparaison que l'opérateur " sup ou égal ".

- **Supérieur à :** cet opérateur est défini à partir de l'opérateur " inférieur ou égal " en calculant son complément :

$$\mu_{\succ(A,B)} = 1 - \mu_{\leq(A,B)}$$

- I**nférieur à** : cet opérateur est défini à partir de l'opérateur "supérieur ou égal" en calculant son complément :

$$\mu_{\prec(A,B)} = 1 - \mu_{\geq(A,B)}$$

La figure 3.3 montre graphiquement les opérateurs relationnels flous décrits.

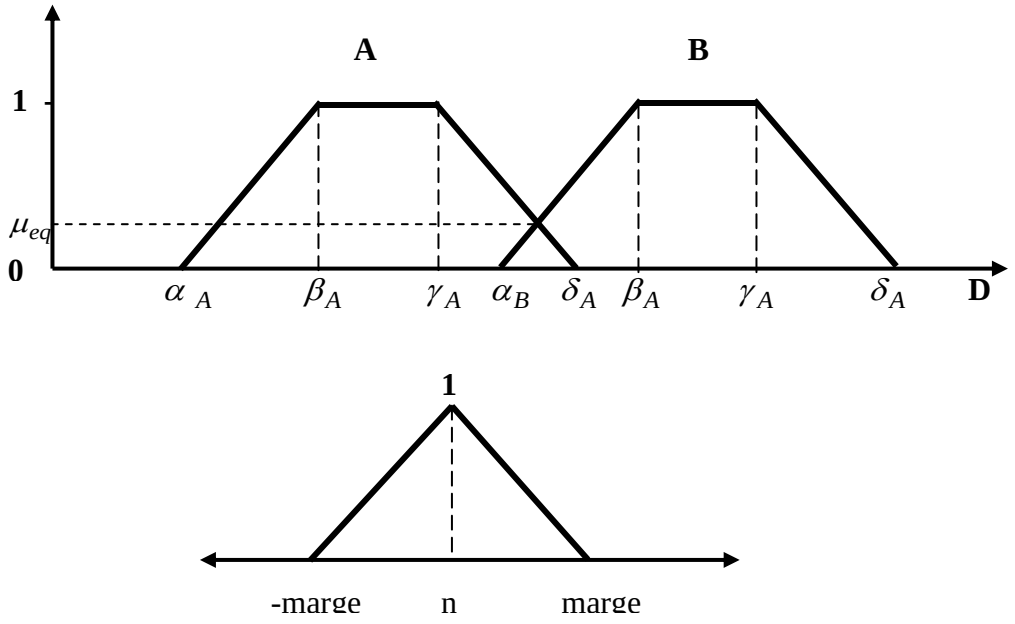


Figure 3.2 Opérateurs égal et approximativement égal

3.3.3 Qualificateurs de seuil de requête

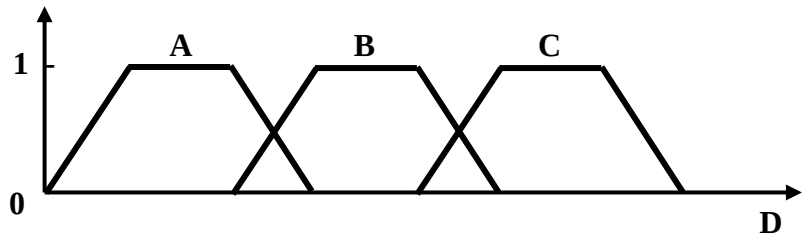
Quand nous interrogeons une base de données floue nous imposons quelques conditions que les tuples résultants doivent satisfaire. Etant donné la nature imprécise des données et des opérateurs sur lesquels nous travaillons, il ya un degré de satisfaction pour chaque condition impliquée dans une requête. Ce degré de satisfaction est dans l'intervalle [0, 1]. En utilisant un seuil minimal pour ce degré nous pouvons contrôler la précision avec laquelle les conditions de la requête sont satisfaisantes.

La qualification de condition est l'action d'établir un seuil de satisfaction pour une condition simple atomique impliquée dans une requête .Un tel seuil sera nommé "qualificateur ". Ce qualificateur est une valeur entre 0 et 1, elle peut être représentée par une valeur linguistique ; par exemple, si nous déclarons que le degré auquel une condition est satisfaite est haut, il signifie que nous acceptons tous les tuples dont le degré de satisfaction est supérieur ou égal à 0.8. C'est-à-dire, nous pouvons associer des valeurs linguistiques avec des qualificateurs. Les valeurs seuils associées à chaque étiquette linguistique doivent être stockées dans le système et possèdent, comme les étiquettes linguistiques, des significations subjectives [MED 95b].

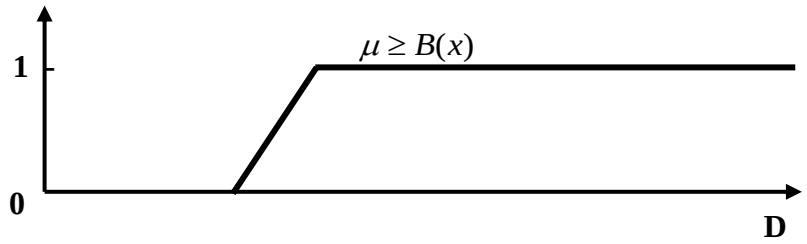
3.3.4 Quantificateurs flous d'une requête

Il y a deux Quantificateurs dans le modèle relationnel classique : Le quantificateur EXISTE ($\exists$) et le quantificateur universel FORALL ($\forall$). Avec le premier nous obtenons une réponse vraie quand une partie des tuples remplissent la condition dans la requête, et avec le second la réponse vraie est obtenue quand tous les tuples dans la base de données remplissent une telle condition. Mais dans le cas flou il y a un large éventail de quantificateurs entre le ($\exists$) et le ($\forall$) précédents qui peuvent être donnés sous forme linguistique comme : "presque aucun", "certains", "beaucoup", "presque tous". Ces valeurs linguistiques ont une représentation en termes de distributions de possibilité sur un domaine D_Q défini comme [MED 95b]:

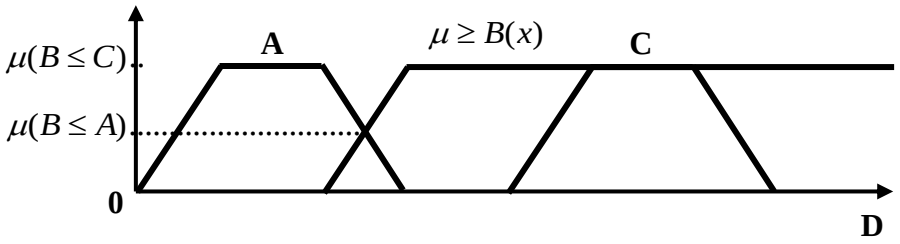
$$D_Q = \left\{ d : d = \frac{\text{nombre de tuples satisfaisant la condition}}{\text{nombre de tuples consultés}} \right\}$$



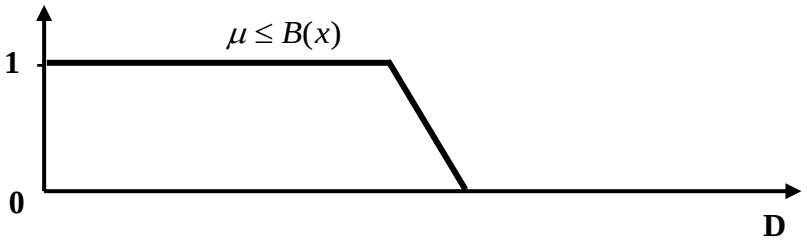
Les trois distributions de possibilité à traiter



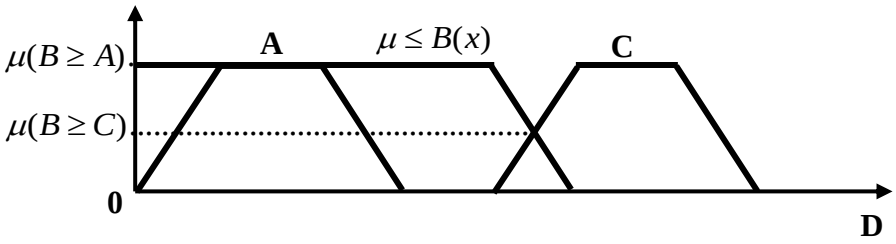
La fonction d'appartenance de l'opérateur $\geq$ appliqué à B



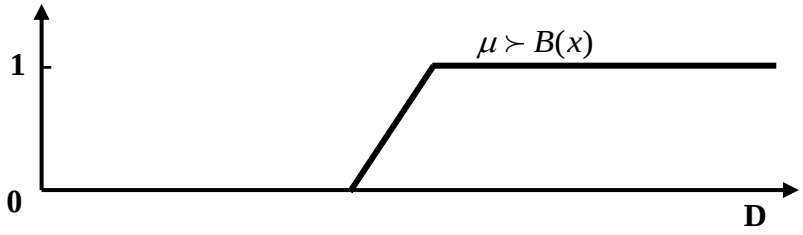
Les degrés avec lesquels A et C sont " supérieurs ou égaux " à B



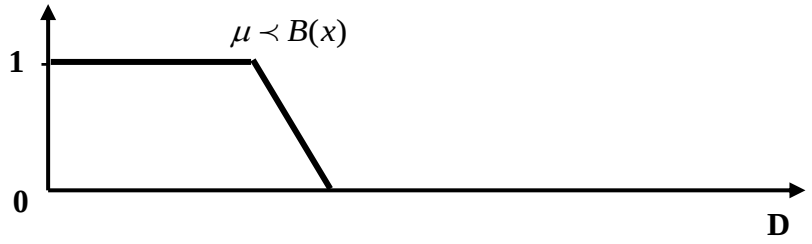
La fonction d'appartenance de $\le$ appliquée à B



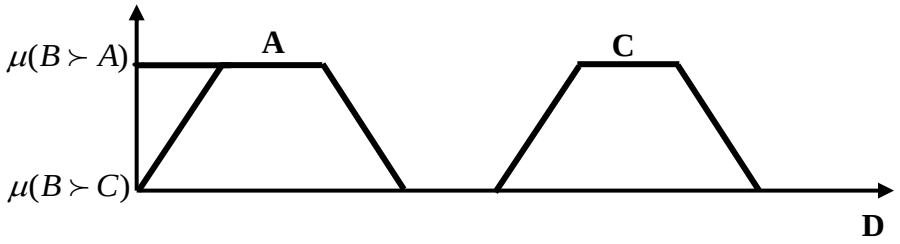
Degrés avec lesquels A et C sont " inférieurs ou égaux " à B



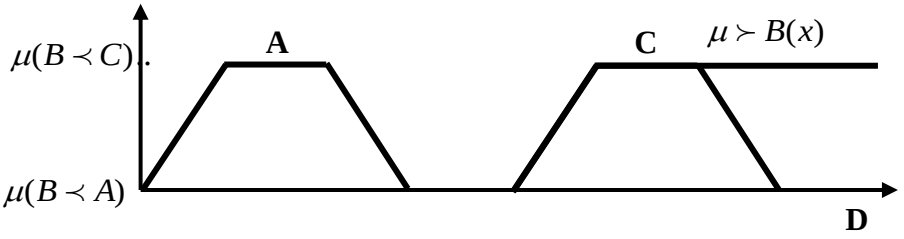
La fonction d'appartenance de l'opérateur > appliqué à B



La fonction d'appartenance de l'opérateur < appliqué à B



Les degrés avec lesquels A et C sont " inférieurs " à B



Les degrés avec lequel A et C sont " supérieurs " à B

Figure 3.3 opérateurs relationnel flous

3.4 L'Architecture FIRST

Afin de pouvoir représenter et manipuler des données imprécises dans un SGBDR classique, l'architecture FIRST (A Fuzzy Interface for Relational Systems) a été proposée par Medina et al. [MED 95a] qui a été ensuite étendue vers FIRST2 [GAL 05], Elle étend la structure existante et ajoute de nouveaux composants pour traiter l'information floue en utilisant GEFRED comme modèle théorique et les ressources du modèle relationnel classique pour représenter ce type d'information. Afin de permettre la création et la description des bases de données floues avec le langage FSQL, L'architecture FIRST a été étendue avec une couche FSQL_TO_SQL [GTB 06,09].

La figure 3.4 montre le schéma général de FIRST. Ses principales composantes sont [MAB 05]:

-SGBDR : toutes les opérations conçues pour l'extension floue se traduisent par une demande au SGBD host (en général en SQL). Ces demandes se réalisent en employant le langage SQL ou FSQL selon le type des conditions et des attributs. 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.

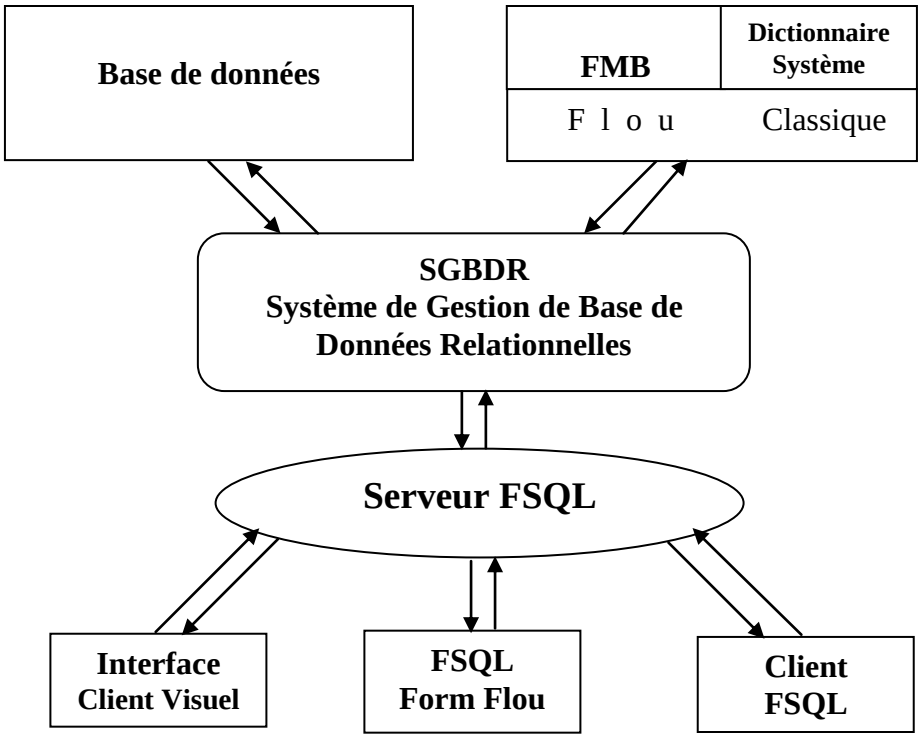


Figure 3.4 Schéma général de FIRST

- **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. 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.

L'architecture FIRST se compose d'un ensemble de composants qui interagissent entre eux afin d'assurer une bonne représentation des donnés floues avec la possibilité d'un traitement simple et efficace de ces données. 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 base de données floue. Ce canal de communication permet aux utilisateurs, soit directement ou à travers des applications, d'extraire et d'entreposer leurs données imprécises.

Afin de modéliser des attributs flous, nous distinguons entre deux classes des attributs flous: les attributs flous dont les valeurs sont des ensembles flous et les attributs flous dont les valeurs sont des degrés flous. Chaque classe inclut quelques différents types de données flous [GAL 06].

Attributs flous représentants des valeurs flous

Ces attributs flous peuvent être classifiés en quatre types de données. Cette classification est basée sur le type du domaine fondamental. Dans tous ces attributs, les valeurs UNKNOWN, UNDEFINED, NULL sont incluses :

Les attributs flous de Type 1 : Ce sont des attributs avec des "données précises", classiques sur lesquels nous pouvons définir des étiquettes linguistiques. La représentation de ces attributs de données est similaire à celle des données précises, en plus l'information sur les étiquettes linguistiques ainsi que les informations sur le type d'attribut sont stockées dans la base de méta connaissances floues.

Les attributs flous de Type 2 : Ce sont des attributs qui rassemblent des données précises aussi bien qu'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 2.2. Ces attributs prennent le type des données dont la représentation a été décrite dans la section 3.3.1. Ils permettent aussi la représentation des informations incomplètes dans la forme de UNKNOWN, UNDEFINED et NULL.

Dans le Tableau 3.2, nous découvrons le système utilisé pour représenter les attributs flous de Type2. Ainsi, nous remarquons qu'un attribut flou de Type 2, appelé par exemple F, est composé, en réalité, par 5 attributs classiques :

Type de donnée	FT	F1	F2	F3	F4
UNKNOWN	0	NULL	NULL	NULL	NULL
UNDEFINED	1	NULL	NULL	NULL	NULL
NULL	2	NULL	NULL	NULL	NULL
CRISP	3	d	NULL	NULL	NULL
LABEL	4	FUZZY_ID	NULL	NULL	NULL
INTERVAL	5	n	NULL	NULL	m
APPROXIMATELY	6	d-marge	marge	-marge	d+marge
TRAPEZE [α,β,γ,δ]	7	α	β-α	γ-δ	δ

Tableau 3.2 : Représentation interne des attributs flous de type 2

- **FT** : entrepose le type de la valeur que peut prendre l'attribut en question comme 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 les entreposer 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. Cependant, pour réduire les opérations 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 : [α, β, γ, δ]. Dans F2 et F3 quelques opérations sont entreposées pour simplifier les équations utilisant ce type de données.

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 sur les domaines linguistiques dont la représentation a été décrite dans la section 3.3.1, comme par exemple, la valeur {0.8/Alger, 0.2/Media} qui exprime qu'une ville est plus possible être proche d'Alger que de Media. Ils acceptent aussi les données UNKNOWN, UNDEFINED et NULL.

La table 3.3 montre le système utilisé pour représenter ces attributs. Un attribut flou de 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 , UNDEFINED , NULL , SIMPLE et DISTRIBUTION de POSSIBILITE .
- **Liste de n paires**, avec n ≥ 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 normalisée).

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 champs à NULL. Toutes ces caractéristiques ainsi que les objets définis sur ce type d'attribut sont entreposés dans la BMCF.

Type de donnée	**FT**	**FP1**	**F1**		**FPn**	**Fn**
UNKNOWN	0	NULL	NULL	...	NULL	NULL
UNDEFINED	1	NULL	NULL	...	NULL	NULL
NULL	2	NULL	NULL	...	NULL	NULL
SIMPLE	3	1	D	…	NULL	NULL
DISTRI .POS	4	p_1	d_1	...	p_n	d_n

Tableau 3.3 Représentation interne des attributs flous de type 3

Les attributs flous de Type 4 : Ces attributs sont définis de la même manière que les attributs de type 3 à la différence qu'il n'est pas nécessaire d'avoir des relations de similitude entre les étiquettes.

Attributs flous représentants des degrés flous

Le domaine de ces degrés peut être représenté dans l'intervalle [0.1], bien que d'autres valeurs soient également permises, comme une distribution de possibilité.

La signification de ces degrés dépend de leur utilisation. Le traitement des données sera différent selon la signification. Les significations les plus utilisée sont : degré d'accomplissement, degré d'incertitude, degré de possibilité et degré d'importance [GAL 06], nous pouvons aussi définir et utiliser d'autres significations.

Les attributs flous de Type 5 (Degré pour chaque valeur d'un attribut)

Quelques attributs peuvent avoir un degré flou associé à eux. Ceci implique que chaque valeur de cet attribut (dans chaque tuple ou instance) a un degré associé, mesurant le niveau d'incertitude en cette valeur. Afin de l'interpréter, nous devons savoir la signification du degré et la signification de l'attribut associé.

Les attributs flous de Type 6 (Degré pour un ensemble de valeurs de différents attributs)

Dans ce cas, le degré est associé à quelques attributs. Il représente l'incertitude de quelques attributs en seulement un degré.

Les attributs flous de Type 7 (Degré pour l'instance entière de la relation)

Ce degré est associé au tuple entier de la relation et pas exclusivement à la valeur d'un attribut spécifique du tuple (ou instance). Habituellement, il peut représenter le degré d'appartenance de ce tuple (ou instance) à la relation (ou à la table) de la base de données. Ce degré représente le degré flou d'une relation floue.

Les attributs flous de Type 8 (Degré non-associé)

Il y a des cas dans lesquels l'information imprécise, que nous souhaitons exprimer, peut être représentée en utilisant seulement un degré, sans associer ce degré à d'autre valeur spécifique différente. Par exemple, le danger d'une maladie peut être exprimé par un degré flou.

La base de méta connaissances floues

Comme nous avons vu dans la section précédente, il y a des informations sur les attributs décrits qui doivent être enregistrés d'une façon accessible par le système. La base de méta connaissance organise toutes les informations concernant la nature imprécise de ces attributs. On considère la base de méta connaissance comme une extension du système de catalogue ; ainsi, les informations sont organisées en utilisant des tables ou des relations. Les éléments enregistrés dans la base de méta connaissances floues concernent principalement:

- Les attributs de la base de données qui nécessitent un traitement imprécis.
- Le type de ces attributs.
- Les éléments définis sur la base de données, c.-à-d., les quantificateurs flous des requêtes.
- Les objets flous définis sur chaque attribut :
    - Étiquettes linguistiques
    - Valeurs approximatives
    - Relations de proximité
    - Qualificateurs de seuil de requête

3.5 Conclusion

Dans ce chapitre nous avons présenté les aspects théoriques nécessaires à la représentation et au traitement des données imparfaites dans les bases de données relationnelles, ce qui implique de nouvelles définitions dans les structures et les traitements des données , ainsi qu'au niveau des opérateurs nécessaires à la manipulation des nouveaux types de données.

Nous avons présenté une interface pour permettre l'extension d'un système de gestion de base de données relationnel classique pour qu'il puisse représenter et manipuler des informations floues.

Bien que le modèle GEFRED intègre des outils importants de représentation de l'information floue, la représentation elle-même des nouvelles définitions est limitée, car en réalité, l'interprétation des concepts flous dépend de plusieurs paramètres qui doivent être pris en considération dans la représentation.

Chapitre 4

Extension du modèle GEFRED

4.1 Introduction

La représentation de l'information floue vise en premier lieu à mettre à la disposition de l'être humain un langage simple qui rapproche le plus possible de son langage naturel, qui permet d'une part de formuler des requêtes flexibles contenant des termes et des expressions simples à utiliser, d'autre part, permet d'obtenir des réponses plus pertinents pour les différents utilisateurs .
Nous avons vu dans les sections précédentes que le langage FSQL basé sur le modèle GEFRED permet d'utiliser divers termes du langage naturel sous formes des concepts flous (comme les étiquettes linguistiques et les modificateurs linguistiques). Cependant, ces concepts flous vont être utilisés par différents utilisateurs et peuvent être interprétés de différentes manières selon le contexte, ce qui peut influer considérablement sur la pertinence des résultats.

Malgré que GEFRED permet de représenter des données floues d'une manière riche et efficace en comparaison avec d'autres modèles de bases de données floues, mais par rapport à la diversité et à la relativité des informations du monde réel, sa représentation s'avère rigide et limitée, le fait de figer le sens d'un terme linguistique ou d'un concept flou d'une manière générale à un seul sens sans tenir compte ni de l'utilisateur, ni du contexte d'utilisation qui peuvent avoir un grand impact sur la sémantique des concepts flous, est considéré comme un défaut dans la représentation des données et une limitation qui peut réduire la flexibilité du système et influer sur la pertinence des résultats délivrés lors du traitement des requêtes .

Pour pallier à cette limite, Nous proposons une extension du modèle GEFRED afin d'avoir plus de flexibilité sur les deux niveaux : représentation et interrogation des données floues, le système donne à un concept flou une représentation multidimensionnelle qui vise à couvrir les différentes situations de l'utilisateur. Dans la phase de traitement, il permet une adaptation floue des requêtes afin d'augmenter la pertinence des résultats.

4.2 Architecture proposée

Nous proposons dans cette partie, une extension de l'architecture FIRST afin de permettre au système une représentation élargie des concepts flous pour couvrir les différents situations existants dans le monde réel, ainsi d'offrir une grande flexibilité dans l'interrogation des données floues, et ceci dans l'objectif de satisfaire les besoins de l'utilisateur autant que possible. Pour ce faire, nous proposons une extension à deux niveaux : base de méta connaissances floues, et serveur FSQL.

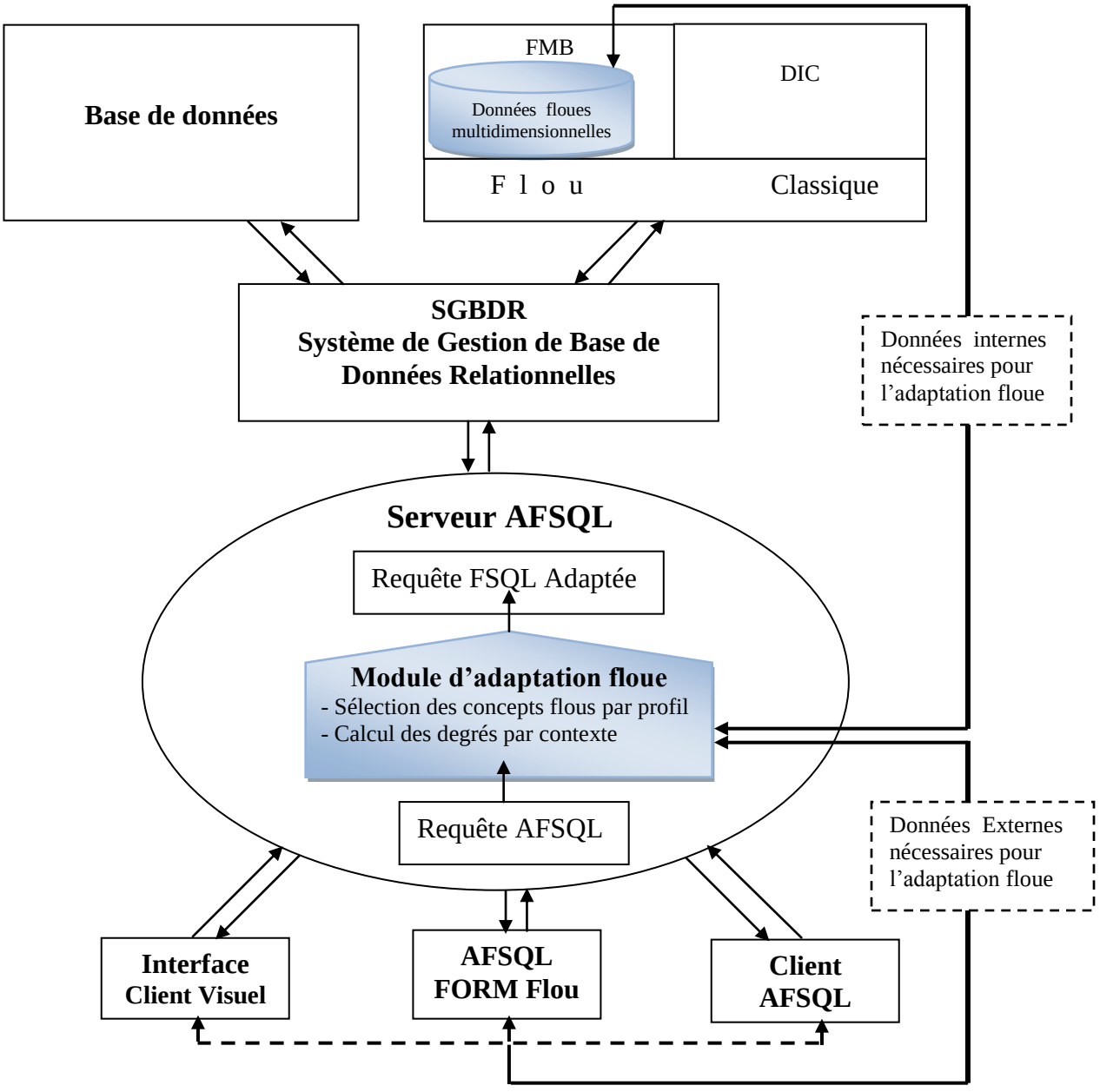


Figure 4.1 Nouvelle architecture de FIRST

Au niveau de la base de méta connaissances floues, une extension est faite pour permettre la représentation de données floues avec plusieurs sens dépendamment aux différentes situations (profil et contexte) en utilisant les différents types flous décrit précédemment.

Au niveau du serveur FSQL, nous procédons à une extension avec un module d'adaptabilité floue des requêtes par rapport aux données multidimensionnelles des concepts flous stockées dans la base de méta connaissances, ce qui permet de traiter les requêtes avec une grande flexibilité.

4.2.1 Avantage du nouveau système

Notre approche fournit une méthodologie pour l'usage de la connaissance contextuelle pour augmenter la pertinence des résultats.

La contribution de cette méthode est d'aider à réaliser des requêtes flexibles en capturant et en utilisant la sémantique d'une requête à travers le profilage d'utilisateur et le contexte d'utilisation. Notre travail ne contribue pas par une nouvelle approche à la construction, représentation, ou utilisation d'un profil utilisateur, mais par montrer comment les informations concernant la situation de l'utilisateur (profil et contexte) modélisées d'une manière floue, peuvent être utilisées dans une approche globale de représentation et d'interrogation de données floues.

Notre système offre les avantages suivants sur les deux niveaux : représentation des données et interrogation des données.

- **Représentation des données** :

Une richesse et diversité de sens par rapport à plusieurs dimensions modélisant les différents situations du monde réel, ce qui permet de donner à un terme représenté par une étiquette linguistique plusieurs sens relativement à la situation de l'utilisateur : profil avec préférences, contexte d'utilisation (temps, localisation).

- **Interrogation des données :**

Le nouveau système offre plus de flexibilité, ainsi qu'une adaptabilité des requêtes aux préférences utilisateur et au contexte d'utilisation, ce qui augmente considérablement la pertinence des résultats renvoyés par le système.

4.3 Représentation multidimensionnelle des données floues

Pour avoir une représentation qui rapproche aux concepts du monde réel, ainsi d'obtenir des résultats appropriés dans le traitement des requêtes, il est nécessaire de représenter les concepts flous en prenant en compte :

- ✓ Les préférences des utilisateurs concernant la sémantique des concepts flous.
- ✓ Les éléments du contexte qui peuvent influer sur la sémantique des termes flous utilisés dans les requêtes.

Pour ce faire, nous avons introduit deux nouveaux concepts : profil flou et contexte flou.

4.3.1 Profil flou

La notion de profil utilisateur est loin d'être définie de façon standard malgré les différentes approches développées. La structure et le contenu d'un profil change d'un domaine à l'autre et d'une application à l'autre. Dans [MOU 07], le profil utilisateur est défini comme suit :

"Un profil utilisateur est une collection d'information sur l'utilisateur et cette collection peut être vue comme un ensemble de caractéristiques avec des valeurs associées contenant par exemple les préférences d'utilisateurs''

Dans le cadre de notre étude, nous allons utiliser les profils pour exprimer les préférences des utilisateurs concernant la sémantique des concepts flous utilisés dans le modèle GEFRED, ce qui peut augmenter la pertinence des résultats lors du traitement des requêtes, ces derniers peuvent être récapitulés dans le tableau suivant :

- **Etiquettes linguistiques**
  - ✓ Valeurs trapézoïdes [alpha, beta, gamma, delta,] (Attribut flou de types 1 et 2)
  - ✓ Distributions de possibilités (Attribut flou de type 3, 4)
- **Valeurs approximatives**
  - ✓ La marge des valeurs approximatives
  - ✓ La distance minimale pour considérer deux valeurs comme très séparées
- **Relations de similitude** pour les attributs flous de type 3
- **Qualificateurs flous**
- **Quantificateurs flous**

Tableau 4.1 concepts flous d'un profil flou

4.3.2 Contexte flou:

- **Notion de contexte**

Plusieurs définitions du terme *contexte* peuvent être trouvées dans la littérature, notre objectif n'est pas de discuter la différence entre ces définitions, ni d'étudier son application dans les différent domaines (recherche d'information, base de données, interface homme machine, etc.).

Dans le cadre de notre étude, nous nous intéressons aux éléments du contexte qui peuvent influer sur la sémantique des concepts flous modélisés dans GEFRED.

Selon le dictionnaire free on-line dictionary [HOW 11], le contexte est tout ce qui entoure et donne du sens à quelque chose d'autre. Dans le dictionnaire en ligne Longman [LON 11], le contexte est la situation, les événements ou l'information qui sont liés à quelque chose et qui vous aident à le comprendre.

Dans ce cas, le contexte est lié directement au sens des concepts, et ne peut pas être ignoré si on veut une meilleure interprétation des concepts.

- **Le contexte dans un environnement imprécis:**

Les informations du contexte peuvent être incomplètes si certaines données sur le contexte sont inconnues. Généralement ça concerne les informations qui peuvent être changées dans le temps, ajoutant les situations d'imprécisions des indicateurs du contexte.

Les informations sur le contexte peuvent être récupérées à partir de sources différentes (par exemple le lieu parvient d'un outil de localisation GPS, la date du système d'exploitation, .etc.) avec des représentations aussi différentes. Donc le modèle de contexte doit supporter plusieurs représentations du même contexte pour faciliter l'interfaçage entre la source de l'information du contexte et l'application qui l'utilise.

Dans le cadre de notre étude, nous nous intéressons au contexte d'environnement dont les informations sont le lieu et le temps.

- **Lieu :** cette propriété représente le lieu où se trouve l'utilisateur lorsqu'il lance la requête Le sens d'un terme linguistique peut varier selon le lieu, si on prend le cas d'une agence immobilière, si on considère deux régions qui sont complètement différents dans le niveau de

vie, le terme 'bon marché' ne peut pas avoir le même sens pour les deux régions, en prenant les applications du commerce électronique par exemple, on doit tenir compte des aspects de la localisation afin de donner à chaque région le sens approprié.

➢ **Temps :** cette propriété décrit le moment exact de l'interrogation, elle inclut le jour et l'heure.
Très souvent, un terme linguistique peut avoir un sens qui peut changer en fonction du temps, par exemple en fonction de la saison, il est très intéressant de représenter les différents sens avec sa variation en fonction du temps dépendamment du contexte d'utilisation, on peut modéliser des étiquettes linguistique du temps comme : saison, mois, semaine.
La représentation temporelle floue est inspirée de [EUV 98]. La discrétisation du temps se fait suivant la précision requise (granularité) pour les informations temporelles à modéliser : si toutes les données sont connues à une minute près par exemple, une unité sur l'axe du temps représentera une minute. Dans de nombreux problèmes, cette modélisation fournit une précision suffisante pour les calculs effectués et permet une implémentation plus simple. On peut distinguer les cas suivant :

❖ **Dates**

Les dates sont décrites par des ensembles flous μ_d. $\mu_d(x)$ est le degré d'appartenance de l'instant x à la date d. d est l'ensemble de tous les instants qui représentent la date. On peut aussi interpréter l'ensemble flou μ_d comme une distribution de possibilités π_d, auquel cas $\pi_d(x)$ est le degré de possibilité pour x d'être réellement cette date d. Pour une date donnée, il est plus ou moins possible que cette date corresponde précisément à l'instant que l'on cherche à représenter. Suivant la nature de cet instant, la distribution de possibilités peut prendre plusieurs formes.

❖ **Durées**

Les durées sont elles aussi représentées par des ensembles flous. Si on a deux instants flous t1 et t2, représentés par deux distributions trapézoïdales ($\alpha i,\beta i,\gamma i,\delta i$), la durée floue qui les sépare est aussi une distribution trapézoïdale de la forme ($\alpha 2-\delta 1,\beta 2-\gamma 1,\gamma 2-\beta 1,\delta 2-\alpha 1$). Une durée négative (resp. positive) correspond à t_1 avant (resp. après) t_2.

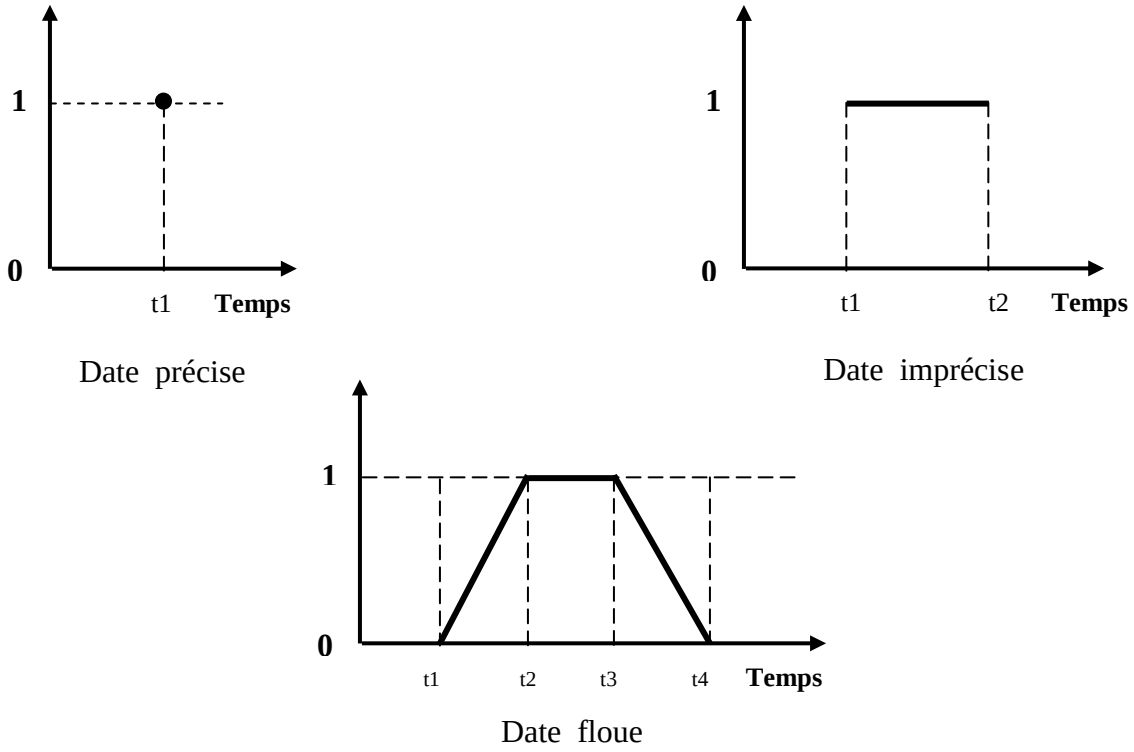


Figure. 4.2 : différentes représentations floues de dates

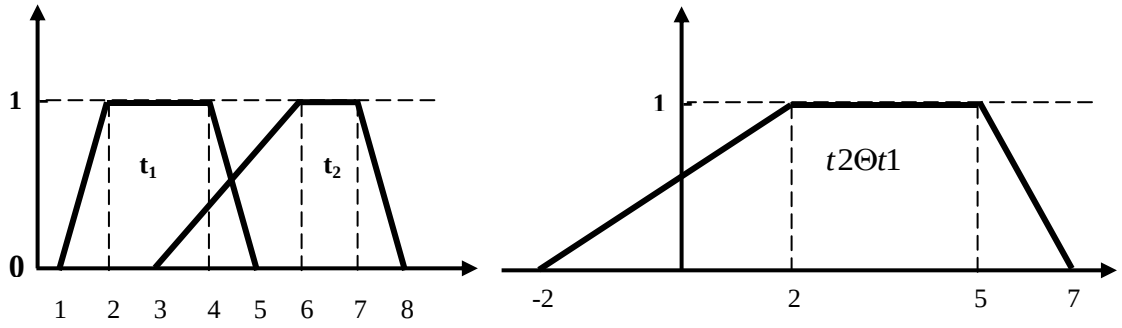


Figure. 4.3 : calcul d'une durée à partir des dates floues

4.3.3 Application d'un profil flou à un contexte flou

En pratique, une requête qui provient d'un utilisateur appartenant à un profil donné, se produit automatiquement dans un contexte donnée (temps, lieu, etc.). Par conséquent, pour chaque profil i, les concepts flous doivent être définis par rapport aux différents contextes possibles.

Contextes	Profil flou i
Contexte flou 1	Définitions des concepts flous
Contexte flou 2	Définitions des concepts flous
……	……
Contexte flou n	Définitions des concepts flous

Tableau 4.2 : Définitions des concepts flous d'un profil donné i dans différents contextes

4.4 Adaptation floue aux données multidimensionnelles

4.4.1 Adaptation floue d'une requête

La procédure d'adaptation se fait en suivant les étapes suivantes :

Etape1 : Capturer les données sur le profil et le contexte.

Etape2 : Pour chaque concept flou, on récupère de la base de méta connaissances floues:

- La valeur appropriée pour le profil
- Le contexte flou défini pour le concept et le profil

Dans cette étape, la valeur retournée est le résultat d'adaptation par rapport au profil, il reste à adapter cette dernière au contexte.

Etape3 : pour chaque paramètre de contexte *p* :

- Calculer le degré d'appartenance du paramètre *p* par rapport au contexte flou (lié au concept flou) si le contexte appartient à un domaine ordonné.
- Calculer le degré de similarité du paramètre *p* par rapport au contexte flou (lié au concept flou) si le contexte appartient à un domaine non ordonné.

Etape4 : appliquer le degré obtenu sur les concepts flous récupérés précédemment et exécuter la requête.

Etape5 : afficher le résultat en prenant en considération les préférences de représentation.

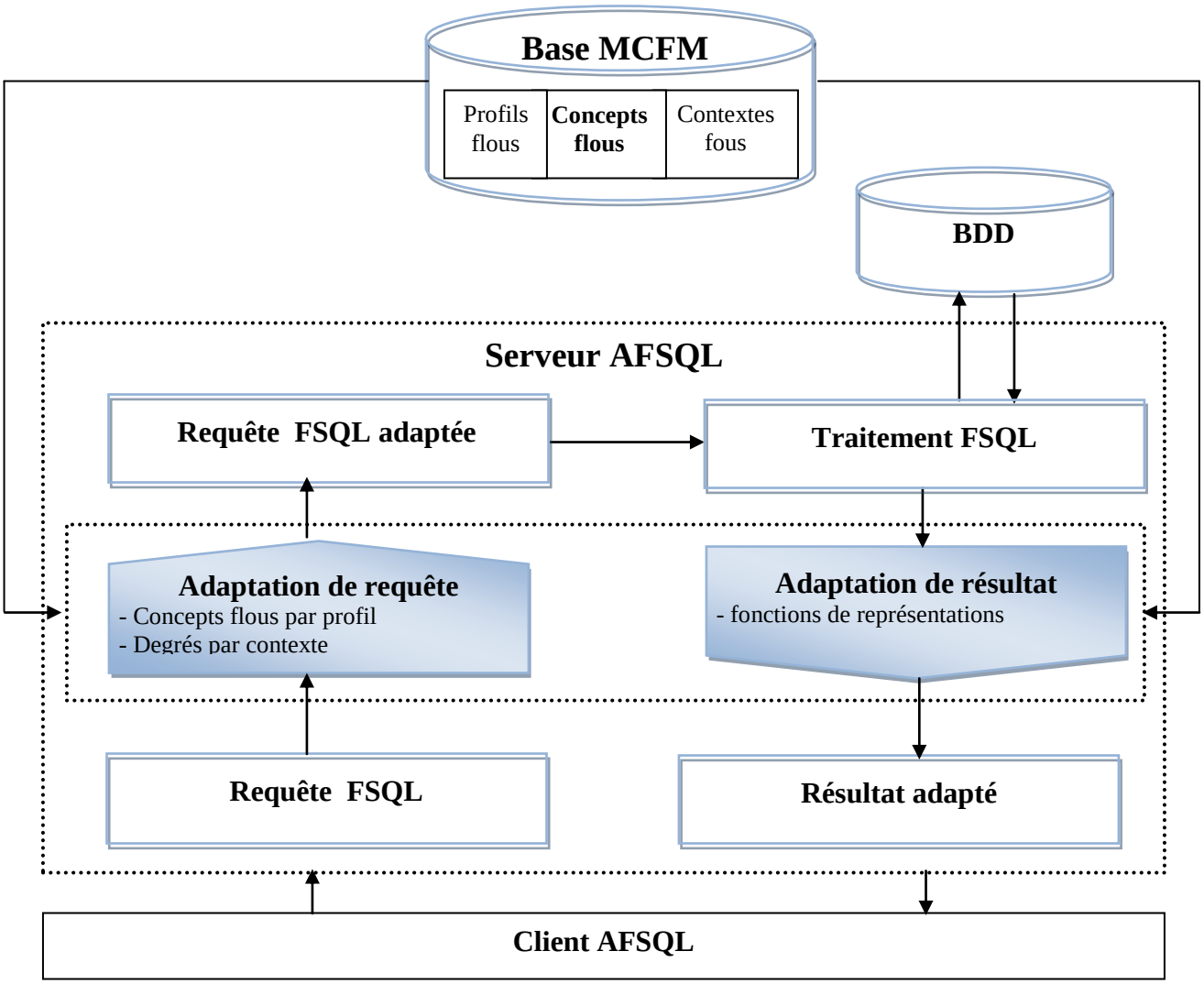


Figure. 4.4: Adaptation floue d'une requête FSQL

Les paramètres du contexte qu'on va utiliser sont le temps et le lieu. Nous allons détailler la procédure d'adaptation par rapport à chaque paramètre.

- **Adaptation par rapport au temps**

Dans ce cas, il faut calculer le degré d'appartenance du temps récupéré du contexte de l'utilisateur, par rapport au contexte lié à l'information multidimensionnelle stocké dans la base de méta connaissances floues, nous pouvons distinguer les cas suivants :

- **Date simple d:**

Dans ce cas, le degré d'appartenance d'un instant t

μt = 0 si t est différent de la date d

μt = 1 si t est égal de la date d

- **Durée**

Dans ce cas, on doit calculer le degré d'appartenance du paramètre temporel t du contexte courant par rapport à la durée liée à l'information. Reprenant la durée représentée dans la figure 4.3 entre deux instants t1, t2, le résultat sera t2 Θ t1.

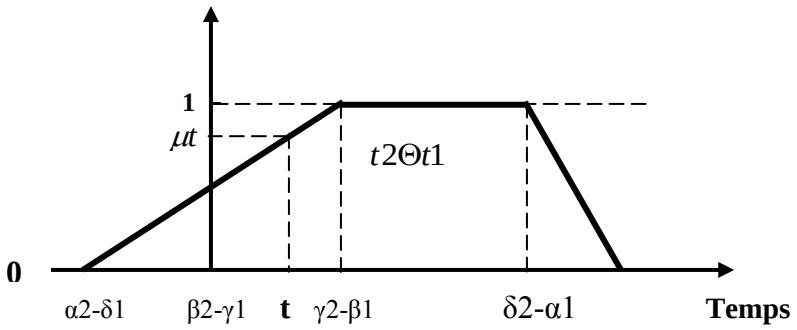


Figure. 4.5: calcul de degré d'appartenance d'un instant à une durée

Exemple 4.1 :

Soit un profil flou *P1* contenant un concept flou *chaud.*

Nous Considérons maintenant la représentation du terme contextuel ''*Hiver*'' qui représente la saison d'hiver comme dans la figure 4.6

Si un utilisateur appartient à un profil donné P1, et se trouve dans un contexte flou C1, les concepts liés à ce dernier vont être appliqués avec un degré égal au degré d'appartenance du contexte actuel par rapport au contexte flou.

Dans notre exemple, si on considère deux cas :

Cas 1 : - l'information *temps* du contexte utilisateur indique la valeur suivante : date = 15 décembre.

- Le contexte flou = *Hiver*

En calculant le degré d'appartenance du contexte de l'utilisateur au contexte flou défini dans la base de méta connaissances floues le résultat sera égal à 0,7.

Cas 2 : - l'information *temps* du contexte utilisateur indique la valeur suivante : date = 13 Mars

- Le contexte flou = *Hiver*

En calculant le degré d'appartenance du contexte de l'utilisateur au contexte flou défini dans la base de méta connaissances floues le résultat sera égal à 0,5.

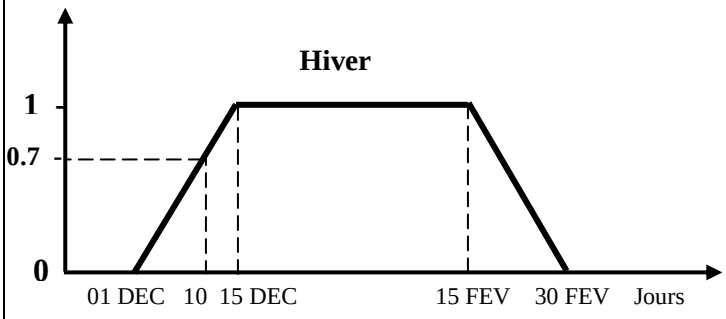
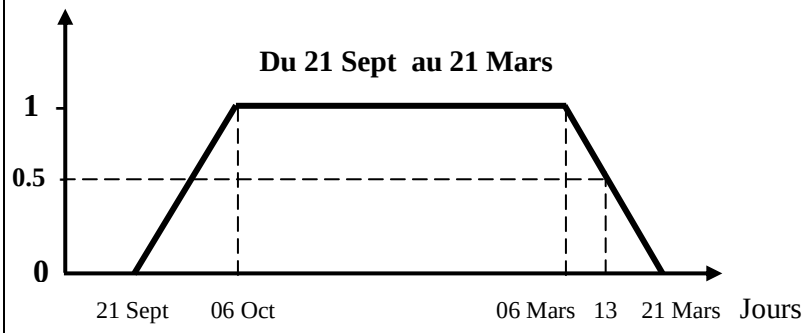
Durée	**Représentation**
Contexte d'application =hiver Contexte de requête= 10 Dec	
Contexte d'application = Du 21 Sept au 21 Mars Contexte de requête = 13 Mars	

Figure 4.6 : Adaptation par rapport à une durée floue

- **Adaptation par rapport à la localisation :**

L'adaptation par le paramètre de localisation se fait en suivant les étapes suivantes :

- Capturer la localisation actuelle de l'utilisateur (par exemple par un outil de localisation GPS)
- Calculer le degré de similarité de la localisation obtenue du contexte par rapport aux différentes localisations des contextes d'application liée au terme linguistique donné.
- Le degré de similarité de la localisation du contexte pour ce profil est égal au degré maximal par rapport aux différentes localisations.

Exemple 4.2 :

Supposant que nous avons les différentes valeurs trapézoïdes du terme linguistique *''chaud''* pour un profil donné.

Localisation	**Profil flou P_{Fi}**
Nord	**concepts flous** : **Chaud** : **[25, 30, 35, 40]**
Sud	**concepts flous** : **Chaud** : **[30, 35, 40, 50]**

Tableau 4.3 : Représentation des valeurs du terme *chaud*

Si un utilisateur situé à Ouargla par exemple, lance une requête comprenant le terme linguistique ''chaud '', alors on peut avoir le degré de similarité de son localisation en se basant sur une table de similarité comme celle présentée dans le tableau suivant :

Ville	**Sud**
Alger	0
Média	0.2
Djelfa	0.4
Elbayath	0.6
Ouargla	0.8
Adrar	1
Tamanrasset	1

Tableau 4.4 : valeurs de similarité entre ville et région

La valeur qui correspond à la ville *''* Ouargla *''* est 0.8, ce qui peut être traduit par le fait que le terme utilisé dans la requête sera valide à un degré de 0. 8 :

$$\begin{cases} Terme = chaud \\ contexte.localisation = \text{Ouargla} \end{cases} \rightarrow terme = [30,35,40,50]\ 0.8$$

4.5 Attributs flous multidimensionnels

Pour permettre une représentation multidimensionnelle, nous définissons trois nouveaux attributs sur lesquels nous pouvons enregistrer des paramètres contextuels pour l'utiliser dans l'opération d'adaptation, les nouveaux attributs sont :

- **Attribut flou de Type 1 Etendu :** Ce sont des attributs avec des "données précises", classiques sur lesquels nous pouvons définir des étiquettes linguistiques. Ces attributs sont définis de la même manière que les attributs de type 1 à la différence qu'ils seront définit pour un profil et dans un contexte d'utilisation, les informations sur les profils et les contextes d'applications sont stockées dans la base de méta connaissances floues.
- **Attribut flou de Type 2 Etendu :** Ce sont des attributs qui rassemblent des données précises aussi bien qu'imprécises dans un référentiel ordonné définit pour un profil et un contexte d'application, les informations sur les profils et les contextes d'applications sont stockées dans la base de méta connaissances floues.
- **Attribut flou de Type 3 Etendu** : Ce sont les attributs du "domaine discret non ordonné avec ressemblance". Ces attributs sont définis de la même manière que les attributs de type 3 à la différence qu'ils seront définit pour un profil et un contexte d'application, les informations sur les profils et les contextes d'applications sont stockées dans la base de méta connaissance floue.

4.6 Implémentation

L'implémentation du modèle est effectuée à trois niveaux.

- Au niveau du SGBDR : Le système doit être capable de traiter les nouveaux opérateurs flous.
- Au niveau de la base de données : La base de données se compose de tous les éléments permanents décrivant une partie de l'univers. Comme nous sommes concernés par la représentation des données imprécises, nous devons définir la forme de stockage de ces derniers. Ainsi, la représentation des données doit être étendue afin de traiter ce type d'informations.
- Au niveau de la base de méta connaissances : où toute information que le système doit avoir sur les données présentées est stockée d'une façon accessible par le système (métadonnées).

4.6.1 Implémentation sous PostgreSQL

L'implémentation du modèle peut être réalisée sur n'importe quel système de gestion de base de données relationnel, cependant, dans notre travail, nous avons décidé de la réaliser en utilisant le SGBDR libre PostgreSQL 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/pgSQL qui est avéré pour être complètement efficace. Ainsi, l'extensibilité de PostgreSQL nous a permis de procéder à une méthode d'implémentation facile et efficace.

4.6.1.1 Présentation de PostgreSQL

PostgreSQL est un système de gestion de bases de données relationnelles objet (SGBDRO) fondé sur POSTGRES qui a été développé à l'université de Californie au département des sciences informatiques de Berkeley. POSTGRES est à l'origine de nombreux concepts qui ne seront rendus disponibles au sein des systèmes de gestion de bases de données commerciales que bien plus tard, PostgreSQL est un descendant OpenSource du code original de Berkeley. Il supporte une grande partie du standard SQL tout en offrant de nombreuses fonctionnalités modernes [PGS 08] :

- requêtes complexes ;
- clés étrangères ;
- triggers ;
- vues ;
- intégrité des transactions ;
- contrôle des accès concurrents (multiversion concurrency control).

De plus, PostgreSQL est extensible par l'utilisateur de plusieurs façons. En ajoutant, par exemple :

- de nouveaux types de données ;
- de nouvelles fonctions ;
- de nouveaux opérateurs ;
- de nouvelles fonctions d'agrégat ;
- de nouvelles méthodes d'indexage ;
- de nouveaux langages de procédure.

Et grâce à sa licence libre, PostgreSQL peut être utilisé, modifié et distribué librement, quelque soit le but visé, qu'il soit privé, commercial ou académique [PGS 08].

4.6.1.2 Extensibilité de PostgreSQL

Le SGBD PostgreSQL est extensible parce qu'il opère grâce à un système de catalogues. Quiconque est familier des systèmes de bases de données relationnelles standard sait que les informations concernant les bases, les tables, les colonnes, etc. y sont stockées dans ce qu'on nomme communément des catalogues systèmes (certains systèmes appellent cela le dictionnaire de données). Pour l'utilisateur, les catalogues ressemblent à des tables ordinaires, mais le SGBD y enregistre ses registres internes. À la différence des autres systèmes, PostgreSQL enregistre beaucoup d'informations dans ses catalogues : non seulement l'information concernant les tables et les colonnes, mais aussi l'information concernant les types de données, les fonctions, les méthodes d'accès, etc.

Ces tables peuvent être modifiées par l'utilisateur, Puisque PostgreSQL fonde ses opérations sur ces tables, il peut être étendu par les utilisateurs. En comparaison, les systèmes de bases de données conventionnels ne peuvent être étendus qu'en modifiant les procédures dans le code source ou en installant des modules spécifiquement écrits par le vendeur de SGBD.

De plus, le serveur PostgreSQL peut incorporer du code utilisateur par chargement dynamique. C'est-à-dire que l'utilisateur peut indiquer un fichier de code objet (par exemple une bibliothèque partagée) qui code un nouveau type ou une nouvelle fonction et PostgreSQL le charge au besoin. Il est encore plus facile d'ajouter au serveur du code écrit en SQL. La possibilité de modifier son fonctionnement ‘’à la volée‘’ fait de PostgreSQL un outil unique pour le prototypage rapide de nouvelles applications et de structures de stockage [PGS 08].

4.6.1.3 FIRST sous PostgreSQL

Comme nous avons vu précédemment, l’implémentation de GEFRED se fait sur plusieurs niveaux :

- Niveau de base de données : La base de données 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 base de données. Par conséquent, il doit étendre la représentation des données pour héberger ce type d'information.
- Niveau des classes du système : c’est la 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 base de données.

- 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 AFSQL est la partie du système qui contient les règles qui se chargent d'implémenter ces opérations.

Cependant, l'extensibilité du PostgreSQL, nous a permis d'implémenter FIRST d'une manière simple et efficace. Du moment où il donne la possibilité de créer de nouveaux types avec tous les opérateurs nécessaires pour leurs manipulation, ainsi le dictionnaire peut être enrichi avec les nouveaux types dont leur manipulation par PostgreSQL sera facile et rapide.

4.6.2 La base de méta connaissances floues multidimensionnelle

En plus des données concernant la nature imprécise des attributs, nous avons étendu la base de méta connaissances de FIRST présentée dans [MED 95b] afin de permettre le stockage des concepts flous avec une variété de sens qui permet de couvrir les différentes situations des utilisateurs : les données sur les profils des utilisateurs et les contextes d'application des différents concepts flous.

Nous allons présenter les relations composant la base de méta connaissances floues ainsi que ses attributs. La figure 4.7 montre les relations dans la base de méta connaissances floues étendue.

- FUZZY-COL-LIST

Elle contient une description des attributs de la base de données susceptibles du traitement flou. Elle décrit les attributs flous identifiés par (OBJET, COLUMN). 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

Cette table contient une liste d'objets flous définis dans les colonnes de la base de données. En outre, elle contient une classification de ces objets par l'élément OBJECT_TYPE. Cette table contient les éléments suivants :

-*COLUMN_ID* (*Numérique):* Contient le numéro qui identifie la colonne sur laquelle l'objet nommé dans l'élément OBJECT_NAME de cette table est défini. Il constitue une clé étrangère de la table de FUZZY_COL.

- *FUZZY_ID (Numérique):* Assigne un numéro à chaque objet ; ce numéro est utilisé pour identifier l'objet dans le reste des tables. Cet élément, avec l'élément de COLUMN_ID constituent la clé primaire de cette table.

-*FUZZY_NAME* (*Caractère) :* Contient le nom d'objet.

-*FUZZY_TYPE* (Numérique) : Spécifie le type d'objet identifié par l'élément OBJECT_ID. Les valeurs autorisées sont :

- 0 pour l'étiquette linguistique (type trapézoïdal)
- 1 pour un scalaire associé à un traitement de relations de proximité
- 2 pour les étiquettes de qualificateurs définies sur la satisfaction de la requête
- 3 pour les étiquettes de quantificateurs sur la requête.

- FUZZY_LABEL_DEF

Cette table contient les points qui déterminent la fonction d'appartenance correspondant aux étiquettes linguistiques du type trapézoïdal. Les éléments dans cette table sont :

- *LABEL_ID (Numérique) :* Contient le numéro qui identifie l'étiquette par l'élément OBJECT_ID dans la table FUZZY_ OBJECT_LIST. Cet élément constitue une clé étrangère à cette table. L'élément LABEL_ID est la clé primaire de la table FUZZY_ LABEL_DEF.

- ALPHA: nombre. $\alpha = \inf \{x: x \in \text{support(étiquette)}\}$
- BETA: nombre. $\beta = \inf \{x: x \in \textit{noyau}\ (\text{étiquette})\}$
- GAMMA: nombre. $\gamma = \sup\{x: x \in \textit{noyau}\ (\text{étiquette})\}$
- DELTA: nombre. $\delta = \sup\{x: x \in \text{support(étiquette)}\}$

- FUZZY_PROF

Cette table contient les profils flous pour lesquels les différents concepts flous sont associés, les attributs de cette table sont:

- *PROFL_ID (Numérique) :* Contient le numéro qui identifie le profil. Cet élément constitue la clé primaire de cette table.

- *PROFIL_NAME (Caractère) :* Contient le nom qui désigne ce profil et ceci pour faciliter la manipulation des profils.

- FUZZY_CONTEXT

Cette table contient les contextes d'applications liés aux différents concepts flous, les attributs de cette table sont:

- *CON_ID (Numérique) :* Contient le numéro qui identifie le contexte. Cet élément constitue la clé primaire de cette table.
- *CON_NAME (Caractère) :* Contient le nom qui désigne le contexte
- *CON_TYPE (Numérique) :* Contient le type de contexte.

FUZZY_APPROX_MUCH

Cette table contient toutes les informations sur la fonction d'appartenance des étiquettes de type *approximativement*, Ces fonctions sont triangulaires, avec la valeur d'appartenance 1 pour le point sur lequel l'approximation est considérée et peut être caractérisée par ce point, qui sera donnée dans la requête, et par la largeur de la base de triangle. Par conséquent, la table sauvegarde les valeurs ''MARGE'' et ''MUCH'' pour les types 1 et 2.

FUZZY_NEARNESS_DEF

Cette table représente les mesures de proximité ou de similitude entre les différentes valeurs de domaine autorisées pour les éléments de type 3 de la table de FUZZY_COL.

L'information est structurée de cette façon : deux éléments établissent la relation entre deux valeurs du domaine, et le troisième contient leur degré de proximité. Les éléments sont structurés comme suit :

- *OBJECT_ID1(Numérique) :* Contient la première valeur des couples relatifs.
- *OBJECT_ID2(Numérique) :* Contient la deuxième valeur.
- *DEGRÉ (Numérique) :* Contient le degré de la proximité des concepts relatifs des OBJECT_ID1 et OBJECT_ID2. Cette valeur doit être dans l'intervalle [0,1]

Il est à noter que la clé primaire de cette table est constituée par OBJECT_ID1 et OBJECT_ID2.

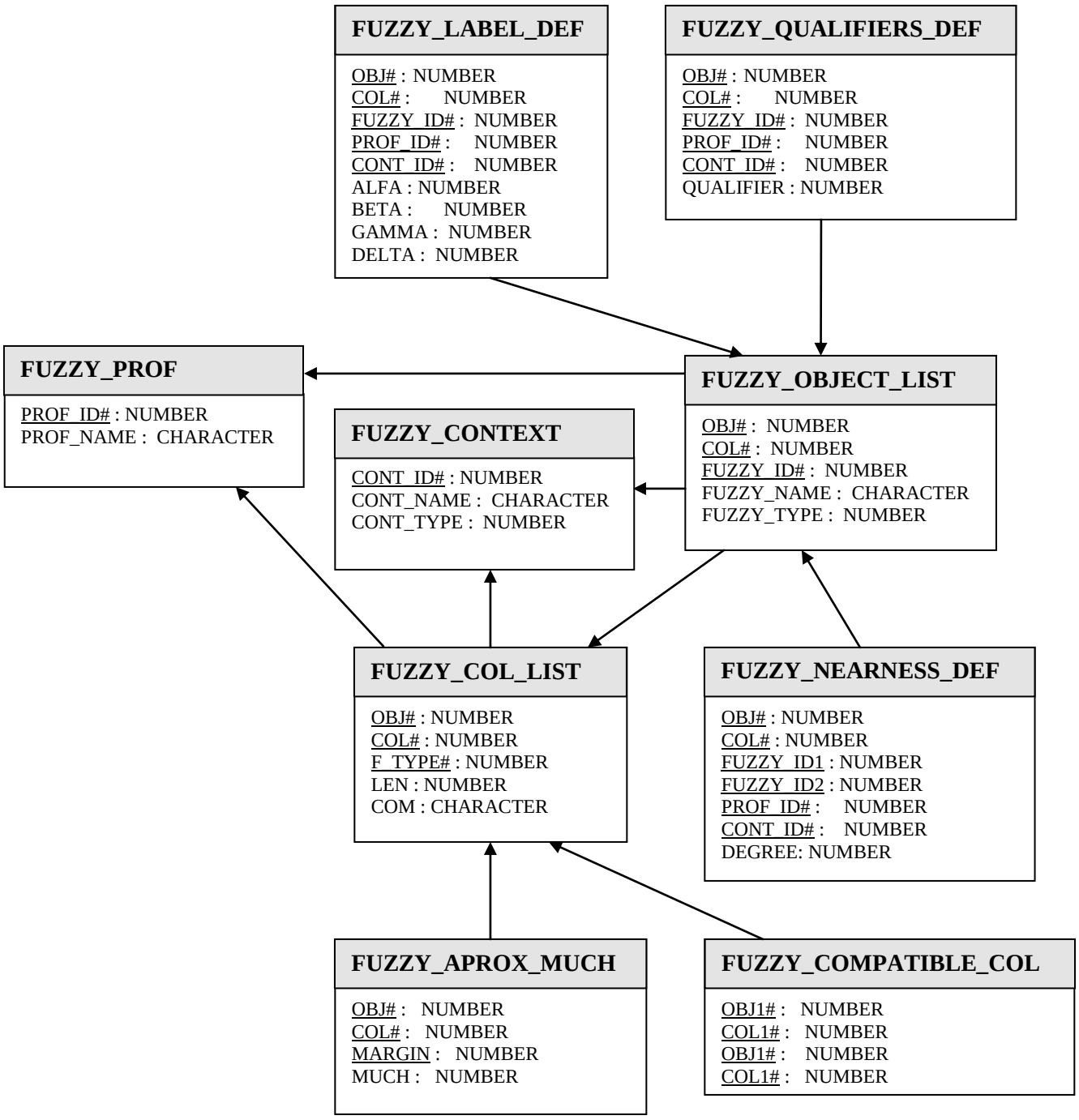


Figure 4.7 : Tables de la base de méta connaissances floues étendue

FUZZY_QUALIFIERS_DEF

Cette table contient les qualificateurs associés à la valeur linguistique donnée dans la table FUZZY_OBJECT_LIST. Les éléments dans cette table sont :

- *LABEL_ID (Numérique) :* est la clé primaire et identifie l'étiquette sur laquelle la définition est établie.
- *QUALIFICATIF (Numérique)* : Valeur de qualificateur pour l'étiquette référencée dans l'élément LABEL_ID.

FUZZY_COMPATIBLE_COL : Elle sauvegarde les attributs flous compatibles entre eux, ç-à-d., ceux qui utilisent les mêmes étiquettes linguistiques.

4.6.3 Le serveur AFSQL

Il a été programmé totalement en PL/pgSQL, un langage de procédures chargeable pour le SGBD PostgreSQL. Grâce à PL/pgSQL nous pouvons grouper un bloc de traitement et une série de requêtes au sein du serveur de bases de données, et bénéficier ainsi de la puissance d'un langage de procédures, mais avec de gros gains en terme de communication client/serveur.

- Les allers/retours entre le client et le serveur sont éliminés
- Il n'est pas nécessaire de traiter ou transférer entre le client et le serveur les résultats intermédiaires dont le client n'a pas besoin
- Les va-et-vient des analyses de requêtes peuvent être évités

Le serveur AFSQL inclut globalement trois types de fonctions :

- **Fonctions d'adaptation floue** : Elles sont utilisées pour comparer les différents paramètres contextuels de la requête de l'utilisateur par rapport aux contextes liés aux concepts de la requête. Elles incluent : les fonctions de calcul de degré d'appartenance, de degré de similarité, ainsi d'agrégation.
- **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).

Pour les fonctions de traduction FSQL vers SQL, comme nous l'avons précédemment présentée, l'extensibilité du PostgreSQL, nous a permis de créer les concepts flous nécessaires pour FSQL

(types floues, comparateurs flous) , ce qui permet au serveur AFSQL de les manipuler d'une manière facile sans avoir besoin d'un autre outil de traduction manuel.

4.6.4 Le Client AFSQL

Le Client AFSQL est un programme indépendant qui sert comme une interface entre l'utilisateur et le Serveur AFSQL. L'utilisateur introduit une requête FSQL et le programme client communique avec le serveur et avec la base de données dans le but d'obtenir le résultat final.
Le client AFSQL permet d'effectuer des interrogations floues avec le langage FSQL, bien qu'il puisse aussi être utilisé pour faire des interrogations classiques avec SQL.

Objectifs et Fonctionnement

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

Modélisation d'une requête dans AFSQL Client

Le logiciel AFSQL 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.
Pour l'édition d'une requête de type classique, Les concepts de base du SQL restent encore valides à savoir les opérateurs logiques, ensemblistes, les comparateurs classiques (exacts).

En plus des opérations classiques du langage SQL, le Client AFSQL offre les différents comparateurs et constantes floues présentés précédemment.

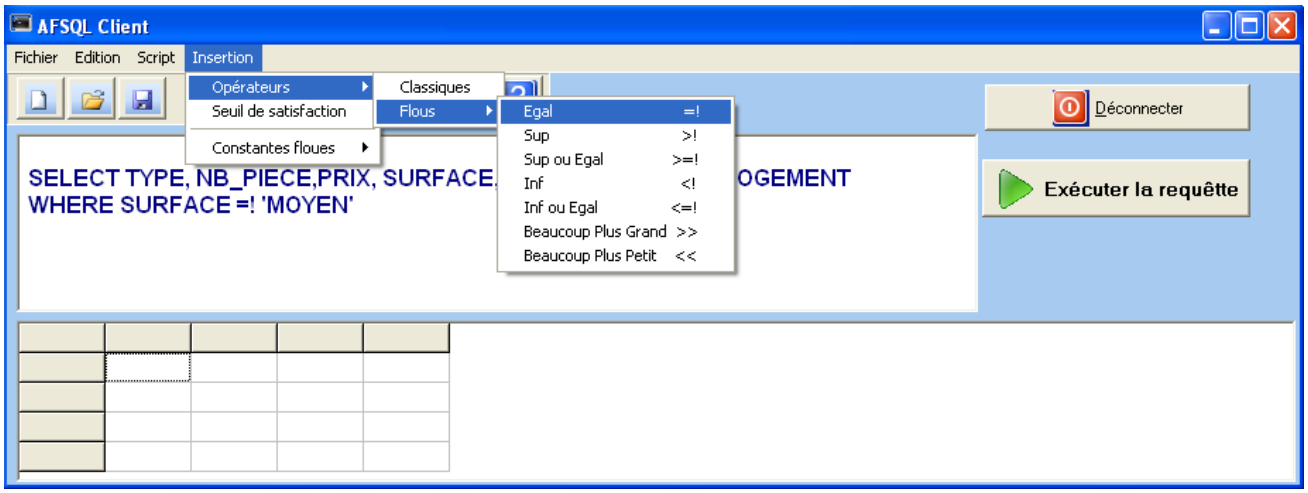


Figure 4.8 : Les comparateurs flous dans AFSQL Client

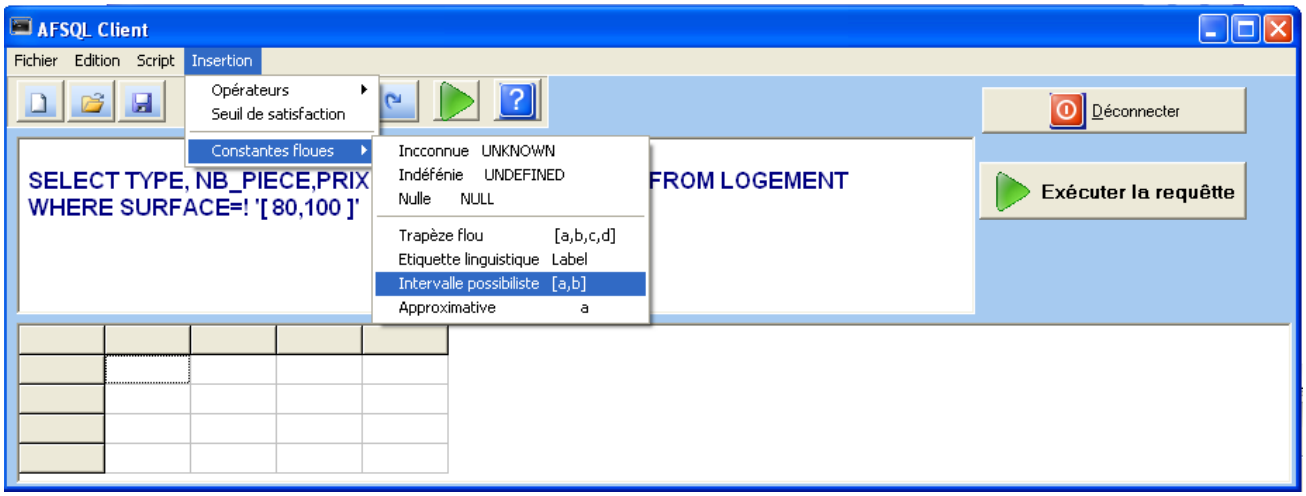


Figure 4.9 : Les constantes floues dans AFSQL Client

Pour l'édition des requêtes floues, dans la première version de ce programme, l'utilisateur doit écrire sa requête en mode textuel, donc il doit avoir une connaissance sur le langage FSQL. Nous comptons développer des outils supplémentaires pour faciliter l'édition des requêtes et éviter aussi les erreurs de différents types.

4.6.5 Interrogation du serveur AFSQL

Le processus d'appel au serveur AFSQL pour une requête FSQL passe par les étapes suivantes:

1. Le client AFSQL envoie une requête FSQL au serveur AFSQL.
2. Le serveur AFSQL analyse la requête :

   - Si la requête contient des concepts flous contextuels, alors les fonctions d'adaptation floue seront exécutées en utilisant les paramètres profil/contexte.

   - Si la requête contient des attributs simples, ou flous non contextuels, ils les utilisent directement.
3. Le serveur AFSQL exécute la requête.
4. Finalement, le client recevra les données résultantes qui sont affichés dans une façon compréhensible pour l'utilisateur et ceci en utilisant les fonctions de représentation.

4.6.6 Cas d'application : Agence immobilière

4.6.6.1 Description de la base de données :

Nous présentons dans cette partie la description de la base de donnée Agence Immobilière pour vente et location des logements de différents types (maisons, villas, appartements …) situés aux différents zones à travers l'Algérie, et ceci pour qu'elle puisse supporter le concept des requêtes flexibles adaptatives . Cette description est comme suit :

Un logement possède des informations d'identification générale qui sont un numéro unique d'identification, un type (villa, appartement, studio, etc.)**,** l'ancienneté ou l'âge de son construction, ainsi que des données de description intérieure qui comprend : le nombre des pièces, la surface totale, ainsi que la surface détaillé du jardin ou du garage s'ils existent, l'état d'aménagement du logement, les vues externes.

Un logement se trouve dans un endroit localisé par rapport aux plusieurs villes ou zone de voisinage.

Quant aux individus qui occupent les logements (les signataires du contrat uniquement), on se contentera de leurs matricules, noms, prénoms, sexes, dates de naissance, adresses et numéros de téléphone. L'historique de l'occupation des logements par les individus est mémorisé, On considère qu'un individu peut être signataire de plusieurs contrats de location. On précise aussi qu'un logement peut faire l'objet de plusieurs locations disjointes dans le temps.

L'agence désire gérer ses activités de location et vente des logements d'une manière simple et efficace en permettant aux clients d'exprimer ses besoins en formulant des requêtes flexibles en utilisant des termes du langage naturel.

4.6.6.2 Détermination des attributs flous

Le schéma de la base de données doit inclure des attributs classiques et non seulement des attributs flous, comme les noms des clients, les numéros de téléphone, etc. Généralement, pour donner une grande souplesse au système, plusieurs attributs flous de type 2 peuvent être définis. Néanmoins, il est nécessaire de considérer que les attributs flous de ces types exigent plus d'espace de mémoire et plus de temps dans le traitement. Et donc, nous devons choisir entre la flexibilité (dans la représentation et le traitement flou) et l'efficacité (dans l'espace de stockage et le temps CPU).

Attributs flous de type1 :

Ces attributs sont bien exactes (classiques, sans imprécision), mais permettent des consultations flexibles en les utilisant dans les conditions flous (avec les comparateurs flous, les constantes floues, et les seuils de réalisation).

Nous pourrions considérer comme attributs flous de type 1: le nombre de pièces et le nombre de services, les valeurs de ces attributs sont généralement biens connus sans ambiguïté.

Attributs flous de type2 :

Ces attributs acceptent autant de données exactes que flous, sous forme de distributions de possibilité sur un domaine fondamental ordonné. Avec ces attributs nous pourrons en outre, enregistrer et utiliser tous les types de constantes floues.

Pour notre application en particulier la majorité des attributs pourrait être de type 2, qui mène à obtenir une base de données aussi flexible que possible. On peut citer particulièrement les attributs suivants :

-Prix : Souvent, le prix n'est pas fixe mais négociable, de sorte que le propriétaire fixe une valeur approximative et, généralement, l'agence serait prête à négocier le prix final de location.

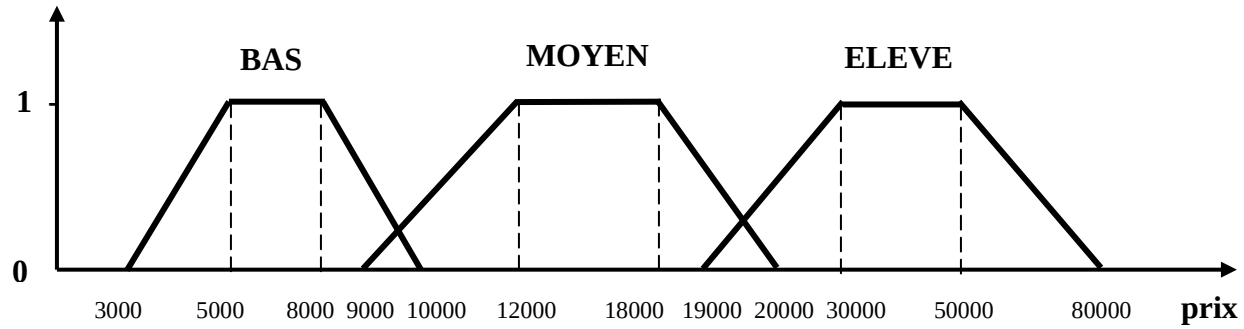


Figure 4.10 : Étiquettes linguistiques de l'attribut prix

Cependant, les valeurs de ces étiquettes peuvent changer en fonctions de profil utilisateur et des paramètres contextuels comme la localisation et le temps. Pour cela, nous allons utiliser la représentation étendue du type 2.

- Surface (m2) : Souvent il est difficile de faire une mesure exacte de la surface d'un logement. La possibilité de stocker des valeurs approximatives est très intéressante, on peut stocker d'autres surfaces (comme la surface du garage et du jardin s'ils existent).

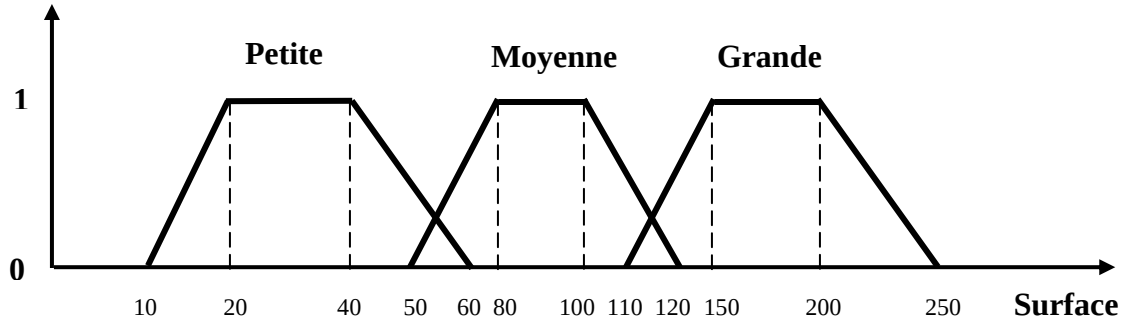


Figure 4.11 : Étiquettes linguistiques de l'attribut surface

- Âge (ancienneté) : Pour quelques locataires, c'est très important de savoir l'ancienneté rapproché du logement qu'ils vont acheter ou même louer, et, généralement, il est tout à fait difficile, sans compter qu'inutile, de savoir l'âge exact. Les bases de données relationnelles floues permettent d'enregistrer des valeurs vagues comme *nouveau, presque nouveau, vieux, très* vieux, *approximativement 15 ans …*

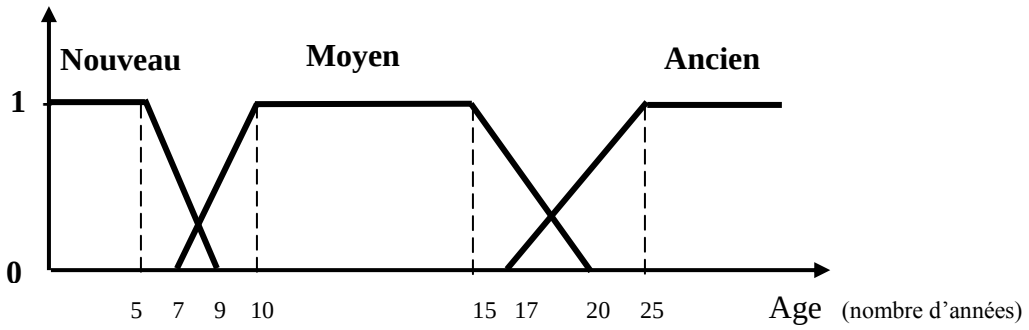


Figure 4.12 : Étiquettes linguistiques de l'attribut ancienneté

Attributs flous de type3 :

Les attributs de ce type sont des scalaires sur lesquels une relation de similitude est définie, de sorte que cette relation indique dans quelle mesure chaque paire d'étiquettes se ressemblent entre eux. Et ceci, parce qu'il n'existe pas de relation d'ordre dans ce domaine. Les attributs flous considérés du type 3 sont particulièrement les suivant : la localisation par rapport aux villes par exemple, le type du logement, la luminosité, le bruit, les vues et l'état d'aménagement.

- Localisation : on définit cet attribut avec la longueur 5, indiquant qu'un logement peut être situé entre cinq villes, avec des divers niveaux (entre 0 et 1) dans chacun d'eux. La relation de similitude entre les différentes villes dépend de la distance entre eux et ses extensions.

Degré de similarité	El-harrach	Oued-smar	Mohammadia	Bordj-elkiffan	Beb-ezouar
El-harrach	1	0.7	0.8	0.5	0.5
Oued-smar	0.7	1	0.4	0.6	0.8
Mohammadia	0.8	0.4	1	0.6	0.5
Bordj-elkiffan	0.5	0.6	0.6	1	0.8
Beb-ezouar	0.5	0.8	0.5	0.8	1

Tableau 4.5 : Relation de proximité de l'étiquette localisation

- Type de logement : Cet attribut distingue les appartements, les villas, les chalets, les studios, Par exemple, on peut dire qu'un appartement est similaire à une maison à un degré de 0.6, de sorte que si nous consultions les logements qui sont approximativement de type *maison* avec le degré minimum

0.6, en résultat de cette consultation nous obtiendrons également *les appartements*. C'est très utile, parce qu'un client qui veut louer une maison, est également un client potentiel pour louer un appartement.

Degré de similarité	Villa	Maison	Chalet	appartement	studio
Villa	1	0.7	0.5	0.4	0.1
Maison	0.7	1	0.8	0.6	0.2
Chalet	0.5	0.8	1	0.4	0.3
appartement	0.5	0.7	0.4	1	0.4
Studio	0.1	0.2	0.3	0.4	1

Tableau 4.6 : Relation de proximité de l'étiquette type logement

- Etat d'aménagement : présente l'état d'aménagement en fonction des conforts et des services qui sont disponible à l'intérieur du logement. Ici nous pouvons définir les étiquettes comme : *luxe*, *bonne*, *normale*, *mauvais*.

Degré de similarité	Mauvais	Normale	Bonne	Luxe
Mauvais	1	0.8	0.5	0.1
Normale	0.8	1	0.7	0.5
Bonne	0.5	0.7	1	0.8
Luxe	0.1	0.5	0.8	1

Tableau 4.7 : Relation de proximité de l'étiquette état d'aménagement

Nous pourrons également enregistrer les valeurs suivantes : la valeur INCONNU si nous ne savons rien sur la valeur d'un certain attribut. La valeur INDEFINIE utilisée si certaine valeur ne s'applique pas (comme par exemple l'attribut pour la taille du jardin, dans un logement qui n'a pas de jardin). La valeur NULL sera utilisée si nous ne savons rien sur cet attribut, c'est-à-dire, si nous

ignorions s'il est ou non applicable (dans l'exemple précédent, si nous ne savons pas s'il existe un jardin ou pas).

4.6.6.3 Modélisation de la base de données floue

Après avoir déterminé les attributs flous, on peut créer les tables de la base de données décrite précédemment avec les attributs floues en langage FSQL comme suit :

```
CREATE TABLE LOGEMENT
(
  N_LOG SERIAL NOT NULL,
  TYPE CHARACTER VARYING (20) NOT NULL,
  NB_PIECE FTYPE1 NOT NULL,
  ADRESSE CHARACTER VARYING (20) NOT NULL,
  PRIX FTYPE2 NOT NULL,
  SURF_LOG FTYPE2 NOT NULL,
  SURF_JAR FTYPE2 NOT NULL,
  SURF_GAR FTYPE2 NOT NULL,
  ANCIENNETE FTYPE2 NOT NULL,
  LOCALISATION FTYPE3 NOT NULL,
  CONSTRAINT LOGEMENT_PKEY PRIMARY KEY (N_LOG)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE LOGEMENT OWNER TO POSTGRES;
```

```
CREATE TABLE CLIENT
(
  MATRICULE CHARACTER NOT NULL,
  NOM CHARACTER VARYING (20) NOT NULL,
```

```
  PRENOM CHARACTER VARYING (20) NOT NULL,
  SEXE CHARACTER VARYING (1) NOT NULL,
  ADRESSE CHARACTER VARYING (40) NOT NULL,
  AGE FTYPE2 NOT NULL,
  N_TEL CHARACTER VARYING (13) ,
  CONSTRAINT CLIENT_PKEY PRIMARY KEY (MATRICULE)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE CLIENT OWNER TO POSTGRES;
```

CREATE TABLE LOCATION

```
(
N_CONTRAT SERIAL NOT NULL,
DAT_DEBUT DATE NOT NULL,
DAT_FIN DATE NOT NULL,
N_LOG INTEGER NOT NULL,
MATRICULE CHARACTER VARYING (5) NOT NULL,
CONSTRAINT N_LOCATION PRIMARY KEY ("N_LOCATION"),
CONSTRAINT MATRICULE FOREIGN KEY ("MATRICULE")
REFERENCES CLIENT ("MATRICULE") MATCH SIMPLE
ON UPDATE CASCADE ON DELETE CASCADE,
CONSTRAINT N_LOG FOREIGN KEY (N_LOG)
REFERENCES LOGEMENT (N_LOG) MATCH SIMPLE
ON UPDATE CASCADE ON DELETE CASCADE
)
WITH (
  OIDS=FALSE
);
ALTER TABLE LOCATION OWNER TO POSTGRES;
```

4.6.6.4 Implémentation de la base de données

L'implémentation de cette base de données est réalisée sous PostgreSQL. Comme nous avons vu dans les sections précédents, PostgreSQL offre une grande capacité d'extension comme la création des nouveaux types complètement personnalisés, ce qui nous a permis de créer les différents nouveaux types flous avec les fonctions et les procédures nécessaires à leurs manipulation , et ceci d'une manière simple, par contre d'autre SGBD n'ont pas cette capacité d'extension, ce qui oblige à traduire le FSQL en langage SQL compréhensible par le système, cette opération est délicate et nécessite un temps considérable.

a) Script de la BDF en langage SQL

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

```
CREATE TABLE LOGEMENT
(
N_LOG SERIAL NOT NULL,
TYPE CHARACTER VARYING (20) NOT NULL,
NB_PIECE NUMERIC (2) NOT NULL,
ADRESSE CHARACTER VARYING (20) NOT NULL,
PRIXT NUMERIC (1) NOT NULL,
PRIX1 NUMERIC (9,2),
PRIX2 NUMERIC (9,2),
PRIX3 NUMERIC (9,2),
PRIX4 NUMERIC (9,2),
SURFACET NUMERIC (1) NOT NULL,
SURFACE1 NUMERIC (7,2),
SURFACE2 NUMERIC (7,2),
SURFACE3 NUMERIC (7,2),
SURFACE4 NUMERIC (7,2),
ANCIENNETET NUMERIC (1) NOT NULL,
ANCIENNETET1 NUMERIC (2),
```

ANCIENNETET2 NUMERIC (2),

ANCIENNETET3 NUMERIC (2),

ANCIENNETET4 NUMERIC (2),

LOCALT NUMERIC (1) NOT NULL,

LOCALP1 NUMERIC (3,2),

LOCAL1 NUMERIC (2),

LOCALP2 NUMERIC (2),

LOCAL2 NUMERIC (2),

LOCALP3 NUMERIC (2),

LOCAL3 NUMERIC (2),

LOCALP4 NUMERIC (2),

LOCAL4 NUMERIC (2),

LOCALP5 NUMERIC (2),

LOCAL5 NUMERIC (2),

TYPET NUMERIC (1) NOT NULL,

TYPEP1 NUMERIC (3,2),

TYPE1 NUMERIC (2),

TYPEP2 NUMERIC (2),

TYPE2 NUMERIC (2),

TYPEP3 NUMERIC (2),

TYPE3 NUMERIC (2),

TYPEP4 NUMERIC (2),

TYPE4 NUMERIC (2),

TYPEP5 NUMERIC (2),

TYPE5 NUMERIC (2),

ETATT NUMERIC (1) NOT NULL,

ETATP1 NUMERIC (3,2),

ETAT1 NUMERIC (2),

ETATP2 NUMERIC (2),

ETAT2 NUMERIC (2),

ETATP3 NUMERIC (2),

```
ETAT3 NUMERIC (2),
ETATP4 NUMERIC (2),
ETAT4 NUMERIC (2),
CONSTRAINT LOGEMENT_PKEY PRIMARY KEY (N_LOG)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE LOGEMENT OWNER TO POSTGRES;

CREATE TABLE CLIENT
(
MATRICULE CHARACTER NOT NULL,
NOM CHARACTER VARYING (20) NOT NULL,
PRENOM CHARACTER VARYING (20) NOT NULL,
SEXE CHARACTER VARYING (1) NOT NULL,
ADRESSE CHARACTER VARYING (40) NOT NULL,
AGET NUMERIC (1) NOT NULL,
AGE1 NUMERIC (3),
AGE2 NUMERIC (3),
AGE3 NUMERIC (3),
AGE4 NUMERIC (3),
N_TEL CHARACTER VARYING (13) ,
CONSTRAINT CLIENT_PKEY PRIMARY KEY (MATRICULE)
)
WITH (
OIDS=FALSE
);
ALTER TABLE CLIENT OWNER TO POSTGRES;
```

CREATE TABLE LOCATION

(

N_LOCATION SERIAL NOT NULL,

DAT_DEBUT DATE NOT NULL,

DAT_FIN DATE NOT NULL,

N_LOG INTEGER NOT NULL,

MATRICULE CHARACTER VARYING (5) NOT NULL,

CONSTRAINT N_LOCATION PRIMARY KEY ("N_LOCATION"),

CONSTRAINT MATRICULE FOREIGN KEY ("MATRICULE")

REFERENCES CLIENT ("MATRICULE") MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT N_LOG FOREIGN KEY (N_LOG)

REFERENCES LOGEMENT (N_LOG) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE

)

WITH (

OIDS=FALSE

);

ALTER TABLE LOCATION OWNER TO POSTGRES;

Nous pouvons représenter les tables créées comme suivant :

Table LOGEMENT :
Représentation floue :

N_LOG *classique*	TYPE *classique*	NB_PIECE *classique*	ADRESSE *classique*	PRIX *FTYPE2*	SURF_LOG *FTYPE2*	SURF_JAR *FTYPE2*

SURF_GAR *FTYPE2*	ANCIENNETE *FTYPE2*	LOCALISATION *FTYPE3*

Représentation classique :

N_LOG *classique*	TYPE *classique*	NB_PIECE *classique*	ADRESSE *classique*	PRIXP *classique*	PRIX1 *classique*	PRIX2 *classique*	PRIX3 *classique*	PRIX4 *classique*

SURF_LOGP *classique*	SURF_LOG1 *classique*	SURF_LOG2 *classique*	SURF_LOG3 *classique*	SURF_LOG4 *classique*	SURF_JARP *classique*

SURF_JAR1	SURF_JAR2	SURF_JAR3	SURF_JAR4	SURF_GARP	SURF_GAR1
classique	*classique*	*classique*	*classique*	*classique*	*classique*

SURF_GAR2	SURF_GAR3	SURF_GAR4	ANCIENNETEP	ANCIENNETEP1
classique	*classique*	*classique*	*classique*	*classique*

ANCIENNETEP2	ANCIENNETEP3	ANCIENNETEP4	LOCALTT	LOCALP1
classique	*classique*	*classique*	*classique*	*classique*

LOCAL1	LOCALP2	LOCAL2	LOCALP3	LOCAL3	LOCALP4	LOCAL4	TYPETT
classique	*classique*	*classique*	*classique*	*classique*	*classique*	*classique*	*classique*

TYPEP1	TYPE1	TYPEP2	TYPE2	TYPEP3	TYPE3	TYPEP4	TYPE4
classique	*classique*	*classique*	*classique*	*classique*	*classique*	*classique*	*classique*

ETATTT	ETATP1	ETAT1	ETATP2	ETAT2	ETATP3	ETAT3	ETATP4	ETAT4
classique	*classique*	*classique*	*classique*	*classique*	*classique*	*classique*	*classique*	*classique*

Table CLIENT :

Représentation floue :

MATRICULE	NOM	PRENOM	SEXE	ADRESSE	AGE	N_TEL
classique	*classique*	*classique*	*classique*	*classique*	*classique*	*classique*

Représentation classique :

MATRICULE	NOM	PRENOM	SEXE	ADRESSE	AGEP	AGE1
classique	*classique*	*classique*	*classique*	*classique*	*classique*	*classique*

AGE2	AGE3	AGE4	N_TEL
classique	*classique*	*classique*	*classique*

Table LOCATION :

Représentation classique :

N_CONTRAT	N_LOG	MATRICULE	DAT_DEBUT	DAT_FIN
classique	*classique*	*classique*	*classique*	*classique*

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 (LOGEMENT, NB_PIECE,1,1) ;

INSERT into FCL values (LOGEMENT,PRIX,2,1,) ;

INSERT into FCL values (LOGEMENT,SURFACE,2,1,) ;

INSERT into FCL values (LOGEMENT,ANCIENNETE,2,1,) ;

INSERT into FCL values (LOGEMENT,LOCAL,3,1,) ;

INSERT into FCL values (LOGEMENT, ETAT,3,1,) ;

- Définition des valeurs "MARGE" et "MUCH" Types 1et 2

INSERT into FAM values(LOGEMENT, ANCIENNETE,5,10) ;

INSERT into FAM values(LOGEMENT, PRIX, 2000,5000) ;

INSERT into FAM values(LOGEMENT, SURFACE,20,50) ;

c) Définition des étiquettes linguistiques en langage FSQL

Pour définir les étiquette linguistiques, on dispose de deux 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 **LOGEMENT**

CREATE LABEL BAS ON LOGEMENT.PRIX VALUES 6000, 8000, 11000,12000 ;

CREATE LABEL MOYEN ON LOGEMENT.PRIX VALUES 9000,12000,15000,18000 ;

CREATE LABEL ELEVE ON LOGEMENT.PRIX VALUES 20000,30000,50000,70000 ;

CREATE LABEL NOUVEAU ON LOGEMENT.ANCIENNETE VALUES 0,0,5,7 ;

CREATE LABEL MOYEN ON LOGEMENT.ANCIENNETE VALUES 5,6,10,12 ;

CREATE LABEL ANCIEN ON LOGEMENT.ANCIENNETE VALUES 11,15,25,30 ;

*La table **CLIENT***

CREATE LABEL JEUNE ON CLIENT.ÂGE VALUES 18,22,30,35 ;

CREATE LABEL ADULTE ON CLIENT.ÂGE VALUES 25,32,45,50 ;

CREATE LABEL VIEUX ON CLIENT.ÂGE VALUES 50,55,60,70 ;

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 **LOGEMENT**:

INSERT into FOL values(LOGEMENT,PRIX,0,'BAS',0) ;
INSERT into FOL values(LOGEMENT,PRIX,1,'MOYEN',0) ;
INSERT into FOL values(LOGEMENT,PRIX,2,'ELEVE',0) ;
INSERT into FOL values(LOGEMENT, ANCIENNETE,0,'NOUVEAU',0) ;
INSERT into FOL values(LOGEMENT, ANCIENNETE,1,'MOYEN',0) ;
INSERT into FOL values(LOGEMENT, ANCIENNETE,2,'ANCIEN',0) ;

La table **CLIENT**:

INSERT into FOL values(CLIENT,ÂGE,0,'JEUNE',0) ;
INSERT into FOL values(CLIENT,ÂGE,1,'ADULTE',0) ;
INSERT into FOL values(CLIENT,ÂGE,2,'VIEUX',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 :

La table **LOGEMENT**:

INSERT into FLD values(LOGEMENT, PRIX,0,6000,8000,11000,12000) ;
INSERT into FLD values(LOGEMENT, PRIX,1,9000,12000,15000,18000) ;
INSERT into FLD values(LOGEMENT, PRIX,2,20000,30000,50000,70000) ;
INSERT into FLD values(LOGEMENT, ANCIENNETE,0,0,0,5,7) ;
INSERT into FLD values(LOGEMENT, ANCIENNETE,1,5,6,10,12) ;
INSERT into FLD values(LOGEMENT, ANCIENNETE,2,11,15,25,30) ;

La table **CLIENT**:

INSERT into FLD values(CLIENT,ÂGE,0,18,22,30,35) ;

INSERT into FLD values(CLIENT,ÂGE,1,25,32,45,50) ;
INSERT into FLD values(CLIENT,ÂGE,2,50,55,62,70) ;

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

La table **LOGEMENT**:
INSERT into FND values(LOGEMENT,LOCALISATION,0,1,0,7) ;
INSERT into FND values(LOGEMENT, LOCALISATION,0,2, 0,8) ;
INSERT into FND values(LOGEMENT, LOCALISATION,0,3, 0,5) ;
INSERT into FND values(LOGEMENT, LOCALISATION,0,4, 0,5) ;
INSERT into FND values(LOGEMENT, LOCALISATION,1,2, 0,4) ;
INSERT into FND values(LOGEMENT, LOCALISATION,1,3, 0,6) ;
INSERT into FND values(LOGEMENT, LOCALISATION,1,4, 0,8) ;
INSERT into FND values(LOGEMENT, LOCALISATION,2,3, 0,6) ;
INSERT into FND values(LOGEMENT, LOCALISATION,2,4, 0,5) ;
INSERT into FND values(LOGEMENT, LOCALISATION,3,4, 0,8) ;
INSERT into FND values(LOGEMENT,TYP_LOG,0,1,0,7) ;
INSERT into FND values(LOGEMENT, TYP_LOG,0,2, 0,5) ;
INSERT into FND values(LOGEMENT, TYP_LOG,0,3, 0,4) ;
INSERT into FND values(LOGEMENT, TYP_LOG,0,4, 0,1) ;
INSERT into FND values(LOGEMENT, TYP_LOG,1,2, 0,8) ;
INSERT into FND values(LOGEMENT, TYP_LOG,1,3, 0,6) ;
INSERT into FND values(LOGEMENT, TYP_LOG,1,4, 0,2) ;
INSERT into FND values(LOGEMENT, TYP_LOG,2,3, 0,4) ;
INSERT into FND values(LOGEMENT, TYP_LOG,2,4, 0,3) ;
INSERT into FND values(LOGEMENT, TYP_LOG,3,4, 0,4) ;
INSERT into FND values(LOGEMENT, ETAT_AMENAG,0,1,0,8) ;
INSERT into FND values(LOGEMENT, ETAT_AMENAG,0,2, 0,5) ;
INSERT into FND values(LOGEMENT, ETAT_AMENAG,0,3, 0,1) ;
INSERT into FND values(LOGEMENT, ETAT_AMENAG,1,2, 0,7) ;
INSERT into FND values(LOGEMENT, ETAT_AMENAG,1,3, 0,5) ;

INSERT into FND values(LOGEMENT, ETAT_AMENAG,2,3, 0,8) ;

Enfin, notre base de méta connaissances peut être représentée avec les tables remplies suivantes :

Table FOL

OBJECT	COLUMN	FUZZY_ID	FUZZY_NAME	FUZZY_TYPE
LOGEMENT	PRIX	0	BAS	0
LOGEMENT	PRIX	1	MOYEN	0
LOGEMENT	PRIX	2	ELEVE	0
LOGEMENT	ANCIENNETE	0	NOUVEAU	0
LOGEMENT	ANCIENNETE	1	MOYEN	0
LOGEMENT	ANCIENNETE	2	ANCIEN	0
CLIENT	AGE	0	JEUNE	0
CLIENT	AGE	1	ADULTE	0
CLIENT	AGE	2	VIEUX	0

Table FLD

OBJECT	COLUMN	FUZZY_ID	ALPHA	BETTA	GAMMA	DELTA
LOGEMENT	PRIX	0	6000	8000	11000	10000
LOGEMENT	PRIX	1	9000	12000	15000	18000
LOGEMENT	PRIX	2	20000	30000	50000	70000
LOGEMENT	ANCIENNETE	0	0	0	5	7
LOGEMENT	ANCIENNETE	1	5	6	10	12
LOGEMENT	ANCIENNETE	2	11	15	25	30
CLIENT	AGE	0	18	22	30	35
CLIENT	AGE	1	25	32	45	50
CLIENT	AGE	2	50	55	62	70

Table FND

OBJECT	COLUMN	FUZZY_ID1	FUZZY_ID2	DEGREE
LOGEMENT	LOCALISATION	0	1	0,7
LOGEMENT	LOCALISATION	0	2	0 ,8
LOGEMENT	LOCALISATION	0	3	0,5
LOGEMENT	LOCALISATION	0	4	0,5
LOGEMENT	LOCALISATION	1	2	0,4
LOGEMENT	LOCALISATION	1	3	0,6
LOGEMENT	LOCALISATION	1	4	0,8
LOGEMENT	LOCALISATION	2	3	0,6
LOGEMENT	LOCALISATION	2	4	0,5
LOGEMENT	LOCALISATION	3	4	0,8
LOGEMENT	TYP_LOG	0	1	0,7

LOGEMENT	TYP_LOG	0	2	0 ,5
LOGEMENT	TYP_LOG	0	3	0,4
LOGEMENT	TYP_LOG	0	4	0,1
LOGEMENT	TYP_LOG	1	2	0,8
LOGEMENT	TYP_LOG	1	3	0,6
LOGEMENT	TYP_LOG	1	4	0,2
LOGEMENT	TYP_LOG	2	3	0,4
LOGEMENT	TYP_LOG	2	4	0,3
LOGEMENT	TYP_LOG	3	4	0,4
LOGEMENT	ETAT_AMENAG	0	1	0 ,8
LOGEMENT	ETAT_AMENAG	0	2	0,5
LOGEMENT	ETAT_AMENAG	0	3	0,1
LOGEMENT	ETAT_AMENAG	1	2	0,7
LOGEMENT	ETAT_AMENAG	1	3	0,5
LOGEMENT	ETAT_AMENAG	2	3	0,8

4.6.6.5 Interrogation de la base de données

Maintenant que nous avons implémenté notre BDF, et pour bénéficier de son apport en matière d'interrogation flexible, nous présentons quelques exemples de ces requêtes formulées en langage FSQL. Pour ceci, nous prenons comme exemple les requêtes qui peuvent être définis sur la table LOGEMENT de la figure 4.13.

N_LOG	TYPE	NB_PIECE	LOCAL	PRIX	SURFACE	AGE
1	VILLA	5	BORDJ EL KIFAN	[35000,40000]	GRANDE	MOYEN
2	MAISON	6	ELHARRACH	ELEVE	[160,180]	NOUVEAU
3	STUDIO	1	KOUBA	5000	[20,25]	[6,9,12,15]
4	APPARTEMENT	3	MOHAMMADIA	[14000,17000]	75	6±2
7	APPARTEMENT	4	AIGER	17000±3000	80±10	[10,15]
8	STUDIO	1	BEB EZZOUAR	BAS	[25,30]	ANCIEN
9	STUDIO	1	ELHARRACH	6000	25±5	3
10	APPARTEMENT	3	OUED SMAR	[10000,14000]	[65,72]	[2,3]
11	MAISON	6	ALGER	ELEVE	[150,170,190,200]	NOUVEAU
12	VILLA	4	KOUBA	30000	MOYENNE	[4,6]
13	APPARTEMENT	5	BEB EZZOUAR	17000±5000	185±20	[1,3]
14	STUDIO	1	ELHARRACH	[6000,7000]	40	[6,9,12,15]
15	MAISON	4	KOUBA	MOYEN	[110,120]	ANCIEN
16	MAISON	5	BORDJ EL KIFAN	[25000,35000]	GRANDE	MOYEN
17	VILLA	6	ELHARRACH	30000±5000	160±20	[4,6]
18	APPARTEMENT	4	MOHAMMADIA	[16000,19000]	[85,95]	NOUVEAU

Figure 4.13 : Extension de la relation Logement

Requête 1 : Considérons la requête qui consiste à trouver les nouveaux logements (avec un degré ≥ 0.5). La formulation de cette requête en FSQL est comme suit :

SELECT TYPE, NB_PIECE, LOCAL, PRIX, SURFACE, AGE, CDEG (ÂGE)

FROM LOGEMENT WHERE AGE =! 'NOUVEAU' THOLD 0.5

Résultats retournés par la requête 1 : Nous remarquons que les tuples retournés correspondent à des logements qui appartiennent à l'étiquette ''NOUVEAU'' avec un degré >= 0.5 même s'ils appartiennent a d'autres étiquettes, par exemple, il ya des logements qui figurent dans le résultat bien qu'ils appartiennent à l'étiquette MOYEN, mais puisqu'il ya une intersection entre les deux fonctions trapézoïdales des étiquettes MOYEN et NOUVEAU, ils figurent dans ces tuples avec un degré de 0.66.

N_LOG	TYPE	NB_PIECE	LOCAL	PRIX	SURFACE	AGE	CDEG
2	MAISON	6	ELHARRACH	ELEVE	[160,180]	NOUVEAU	1
11	MAISON	6	ALGER	ELEVE	[150,170,190,200]	NOUVEAU	1
12	VILLA	4	KOUBA	30000	MOYENNE	[4,6]	1
17	VILLA	6	ELHARRACH	30000±5000	160±20	[4,6]	1
18	APPARTEMENT	4	MOHAMMADIA	[16000,19000]	[85,95]	NOUVEAU	1
1	VILLA	5	BORDJ EL KIFAN	[35000,40000]	GRANDE	MOYEN	0.66
16	MAISON	5	BORDJ EL KIFAN	[25000,35000]	GRANDE	MOYEN	0.66

Figure 4.14 : Les tuples retournés par la requête 1

Requête 2 : Considérons la requête qui consiste à trouver les nouveaux logements ayant un nombre de chambre supérieur ou égal à 5.

La modélisation de cette requête est comme suit :

SELECT TYPE, NB_PIECE, LOCAL, PRIX, SURFACE, AGE, CDEG (ÂGE) FROM LOGEMENT WHERE AGE =! 'NOUVEAU' THOLD 0.5 AND NB_PIECE >= 5

Résultats retournés par la requête 2 :

N_LOG	TYPE	NB_PIECE	LOCAL	PRIX	SURFACE	AGE	CDEG
2	MAISON	6	ELHARRACH	ELEVE	[160,180]	NOUVEAU	1
11	MAISON	6	ALGER	ELEVE	[150,170,190,200]	NOUVEAU	1
17	VILLA	6	ELHARRACH	30000±5000	160±20	[4,6]	1
1	VILLA	5	BORDJ EL KIFAN	[35000,40000]	GRANDE	MOYEN	0.66
16	MAISON	5	BORDJ EL KIFAN	[25000,35000]	GRANDE	MOYEN	0.66

Figure 4.15 : Les tuples retournés par la requête 2

Cette requête montre que nous pouvons utiliser des comparateurs flous et classiques ensembles.

Requête 3 : Considérons maintenant une requête simple lancée par deux utilisateurs différents *'A.HAMMADI'* et *'B.SALEMI'*, la requête consiste à trouver les logements ayant un prix moyen.

La modélisation de cette requête est comme suit :

SELECT TYPE, NB_PIECE, LOCAL, PRIX, SURFACE, AGE, CDEG (PRIX) FROM LOGEMENT WHERE PRIX =! 'MOYEN'

Résultats retournés par la requête 3 :

Pour l'utilisateur A.HAMMADI

N_LOG	TYPE	NB_PIECE	LOCAL	PRIX	SURFACE	AGE	CDEG
4	APPARTEMENT	3	MOHAMMADIA	[14000,17000]	75	6±2	1
10	APPARTEMENT	3	OUED SMAR	[10000,14000]	[65,72]	[2,3]	1
15	MAISON	4	KOUBA	MOYEN	[110,120]	ANCIEN	1
18	APPARTEMENT	4	MOHAMMADIA	[16000,19000]	[85,95]	NOUVEAU	0.66
8	STUDIO	1	BEB EZZOUAR	BAS	[25,30]	ANCIEN	0.16
7	APPARTEMENT	4	AIGER	17000±3000	80±10	[10,15]	0.12
13	APPARTEMENT	5	BEB EZZOUAR	17000±5000	185±20	[1,3]	0.12

Pour l'utilisateur B.SALEMI

N_LOG	TYPE	NB_PIECE	LOCAL	PRIX	SURFACE	AGE	CDEG
10	APPARTEMENT	3	OUED SMAR	[10000,14000]	[65,72]	[2,3]	1
15	MAISON	4	KOUBA	MOYEN	[110,120]	ANCIEN	1
8	STUDIO	1	BEB EZZOUAR	BAS	[25,30]	ANCIEN	0.66
4	APPARTEMENT	3	MOHAMMADIA	[14000,17000]	75	6±2	0.33
14	STUDIO	1	ELHARRACH	[6000,7000]	40	[6,9,12,15]	0.33

Nous remarquons que les tuples retournés pour chaque utilisateur sont différents:

- Il existe des tuples qui appartiennent au résultat du premier utilisateur et ne figure pas pour le deuxième (logements n° 18, 7, 13), comme il existe des tuples qui vérifient l'inverse (logement n° 14 qui figure seulement dans le resultat2).
- Il existe des tuples qui figurent dans les deux résultats, mais avec un degré différent.

Explication : Cette différence est dû au caractère adaptatif du système qui lui permet de traduire le concept flou *''MOYEN''* par rapport à l'utilisateur et d'avoir des résultats différents.

Requête 4 : Considérons maintenant la requête 3 lancée par un seul utilisateur mais dans deux temps différents, la première fois en mois octobre, et le deuxième en mois de Mars,

Résultats retournés par la requête 4 :

Mois d'Octobre

N_LOG	TYPE	NB_PIECE	LOCAL	PRIX	SURFACE	AGE	CDEG
4	APPARTEMENT	3	MOHAMMADIA	[14000,17000]	75	6±2	1
10	APPARTEMENT	3	OUED SMAR	[10000,14000]	[65,72]	[2,3]	1
15	MAISON	4	KOUBA	MOYEN	[110,120]	ANCIEN	1
18	APPARTEMENT	4	MOHAMMADIA	[16000,19000]	[85,95]	NOUVEAU	0.66

Mois de Mars

N_LOG	TYPE	NB_PIECE	LOCAL	PRIX	SURFACE	AGE	CDEG
4	APPARTEMENT	3	MOHAMMADIA	[14000,17000]	75	6±2	0.87
10	APPARTEMENT	3	OUED SMAR	[10000,14000]	[65,72]	[2,3]	0.87
15	MAISON	4	KOUBA	MOYEN	[110,120]	ANCIEN	0.87
18	APPARTEMENT	4	MOHAMMADIA	[16000,19000]	[85,95]	NOUVEAU	0.66

Nous remarquons que les tuples sont les mêmes dans les deux résultats, mais les degrés de certains tuples ont été changés.

Explication : Cette différence est dû au caractère adaptatif du système par rapport au contexte flou *'temps'* qui permet de prendre en considération l'appartenance partielle d'une date à un contexte flou défini pour le concept flou *''MOYEN''*.

4.7 Conclusion

Nous avons présenté, dans ce chapitre, une extension du modèle de base de données floue GEFRED afin de permettre une représentation multidimensionnelle des concepts flous avec une adaptabilité floue par rapport aux différentes situations. Nous avons appliqué cette extension à un SGBDR libre en prenant le cas d'une gestion d'une agence immobilière.

La base de données implémentée manipule plusieurs concepts flous qui peuvent être utilisée dans les requêtes flexibles d'une manière simple.

Nous avons exploité l'extensibilité des SGBDR open source comme PostgreSQL afin d'optimiser l'implémentation d'une BDF toujours sur la base du modèle théorique GEFRED, avec une autre manière d'implémentation qui évite la traduction manuelle de FSQL vers SQL. Ainsi PostgreSQL est étendu afin de lui permettre de traiter les données imprécises d'une manière simple et efficace.

Conclusion et perspectives

Les informations dont on peut disposer sont souvent incomplètes ou imprécises, ceci ne doit évidemment pas être un obstacle à leur gestion par des systèmes d'informations avancés, les SGBD actuelles doivent être capables de traiter d'une façon efficace ce type de données.

Nous avons présenté dans ce mémoire les approches qui visent à traiter les données imparfaites en utilisant les outils de la logique floue, nous avons constaté que la logique floue ouvre des possibilités remarquables de manipulation des connaissances imprécises et incertaines.

Nous avons étudié le modèle GEFRED avec son langage d'interrogation de donnée FSQL, qui se présente comme un modèle généralisé et synthétique des autres modèles.

Cependant, nous avons constaté que le sens assigné à un terme linguistique est souvent relatif, et dépend des différents contextes d'utilisation. Le figer à un sens unique peut être considéré comme une défaillance dans la représentation des concepts du monde réel, pour pallier à cette limitation , nous avons proposé une extension du modèle qui permet une représentation multidimensionnelle des données imparfaites afin de donner à un concept flou plusieurs significations dépendamment du contexte et du profil de l'utilisateur, Nous avons également proposé une extension du langage d'interrogation des données baptisée AFSQL (Adaptive FSQL) permettant d'adapter les requête floues aux différents contextes d'utilisation.

Nous avons profité de l'extensibilité caractérisant le SGBDR open source PostgreSQL, qui a offert des capacités facilitant et optimisant l'implémentation d'un langage composé comme FSQL, Cette extension dote le SGBD d'une large capacité d'expression des données imprécises, une grande flexibilité dans l'interrogation des données, ainsi qu'une adaptabilité efficace. Le SGBD grâce à l'introduction de la logique floue est devenu capable de manipuler des informations imparfaites de façon simple et efficace.

Ce travail nous a ouvert la porte sur d'autres perspectives pour pouvoir construire et manipuler les profils flous des utilisateurs ainsi que les contextes flous par l'intégration des techniques de Data Mining d'une manière automatique pour faire sortir les dimensions souvent utilisés par les utilisateurs. Il est aussi utile d'introduire les ontologies dans la réprésentation des concepts flous pour utiliser la sémantique dans le processus d'adaptation par rapport aux différentes dimensions.

Bibliographie

[ALA 07] Abdeslame ALILAOUAR . Contribution à l'interrogation flexible de données semi-structurées, Thèse de doctorat, Université de Paul Sabatier, Toulouse, 2007.

[AWI 00] R. Agrawal and E. Wimmers. A Framework for Expressing and Combining Preferences. In Proc. SIGMOD, Texas, USA, pp.297-306. 2000

[BAG 07] Barbet A., Guillard F., Refonte Du Prototype D'interrogation Floue iSQLF, ENSSAT LANNION, 2007

[BAS 06] BARTEL Samuel, Interrogation floue de Bases de données: extension de iSQLf, ENSSAT, LANNION, 2006

[BEL 05] Belohlavek R., & Vychodil V. Functional dependencies of data tables over domains with similarity relations. In Proc. IICAI 2005, Indian International Conference on Artificial Intelligence, Pune, India, ISBN 0–9727412–1–6, pp. 2486–2504. (2005).

[BEL 06a] Belohlavek R., & Vychodil V. Data tables with similarity relations: functional dependencies, complete rules and non-redundant bases. In Lee M. L., Tan K. L., & Wuwongse V. (Eds.), DASFAA 2006,Database Systems for Advanced Applications, LNCS 3882, ISBN 3–540–33337–1, pp. 644–658. (2006a).

[BEL 06b] Belohlavek R., & Vychodil V. Relational Model of Data over Domains with Similarities: An Extension for Similarity Queries and Knowledge Extraction. In IEEE IRI 2006, Information Reuse and Integration, Hawaii, USA, ISBN 0–7803–9788–6, pp. 207–213. (2006b).

[BGP 92] Barbara, D., Garcia-Molina, H., & Porter, D. (1992). The management of probabilistic data. IEEE Transactions on Knowledge and Data Engineering, 4, 487-502

[BMM 03] Bernadette Bouchon-Meunier et Christophe Marsala - logique floue, principes, aide à la décision, traité IC2 : Information commande communication. Traité IC2 : Information commande communication. Hernès-Lavoisier, 2003.

[BOS 94] Bosc, P., Lietard, L. & Pivert, O. Soft querying, a new feature for database management system. Proceedings DEXA'94 (Database and EXpert system Application), Lecture Notes in Computer Science #856 , pp. 631{640. Springer-Verlag. 1994

[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, H. Prade , An introduction to the fuzzy set and possibility theory-based treatment of soft queries and uncertain or imprecise databases,Uncertainty Management in Information Systems: From Needs to Solutions. P.Smets, A.Motro (Eds.), Kluwer, Dordrecht, p. 285-324, 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 04] Bosc P., Liétard L. Pivert P., Rocacher. D, Gradualité et imprécision dans les bases de données. éditions Ellipses, 2004.

[BOS 05a] Bosc P., Liétard L. Pivert P., Rocacher. D, A propos de la division usuelle et approchée de relations floues, Technique et Sciences informatique, à paraître, 2005.

[BOS 05b] Bosc P., Liétard L. Pivert P., Rocacher. D, Characterizing the result of the division of Fuzzy relations, 24th International conference of NAFIPS, the north American fuzzy information processing society, à paraitre, 2005.

[BOS 05c] Bosc P., Liétard L. Pivert P., Rocacher. D, About quotient and division of crisp and fuzzy relations, Journal of intelligent information system,p.4 2005.

[BOS 05d] Bosc, P., Pivert, O.: About projection-selection-join queries addressed to possibilistic relational databases. IEEE Trans. on Fuzzy Systems 13, 124–139 (2005)

[BOS 08] Bosc P., Liétard L. "Aggregates computed over fuzzy sets and their integration into SQLf", in International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems", vol. 16, no 6, p. 761-792, 2008.

[BOS 09] Bosc P., Pivert, O. H Prade A Model Based on Possibilistic Certainty Levels for Incomplete Databases, LNAI 5785, pp. 80–94, 2009.

[BOS 10a] P. Bosc, A. Hadjali, O. Pivert, and G. Smits. CORTEX -- CORrelaTion-based Query EXpansion. In Actes des 25èmes Journées Bases de Données Avancées (BDA'10), Session démonstration, Toulouse, France, 2010.

[BOS 10b] P. Bosc, O. Pivert, A propos de requêtes à préférences et diviseur stratifié, Actes du XXVIII° congrès INFORSID, Marseille, mai 2010.

[BOS 11] Bosc, P.; Pivert, O.; Smits G. On a Fuzzy Group-By and Its Use for Fuzzy Association Rule Mining. In proc. of Advances in Databases and Information Systems,Lecture Notes in Computer Science, 2011.

[BUC 06] P. Buche, Intégration de données hétérogènes, imprécises et incomplètes: Application dans le domaine du risque alimentaire, HDR, Université Paris Sud – Orsay, 2006.

[BUP 82] Buckles, B. P., & Petry, F. E. (1982). A fuzzy representation of data for relational databases. Fuzzy Sets Systems, 7, 213-226.

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

[CHO 03] Chomicki, J. Preference formulas in relations queries. ACM Transactions on Database Systems, 28:1–39. 2003

[COD 70] Codd, E. F. 1970, A relational model of data for large shared data banks, Communications of The ACM, 13 (6): 377-387.

[COD 79] Codd, E. Extending the database relational model to capture more meaning. ACM Transactions on Database Systems, 4:156{174. 1979
[DEA 07] A. Denguir-Rekik, Un Cadre Possibiliste pour l'Aide à la Décision Multicritère et Multi-acteurs Application au Marketing et au Benchmarking de sites E-commerce, These De Doctorat, université de Savoie, 2007

[DET 02] de Tré, G. . Extended possibilistic truth values. International Journal of Intelligent Systems, 17, 427-446. 2002

[DET 03] de Tré, G., & de Caluwe, R.. Modelling Modelling uncertainty in multimedia database systems: An extended possibilistic approach. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 11(1), 5-22.2003

[DET 04] de Tre, G., de Caluwe, R. et Prade, H. Null values in prospect of data integration. In Bouzeghoub, M., Goble, C., Kashyap, V. et Spaccapietra, S., editeurs : Semantics of a Networked World, semantics for grid databases,volume LNCS 3226. IC5NW, Springer. 2004

[DUP 94] Dubois, d. & Prade, H.. Logique Floue. Série Arago. Observatoire francais des techniques avancées. (1994).

[EUV 98] Virginie EUDE, . Modélisation spatio-temporelle floue pour la reconnaissance d'activités militaries, , Thèse de doctorat, Université de Paris 6, Paris, 1998.

[GAL 99] 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 05] Galindo J. "New Characteristics in FSQL: A Fuzzy SQL for Fuzzy Databases, Computer Journal of WSEAS Transactions on Information Science and Applications, vol. 2, no. 2, pp. 161' 169, 2005.

[GAL 06] Galindo J.,Urrutia A., Piattini M., Fuzzy Databases: Modeling, Design and Implementation". Idea Group Publishing Hershey, USA, 2006.

[GAL 07] Galindo,J. FSQL (fuzzy SQL): A fuzzy query language http://www.lcc.uma.es/ ~ppgg/FSQL, (2007)

[GAL 08a] José Galindo, introduction and Trends to Fuzzy Logic and Fuzzy Databases, IGI Global, 2008.

[GAL 08b] José Galindo, Handbook of Research on Fuzzy Information Processing in Databases, IGI Global, 2008.

[GES 91] Gessert, G.. Handling missing data by using stored truth values.ACM SIGMOD Record, 20:30{42. (1991)

[GRA 80] Grant, J. (1980). Incomplete information in a relational database. Fundamental Informaticae, 3, 363-378.

[GTB 06] Grissa Tousi A., Ben Hassine A., and Ounelli H., "Extended_FSQL_Server: A Server for the Description and the Manipulation of FRDB," in Proceedings of the 4th International Multi Conference on Computer Science and Information Technology (CSIT), Jordan, pp. 454' 464, 2006.

[GTB 09] Grissa Tousi A., Ben Hassine A., New Architecture of Fuzzy Database Management Systems , The International Arab Journal of Information Technology, Vol. 6, No. 3, July 2009

[GYP 99] Glorennec Pierre Yves. - Algorithmes d'apprentissage pour systèmes d'inférence Floue., Hermès , 1999.

[HKP 08] Hadjali A., Kaci S., Prade H., « Database preference queries—A possibilistic logic approach with symbolic priorities », Proc. of the 5th Symposium on the Foundations of Information and Knowledge Systems, p. 291-310,2008.

[HMA 11] H.Martin, le language SQL, Université Pierre-Mendès France,Grenoble 2, http://membres-liglab.imag.fr/Herve.Martin/HTML/SQL.htm , 2011

[HOW 11] Howe, D. Free On-Line Dictionary Of Computing. London, UK, ImperialCollege Department of Computing. Année d'accès: 2011. Url du document: http://foldoc.org/context.

[HUM 09] Miroslav Hudec, An Approach to Fuzzy Database Querying, Analysis and Realisation, ComSIS Vol. 6, No. 2, December 2009

[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.

[JCL 05] L.JIMÉNEZ CANDIA, Gestion des connaissances imparfaites dans les organisations industrielles : cas d'une industrie manufacturière en Amérique Latine, Thèse de doctorat, ENSIACET, Toulouse cedex 4 2005.

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

[KIE 02] W. Kiessling. Foundations of Preferences in Database Systems. Proc. Very Large Data Bases, 2002.

[KIK 02] Kießling W., Köstler G., 2002. Preference SQL – Design, implementation, experiences, 28th Very Large Ddata Bases Conference,p.990-1001, 2002

[KOL 04] Koutrika, G. and Ioannidis, Y. E. (2004). Personalization of queries in database systems. In ICDE, pages 597–608.

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

[LIP 79] Lipski, W. (1979). On semantic issues connected with incomplete information databases. ACM Transactions on Database Systems, 4, 262-296).

[LIR 09a] Liétard L. Rocacher R. "On the extension of SQL to Fuzzy Bipolar Conditions". In proc. of the North American Fuzzy Information Processing Society, Cincinnati, Ohio, USA, 2009

[LIR 09b] Liétard L. Rocacher R. "On the Definition of Extended Norms and Co-norms to Aggregate Fuzzy Bipolar Conditions". In Proc. of the International Fuzzy Systems Association – European Society fo Fuzzy Logic and Technology, Lisbon, Portugal, 2009

[LOI 04] LOISEAU Y. Recherche flexible d'information par filtrage flou qualitatif, Thèse de doctorat, Université de Paul Sabatier, Toulouse, 2004.

[LON 11] Dictionnaire en ligne Longman http://www.ldoceonline.com/dictionary/context , 2011.

[MAB 05] M.A. Ben Hassine , Contribution à l'implémentation d'une base de données floue sous un système de gestion de base de données relationnel, mémoire de mastère, ENI, Tunis, 2005

[MAR 05] A. Martin, « La fusion d'informations », ENSIETA - Réf. : 1484, Janvier 2005.

[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 95a] 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.

[MED 95b] J.M. Medina, M.A. Vila ,J.C Cubero, O. Pons, , "Towards the implementation of a generalized fuzzy relational databases". Fuzzy sets and systems , 75 273-289 , 1995.

[MGT 10] T. Matthé ,G. De Tré The Bipolar Semantics of Querying Null Values in Regular and Fuzzy Databases Dealing with Inapplicability, CIS 81, pp. 137–146, 2010.

[MOU 07] H Mouloudi. Personnalisation de requêtes et visualisations OLAP sous contraintes. Thèse de doctorat, Université Franois Rabelais, Tour, France, 2007.

[MOT 89] Motro A., A trio of database user interfaces for handling vague retrieval requests, , 12, p. 54-63. 1989

[PCK 03] S. Prabhakar R. Cheng, D. Kalashniov. Evaluating probabilistic queries over imprecise data. In Proceedings of the ACM SIGMOD, 2003.

[PET 96] Petry, F. E. . Fuzzy databases: Principles and applications. Boston: Kluwer Academic Publishers, (1996)

[PGS 08] Documentation PostgreSQL 8.3.5 The PostgreSQL Global Development Group Copyright© 1996-2008 The PostgreSQL Global Development Group, 2008

[PRT 84] Prade, H., & Testemale, C. (1984). Generalizing database relational algebra for the treatment of incomplete/uncertain information and vague queries. Information Sciences, 34, 115-143.

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

[RAG 03] N. Ragot , MELIDIS :Reconnaissance de formes par modélisation mixte intrinsèque discriminante à base de systèmes d'inférence floue hiérarchisés, thèse Université de Rennes 1, 2003 .

[RSG 05] Robert Ross, V. S. Subrahmanian, and John Grant. Aggregate operators in probabilistic databases. J. ACM, 52(1):54–101, 2005.

[RZK 06] Antonio Rosado,Rita A. Ribeiro, Slawomir Zadrozny, Janusz Kacprzyk, Flexible Query Languages for Relational Databases: An Overview, Springer-Verlag Berlin Heidelberg 2006

[SAM 07] Sakli MOUADH, commande numérique et notions de régulation de processus projets de fin d' etudes, Ecole Nationale d' Ingénieurs de GABES, TUNISIE, 2007

[SKT 08] Srdjan Skrbic, Aleksandar Takači, On Development of Fuzzy Relational Database Applications, Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU 2008), Malaga, Spain, pp. 268-273, 2008

[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.

[TAL 08] Aleksandar Takači, Priority, Weight and Threshold in Fuzzy SQL Systems, Acta Polytechnica Hungarica , Vol. 5, No. 1, 2008

[TAC 93] Takahashi, Y.. Fuzzy database query languages and their relational completeness theorem. IEEE Transactions on Knowledge and Data Engineering, 5, 122-125. 1993

[THO 03] R. THOMOPOULOS, Représentation et interrogation élargie de données imprécises et faiblement structurées, thèse U.E.R d'informatique, 2003

[TRC 06] G. de Tre and R. de Caluwe: On the Applicability of Extended Possibilistic Truth Values in Flexible Database Modelling and Querying, Studfuzz 203, 75–90 ,2006

[UMA 82] Umano, M. (1982). Freedom-O: A fuzzy database system. In M. Gupta & E. Sanchez (Eds.), Fuzzy information and decision processes (pp. 339-347). North-Holland, Amsterdam.

[UMA 94] Umano, M., & Fukami, S. Fuzzy relational algebra for possibility-distribution-fuzzy-relation model of fuzzy data. Journal of Intelligent Information Systems, 3, 7-28. 1994.

[VIA 06] Viappiani, P., Faltings, B.,and Pu, P. Preference-based search using example-critiquing with suggestions. Journal of Artificial Intelligence Research, 27. 2006

[YAW 05] Yager R. R., Walker C L., Walker E. A., Generalizing leximin to t-norms and t-conorms: the lexiT and lexiS orderings,)X, 151, p. 327-340. 2005.

[ZAD 65] Zadeh, L. A., Fuzzy sets, Information & Control, 8 (3): 338-353, 1965

[ZAD 78] Zadeh L.A. (1978) Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems, 1 pp. 3-28.

[ZAD 05] Zadeh L.A, What is fuzzy logic and what are its applications? http://www.eecs.berkeley.edu/IPRO/Summary/03abstracts/zadeh.13.html , 2005.

[ZEK 84] Zemankova-Leech, M., & Kandel, A. (1984). Fuzzy relational databases: A key to expert systems. Köln, Germany: Verlag TUV Rheinland.

[ZMA 06] Z. Ma: Fuzzy Database Modeling of Imprecise and Uncertain Engineering Information, Springer-Verlag Berlin Heidelberg 2006

[ZML 08] Z. M. MA AND LI YAN, A Literature Overview of Fuzzy Database Models, journal of information science and engineering 24, 189-202 , 2008