République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri de Tizi-Ouzou

Faculté du Génie Electrique et Informatique
Département d'Informatique

Mémoire en vue de l'obtention du Diplôme de
Master en Informatique

Option : Ingénierie des Systèmes d'Information

Thème

Contribution à l'Etude de Requêtes Flexibles

Approche basée sur la Théorie des Sous- Ensembles Flous

Préparé par
MEZIANI Hacene et **ISSOLAH Samir**

Dirigé par

Mlle YESLI Yasmine

Année Universitaire : 2011/ 2012

Dédicaces

Je dédie ce modeste travail

A mes chers parents, pour leur amour,

leurs encouragements, leur patience et leurs sacrifices.

A mon frére Mouhamed, et sa femme Assia

A mes très chères sœurs : Lilia,Samia,Nessrine,Souad,Amel
Ainsi qu'a Mes nièces et mes neveux. :

Amine , Malak ,Aya,Hani,Aymane
A ma grand-mère

A toute ma famille.

A mon binôme Hacene et toute sa famille.

A mes amis (es).
A tous ceux que j'aime Et qui m'aiment…
Où qu'ils soient

Samir

Dédicaces

Je dédie ce modeste travail en signe de respect, d'amour et de reconnaissance à :

Ma Grand-mère

Ma mère et mon père

Mes sœurs : Sonia et Mellissa.

Mes frères : Sofiane, Toufik, Koceila et Ghilas

A mon camarade Samir et toute sa famille

A notre promotrice Yesly Yasmine

Et toute sa famille

Comme je dédie ce travail à mes amis, Mes enseignants

Et à tous ceux qui nous' ont encouragé à réaliser ce modeste travaille.

Hacene

Remerciements

Dieu merci d'avoir été toujours et d'être là, prés de nous ; nous éclaircir la vue, nous remplir le cœur de foi, de courage et de patience.

On ne remerciera jamais assez nos parents, sans lesquels, rien n'aurait été possible. Leur amour et leur dévotion, ont fait de nous ce que nous sommes. Ils nous ont vus naître puis grandir, et n'ont jamais rechigné à nous apporter tout ce qu'ils pouvaient, de nous supporter et de nous réconforter

Nous tenons à adresser nos plus sincères remerciements à notre promotrice Mlle YESLI Yasmine qui a su nous conseiller par ses critiques constructives et nous guider dans notre travail.

Nos remerciements s'adressent à tous les membres du jury pour l'honneur qu'ils nous fait en acceptant de juger notre travail. Nos remerciements vont aussi à tous ceux et celles qui ont participé de près ou de loin à l'élaboration du présent travail. Nos sentiments de gratitude vont à nos professeurs qui tout au long de notre cursus nous ont transmit leur savoir sans réserve.

Enfin, nous tenons à remercier tous nos amis et collègues pour leur soutien pendant la préparation de ce mémoire.

Samir & Hacene

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

- **Contexte et Problématique**

Notre mémoire s'inscrit dans le contexte de l'interrogation flexible de bases de données Relationnelles. L'objectif est de permettre à des utilisateurs de formuler des requêtes comportant des termes imprécis sur une base de données relationnelle classique. Tous les SGBD Relationnels supportent des requêtes dont les conditions sont exprimés par des prédicats booléens, d'où la nomination de SGBDR booléens. Pour surmonter les limites des conditions booléennes, l'interrogation flexible a été proposée comme un enjeu scientifique dans le but de développer de futurs systèmes de gestion de bases de données couvrant de nouveaux besoins d'utilisateurs. A savoir, permettre aux utilisateurs de formuler des requêtes comportant leurs préférences et exprimés via des termes vagues et imprécis du langage naturel. Par exemple *" Quels sont les étudiants ayant une bonne note ? ", Quels sont les hôtels les plus proches à l'aéroport ? ..etc.*

- **Résumé de la contribution**

Notre mémoire contribue à **l'expression des préférences dans les requêtes** en faisant appel à des **conditions graduelles** (ou flexibles) modélisées en utilisant les ensembles flous. Notons que la théorie des ensembles flous [ZAD 65] n'est pas le seul moyen pour exprimer les préférences, cependant elle reste l'outil le plus adapté. On peut résumer l'idée de notre contribution en deux points :

1 – faire une étude bibliographique sur les requêtes flexibles basées sur la théorie des ensembles flous,

2 - Appliquer l'étude théorique sur un cas réel. Notre choix s'est orienté vers la réalisation d'un système d'interrogation flexible en ligne où l'introduction du caractère flou semble pertinente pour répondre aux mieux aux besoins des utilisateurs ; entre autre : un système de

réservation d'hôtels en ligne. La suite du document portera sur tous les pas entrepris pour atteindre cet objectif.

- **Structure du mémoire**

Nous avons structuré notre mémoire en cinq chapitres, les trois premiers forment la partie état de l'art sur l'interrogation flexible de base de données. Nous avons commencé par rappeler quelques concepts de bases de données relationnelles, puis critiquer le principe d'interrogation booléenne rigide sur lesquelles elles s'appuient. Ensuite, un petit rappel sur la théorie des ensembles flous était nécessaire pour mieux appréhender le principe de l'interrogation flexible basée sur le flou. La seconde partie constitue l'application de l'étude théorique menée sur un cas d'étude réel. Nous terminons notre mémoire par une conclusion et quelques perspectives envisagées pour de futurs travaux. Deux annexes complètent notre mémoire. L'annexe A porte sur un résumé du langage UML et l'annexe B porte sur quelques formules et notions mathématiques utiles pour le développement de notre système.

Partie I

Interrogation Flexible de Base de Données Relationnelles

Chapitre 1

Les Base de Données Relationnelles

1.1 Introduction

Le premier chapitre de notre travail a pour objectif de définir le contexte dans le lequel nous nous situons, à savoir l'interrogation de bases de données dans un contexte relationnel. Nous allons commencer par y rappeler les concepts fondamentaux, notamment le modèle de données fondé sur la notion de relation et les opérations algébriques qui constituent la base de la manipulation de celles-ci. Un bref aperçu sur le langage SQL sera présenté et nous terminons par mettre l'accent sur la notion de requête relationnelle *booléenne* et les limites de ce type d'interrogation.

1.2 Model relationnel

Le modèle de données relationnel [COD 70] formalisé par Codd en 1970 repose sur la théorie mathématique des relations. L'idée est de considérer une base de données comme un ensemble de relations dont chacune décrit une partie de l'univers concerné (personnel d'une entreprise, bibliothèque en ligne, ..etc). Dans ce cadre, *une relation r* correspond à un sous ensemble du produit cartésien de *domaines de valeurs* non nécessairement distincts *Di*, $i \in [1,n]$, donc $r \subseteq D_1 \times D_2 \times \ldots \times D_n$. Ce sous ensemble est appelé *extension* de la relation *r*.

En pratique, une relation d'une base de données rassemble les associations de valeurs (appelés *n-uplets* ou *tuples*) décrivant les éléments de la réalité considérée. Le schéma (ou *intension*) *R* d'une relation notée *r(R)* est défini sur un ensemble d*'attributs* (ou champs) *Aj* distincts associés à chacun des domaines. On note $R(A_1, A_2, \ldots A_n)$ le schéma de la relation *r*.

Le nombre d'attributs d'une relation est son *degré* (ou *arité*) et son nombre de n-uplets est appelé *cardinalité*. Toute relation comporte au mois un *identifiant* (ou *clé*), c'est-à-dire un ensemble minimal d'attributs permettant d'identifier de manière unique chacun de ses n-uplets. Un ensemble d'attributs constituant une clé d'une relation *r* et apparaissant dans une autre relation *s* est appelée *clé étrangère* de *s*. Usuellement, une relation est représentée par un tableau bi-dimensionnel (ou table) dont les colonnes sont les attributs et les lignes sont les n-uplets de l'extension considérée. L'ordre des colonnes, de même que celui des lignes, importe peu puisqu'on représente un ensemble d'attributs et un ensemble de n-uplets et que la notion d'ensemble ne véhicule aucun ordre sur les éléments. [BOS 04]

Exemple 1.1 : exemple d'extension (à 4 n-uplets) de la relation *EMPLOYE* dont le schéma est : *EMPLOYE (Num , Nom, Salaire, Age, Adresse).*

Num	Nom	Salaire	Age	Adresse
17	Rami	35000	42	Alger
76	Maria	30000	27	Tipaza
26	Ryad	25000	37	Alger
12	Myriam	25000	39	Alger

1.3 Langages de manipulation des données relationnelles

1.3.1 L'algèbre relationnelle

L'algèbre relationnelle est le langage de base de la manipulation des relations. Elle se compose d'un ensemble d'opérateurs qui, à partir d'une ou deux relations existantes, créent en résultat une nouvelle relation temporaire. La relation résultat a exactement les mêmes caractéristiques qu'une relation de la base de données et peut donc être manipulée de nouveau par les opérateurs de l'algèbre [COD70].

On distingue deux catégories d'opérateurs : les opérateurs *ensemblistes* et les opérateurs *relationnels.*

1.3.1.1 Opérateurs ensemblistes

L'*union*, l'*intersection* et la *différence* de deux relations *r et s* (dont les attributs sont deux à deux compatibles) délivrent des relations de schéma identique à celui des relations d'entrées (*r et s*) et sont définies par [BOS 04] :

- *union (r, s)* = $\{t, t \in r$ ***ou*** $t \in s\}$
- *inter (r, s)* = $\{t, t \in r$ ***et*** $t \in s\}$
- *differ (r, s)* = $\{t, t \in r$ ***et*** $t \notin s\}$

L*e produit cartésien* de deux relations *r* et *s* de schémas quelconques *R(X) et S(Y)* avec $X \cap Y = \emptyset$, retourne une relation de schéma $T(X \cup Y)$ définie par :

- *prod-cart(r,s)* = $\{t=uv \mid u \in r$ *et* $v \in s\}$ où *uv* désigne la concaténation des n-uplets *u et v.*

1.3.1.2 Opérateurs relationnels

Les quatre opérateurs relationnels usuels sont : *la sélection*, *la projection*, *la thêta-jointure* et *la division* [BOS 04] :

La *sélection* est un opérateur unaire qui permet d'extraire une sous-relation d'une relation en imposant une condition (prédicat ou expression logique de prédicats) sur les n-uplets de la relation d'entrée. Formellement, si r est une relation de schéma $R(X)$ et p est un prédicat référençant les attributs de X, on a :

- $select\ (r, s) = \{t \mid t \in r\ \textbf{et}\ p\ (t)\}$

Le second opérateur unaire est la *projection* qui a pour but de ne conserver que les attributs mentionnés en paramètre. Si r est une relation de schéma $R(X)$ et $Z \subset X$, la projection de r sur Z est définie par :

- $project\ (r, Z) = \{z \mid \exists\ t,\ t \in r\ \textbf{et}\ t.\ Z = z\}$ où $t.\ Z$ dénote la valeur de l'attribut ou ensemble d'attributs Z du n-uplet t.

Les deux opérateurs restants sont binaires et correspondent à des combinaisons spécifiques d'opérateurs vus précédemment. La *théta- jointure* dérive du produit cartésien mais s'en distingue en appariant que des couples de n-uplets vérifiant une propriété (de jointure) selon la formule :

- $join\ (r, s, \theta, A, B) = \{ uv \mid u \in r\ \textbf{et}\ v \in s\ \textbf{et}\ \theta\ (u.A, v.B) \}$
  $= select\ \{ prod\text{-}cart\ (r, s), \theta\ (A, B) \}$

où R (resp. S) le schéma de r (resp. s) est défini sur X (resp. Y), A et B désignent deux sous-ensembles respectifs de X et Y qui sont compatibles et θ est un comparateur.
Plusieurs variantes de *théta- jointure* existent, la plus courante est l'*équi-jointure* dont le comparateur θ est *l'égalité.*

Le dernier opérateur est *la division.* Si on considère les schémas *R(X, A)* et *S (Y, B) où A* et *B* deux ensembles d'attributs compatibles, on définie la division de *r(R)* par *s(S)* de la façon suivante :

- $div\ (r, s, A, B) = \{ x \in project\ (r, X)\ \mid \forall a, (a \in s) \Rightarrow (<a, x> \in r) \}$

En d'autres termes, une valeur *x* appartient au résultat de la division si elle est associée dans la relation *r* à (au moins) toutes les valeurs de l'ensemble d'attributs *B* de la relation *s.*

Exemple 1.2 :

Soit la requête *"Trouver les employés ayant un salaire supérieur à 45 000 dinars et habitants à Tizi-Ouzou"* exprimée en algèbre relationnelle comme suit:

select *(EMPLOYE , salaire* $\geq$ *45 000* ***and*** *adresse ='TIZI-OUZOU')*

1.3.2 Le langage SQL

Le langage "Structured Query Language "(SQL) a été introduit dans les années 70 par IBM. Relativement simple par sa syntaxe, il se voit comme un langage proche du langage naturel. SQL a fait l'objet de plusieurs efforts de normalisation qui ont débuté en 1989 et qui poursuivent toujours. Il y a plusieurs normes SQL, chacune enrichissant la version précédente : SQL-89, SQL-92, SQL-93.

On trouve trois facettes de ce langage quelque soit le SGBD relationnel :

- **LDD** (**Langage de Définition de Données**), pour la définition des tables, des des index, des vues relationnelles.
- **LMD** (**Langage de Manipulation de Données**) pour la manipulation des tables et plus précisément les manipulations des tuples de relations.
- **LCD** (**Langage de Control de Données**) permet d'autoriser ou d'interdire l'accès à certaines données à certaines personnes.

Soit le format de base d'une requête SQL :

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

Exemple 1.3 :

La requête précédente de l'exemple 1.2 est modélisée en langage SQL comme suit :

***select** * **from** EMPLOYÉ*

***where** salaire > 45000 **and** adresse = 'TIZI-OUZOU'*

1.3.3 Transcription des opérateurs algébriques dans SQL

Le tableau ci-après présente la transcription des opérateurs algébriques en langage SQL. On constate que l'opérateur de division est le seul opérateur qui n'a pas été implémenté au sein du standard SQL.

Sachant que *r* et *s* sont deux tables de la base de données, les règles de transcriptions se résument ainsi :

Opération algébrique	**Expression dans SQL**
Projection	`SELECT [DISTINCT] *` `FROM r`

Sélection	`SELECT *`
	`FROM r`
	`WHERE condition`

Jointure	`SELECT *` `FROM r JOIN s ON r.attribut = s.attribut`
Division	?

Union	`SELECT * FROM r`
	`UNION`
	`SELECT * FROM s`

Intersection	`SELECT * FROM r`
	`INTERSECT`
	`SELECT * FROM s`

Différence	`SELECT * FROM r`
	`EXCEPT`
	`SELECT * FROM s`

Produit-cartésien	`SELECT * FROM r, s`

Tableau 1.1 : Transcription des opérations algébriques en langage SQL [BRO 04]

1.4 Notion de requête booléenne

Une requête relationnelle est une requête dite de type booléen. Les conditions ou les critères de sélection dans ces dernières sont exprimés via des prédicats booléens. Ainsi,

seulement les tuples satisfaisant ces conditions sont retournées. Tous les systèmes de gestion de base de données relationnels existants sur le marché sont des systèmes booléens.

1.5 Limites de l'interrogation relationnelle

Les utilisateurs n'expriment pas leur besoins avec des conditions booléennes, mais au contraire graduellement avec des termes vagues et imprécis. Ils peuvent, par exemple, exprimer les requêtes suivantes :

R1 : « *quels sont les étudiants ayant une* ***bonne*** *note* »,

R2 : « *quels sont les employés* ***jeunes*** *et* ***bien payés*** *de la wilaya de Tizi-Ouzou* »,

ou encore chercher à trouver dans la requête

R3 : « *les employés* ***jeunes*** *ayant un salaire* ***près de*** *75000 dinars* »,

ou R4 : « *trouver l'hôtel le* ***plus proche*** *de l'aéroport et* ***pas trop cher*** » ,

..etc.

Les SGBD Relationnels ne comprennent pas l'imprécision dans les données. Alors pour exprimer ces requêtes dont les attributs sont vagues et imprécis, on devrait les traduire dans une logique booléenne. Pa exemple, les critères « *jeune* » et « *bien payé* » de R2 sont traduits dans une requête SQL de la manière suivante :

select * ***from*** *EMPLOYE*

where *âge <= 35* ***and*** *age = > 16* ***and*** *salaire > 30 000* ***and*** *adresse = 'TIZI-OUZOU'*

La conversion que l'on pourrait apporter aux attributs ne traduit pas forcément la réalité. Exprimer « jeune » par l'intervalle [16 35] est subjectif. L'interprétation de « jeune » peut différer d'une personne à une autre. Donc une petite variation dans les données pourrait conduire à des résultats différents.

De plus, les requêtes booléennes sont à caractère rigide. On risque dans certains cas d'avoir des réponses vides. Une nouvelle génération de systèmes flexibles vient pallier cette rigidité en proposant d'introduire les préférences utilisateurs dans les requêtes. Divers approches ont été avancées dans la littérature [BOS 04]. Nous nous intéressons dans la suite, en particulier, à l'étude de l'approche basée sur la théorie des sous-ensembles flous.

1.6 Conclusion

Nous avons présenté dans ce premier chapitre le principe de l'interrogation relationnelle. Celui- ci est basé sur une logique booléenne très restrictive. Suite à une requête, un objet n'est retourné que s'il satisfait les conditions de la recherche. Par conséquent, des réponses qui peuvent satisfaire l'utilisateur peuvent être rejetées.

Dans le but de satisfaire les désirs des utilisateurs, plusieurs recherches ont porté sur l'introduction de la flexibilité dans les requêtes. L'approche basée sur les ensembles flous constitue le cœur de notre étude. Pour assurer une meilleure compréhension pour celles- ci, nous avons estimé de consacrer le deuxième chapitre pour rappeler les concepts de base relatifs à la théorie des ensembles flous.

Chapitre 2

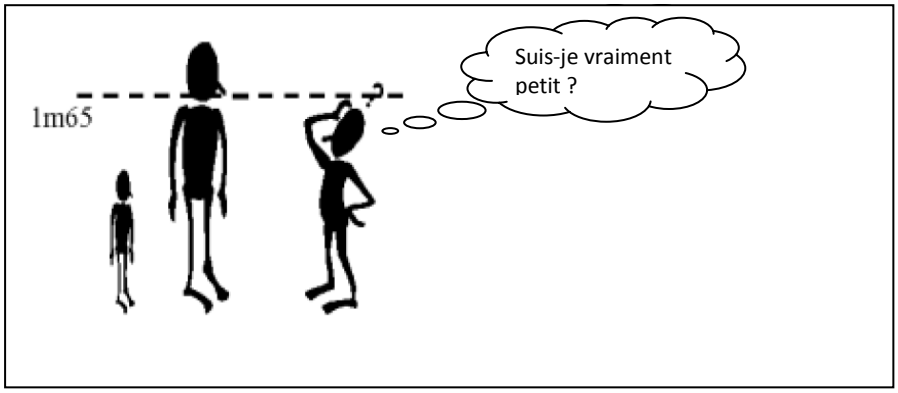
Théorie des Sous- ensembles Flous

2.1 Introduction

Le second chapitre de notre travail se limite à rappeler les concepts de base relatifs à la théorie des ensembles flous et qui sont essentiels à la compréhension des points développés ultérieurement. Nous verrons ce qu'est la logique floue, son apparition et l'objet de son utilisation. Nous présenterons également un ensemble de définitions et de notations et nous terminerons par souligner l'apport de cette dernière (théorie des ensembles flous) dans le contexte des bases de données.

2.2 Logique floue

Une des caractéristiques du raisonnement humain est qu'il est basé sur des données imprécises ou incomplètes [GAB 01]. Ainsi, déterminer si une personne est de petite ou de grande taille est aisé pour n'importe qui d'entre nous, et cela sans nécessairement connaître sa taille. Cependant, un ordinateur, est basé sur des données exactes. Il doit non seulement connaître la taille exacte de la personne mais également posséder un algorithme qui divise une population en deux groupes biens distincts : les grands et les petits. Supposons que la limite soit de 1m65. Je mesure 1m63, suis-je vraiment petit ?



L'idée de la logique floue est de transmettre cette richesse du raisonnement humain à un ordinateur. En effet, c'est une logique qui ne se limite plus à deux valeurs de vérité mais à une infinité de valeurs dans l'intervalle [0,1], ce qui permet de traduire des affirmations du type *un peu vrai, très faux, autant vrai que faux*. En effet, un objet peut appartenir à un ensemble et en même temps à son complément. Ainsi, un individu de 1m63 pourra être à la fois grand et petit [HAS 05].

Le concept du « *flou* » a été introduit pour la première fois par L.A Zadeh en 1965 [ZAD 65] afin de représenter les ensembles ou les classes d'objets aux limites indéterminées (tels que "région pluvieuse", "centre ville" ..etc), dans lesquelles la transition d'appartenance à non-appartenance n'est pas brusque, mais plutôt graduelle. Cela est particulièrement le cas pour de nombreux termes en langage naturel (tels que "grand", "jeune",..etc) qui se représentent mal par des ensembles classiques. (voir la Figure 2.1)

La gradualité a permis de définir les nuances d'appartenance (*appartient peu, appartient beaucoup, appartient plus au moins*..), d'expliciter les choix, de modéliser le langage naturel,.. etc [BOS 04].

2.3 Les ensembles flous

2.3.1 Définition

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

$$\mu_F : \quad X \longrightarrow [0, 1]$$
$$x \longmapsto \mu_F(x)$$

La valeur $(\mu_F(x))$ représente le degré d'appartenance de x à l'ensemble F. Par définition, si $(\mu_F(x) = 0)$ alors x n'appartient pas du tout à F et plus $(\mu_F(x))$ se rapproche de 1, plus la valeur de x appartient à F , si $((\mu_F(x) = 1)$ alors x appartient complètement à F. [BOS 04]

On représente l'ensemble flou F comme suit :

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

Exemple 2.1 :

Les ensembles flous sont bien adaptés pour représenter les termes du langage naturel. Prenons par exemple le terme « *grand* ».

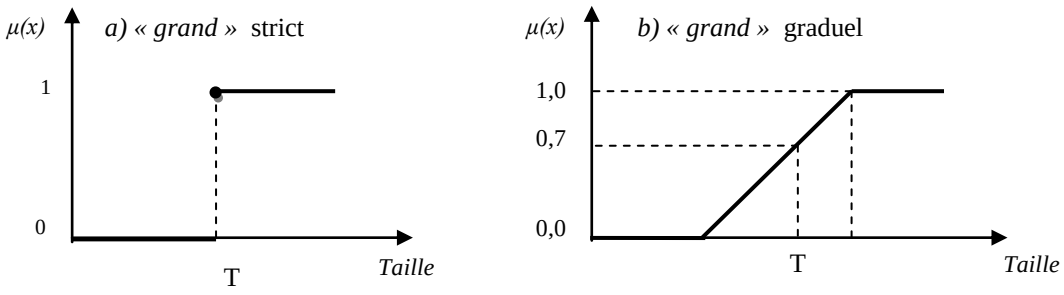


Figure 2.1 : Représentation stricte et représentation graduelle

Le terme graduel « *grand* » de la Figure 2.1 (b) est défini avec une transition graduelle entre l'état « non grand » et l'état « grand ». Il n'existe donc pas la transition stricte de la Figure 2.1(a) qui indique que l'on devient brusquement grand à partir d'une certaine valeur T.

2.3.2 Caractéristiques

Un sous- ensemble flou *F* de l'univers *X* est caractérisé par [BOS 04] :

Son noyau, noté *noy(F),* qui représente l'ensemble des éléments de *X* pour lesquels la fonction d'appartenance $\mu_F(x)$ vaut 1 :

- $noy(F) = \{ x \in X / \mu_F(x) = 1 \}$

Son support, noté *supp(F),* qui représente l'ensemble des éléments de *X* appartenant, quelque peu, à *F*, c-à-d, ayant $\mu_F(x)$ qui n'est pas nulle :

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

Sa hauteur, notée *h(F),* qui représente la plus grande valeur prise par sa fonction d'appartenance :

- $h(F) = sup_{x \in X} \mu_F(x)$, *F* est dit *normalisé* si sa hauteur vaut 1.

Sa $\alpha -coupe$ qui représente l'ensemble contenant les éléments ayant un degré d'appartenance supérieur ou égal à α :

- $\alpha -coupe(F) = \{ x \in X / \mu_F(x) \geq \alpha \}$

Sa cardinalité définie par : $card(F) = \sum_{x \in X} \mu_F(x)$

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

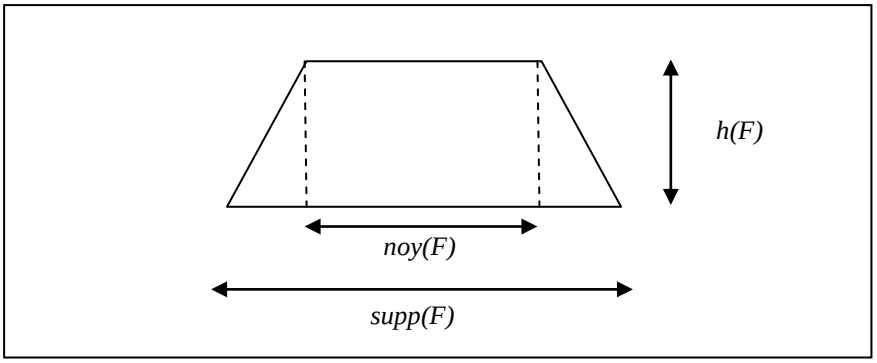


Figure 2.2 : Caractéristiques d'un ensemble flou [GAB 01]

2.3.3 Composition des ensembles flous

Les opérations de combinaison d'ensembles flous étendent les opérations ensemblistes usuelles. Soient *F* et *E* deux ensembles flous définis sur l'univers *X*, nous avons les définitions suivantes [KAU 77]:

- *Intersection :* $\forall x \in X, \mu_{F \cap E}(x) = \top(\mu_F(x), \mu_E(x))$
- *Union :* $\forall x \in X, \mu_{F \cup E}(x) = \perp(\mu_F(x), \mu_E(x))$
- *Différence :* $\forall x \in X, \mu_{F-E}(x) = \top(\mu_F(x), 1 - \mu_{\bar{E}}(x))$
- *Complémentation:* $\forall x \in X, \mu_F(x) = (1 - \mu_F(x))$

où $\top$ et $\perp$ désignent respectivement une *triangulaire norme* et une *triangulaire co-norme.*

2.3.4 Normes et Co-normes

Les opérateurs normes et co-normes triangulaires [BOS 04] généralisent respectivement les opérations d'intersection et d'union pour les ensembles flous, dont la contrepartie logique est la paire conjonction/disjonction (et/ou).

Une norme triangulaire (ou t-norme) $\top$ est une opération binaire sur l'intervalle [0,1]. Cette opération est associative, commutative, monotone et admet 1 comme élément neutre tel que $\top(x, 1) = x$.

De façon analogue, une co-norme triangulaire (ou t-co-norme) $\perp$ est une opération binaire sur l'intervalle [0,1]. Cette opération est associative, commutative, monotone et admet 0 comme élément neutre tel que $\perp (x, 0) = x$.

Tout couple (*norme / co-norme)* est lié par la relation suivante :

- $\top (x, y) = 1 - \perp (1 - x, 1 - y)$

La table 2.1 regroupe les principales normes et les co-normes correspondantes.

t-norme $\top$ (x, y)	**t-co-norme** $\perp$ (x, y)	**Nom**
min (x, y)	max (x, y)	Zadeh
x.y	x+y-xy	Probabiliste
max(x + y – 1, 0)	min(x + y, 1)	Lukasiewicz
		Hamacher

Table 2.1 : Principales normes et co-nromes triangulaires [BOS 04]

2.3.5 Inclusion floue

2.3.5.1 Inclusion booléenne

Une façon habituelle de définir l'inclusion de *A* dans *B (A et B deux ensembles classiques)* repose sur l'expression suivante :

- $A \subseteq B \Leftrightarrow (\forall x \in X, (x \in A) \Rightarrow (x \in B))$ *(2.1)*

Dans le cas de deux ensembles flous *E* et *F*, Zadeh [ZAD 65] exprime l'inclusion en comparant les fonctions d'appartenance de chaque ensemble :

- $E \subseteq F \Leftrightarrow (\forall x \in X, \ \mu_E(x) \leq \mu_F(x))$ *(2.2)*

L'inconvénient majeur de cette définition est qu'une réponse négative est obtenue dès que l'inclusion de l'ensemble *E* dans *F* (flous ou non) soit *presque vérifiée* ou *totalement insatisfaite* tel que c'est illustré dans l'exemple ci-après.

Exemple 2.2 :

Soient E_1, E_2 deux ensembles flous : E_1= {1/a , 1/b} et E_2 = {0.5/a , 0.7/b}
E_1 et E_2 ne sont pas inclus dans l'ensemble flou F= {0.4/a , 0.7/b}, au sens de l'inclusion de Zadeh. L'intuition incite à penser que c'est beaucoup plus net pour E_1 (qui en fait contient F) que pour E_2 (qui est *presque* inclus dans F). Pour distinguer entre ces deux situations, l'inclusion graduelle a été proposée.

2.3.5.2 Inclusion graduelle

L'inclusion graduelle étend, grâce à un degré, la notion d'inclusion d'un ensemble (flou ou non) dans un autre.
En partant de l'expression *(2.1)* : $A \subseteq B \Leftrightarrow (\forall x \in X, (x \in A) \Rightarrow (x \in B))$ qui permet d'obtenir pour des ensembles flous *E* et *F* :

- $deg(E \subseteq F) = min_{x \in X} \ \mu_E(x) \Rightarrow_f \mu_F(x)$ *(2.3)*

où $\Rightarrow_f$ est une implication floue. Plusieurs types d'implications peuvent être utilisés.

2.3.6 Implications floues

L'implication ordinaire $p \Rightarrow q$ *((non p) ou q)* rend vrai quand la proposition *q* est vraie ou quand la proposition *p* est fausse. Plusieurs approches d'extension de l'implication peuvent être envisagées lorsque les propositions *p* et *q* prennent, non plus une valeur booléenne, mais une valeur de l'intervalle unité. Toute implication floue est une fonction définie par :

$$\Rightarrow_f : [1, 0] \times [1,0] \longrightarrow [0, 1]$$
$$(p, q) \longmapsto p \Rightarrow_f q$$

et elle est d'autant plus vraie (resp. fausse) que son résultat est proche de 1 (resp. 0). Les implications floues peuvent être présentées selon plusieurs classifications [BOS 04].

2.3.6.1 S-implications

L'appellation *S-implication* vient du fait que ces implications font appel à une co-norme aussi dénommée *S-norme*. À partir de l'expression *((non p)* ***ou*** *q),* la classe des S implications (notées $\Rightarrow_{S\text{-}i}$) est définie par :

- $p \Rightarrow_{S\text{-}i} q = \perp (1 - p, q)$,

où la disjonction est généralisée par une co-norme $(\perp)$. Il existe une infinité de S-implications et les trois les plus communes sont données dans la Table 2.1

Nom	**Définition**	**Co-nome associée**
Kleene Dienes	$p \Rightarrow_{K\text{-}D} q = max (1- p , q)$	$\perp (x , y) = max (x, y)$
Reichenbach	$p \Rightarrow_{Rb} q = 1- p + p * q$	$\perp (x , y) = x+y - xy$
Lukasiewicz	$p \Rightarrow_{Lu} q = min (1- p+q , 1)$	$\perp (x , y) = min (x+ y, 1)$

Table 2.1 : Les trois principales S-implication

2.3.6.2 R-implications

La seconde classe d'implications floues regroupe les R-implications, ainsi dénommées parce qu'elles utilisent le principe de *résiduation*. Dans le cas classique : $(p \textbf{ and } (p \Rightarrow q)) \Leftrightarrow (p \text{ and } q)$, et donc la valeur de vérité de la proposition $(p \textbf{ and } (p \Rightarrow q))$ est inférieure ou égale à celle de la proposition q , i.e., $(p \textbf{ and } (p \Rightarrow q)) \leq q$.

Dans le cas flou, cette inégalité sert de point de départ pour chercher l'élément maximal la vérifiant, soit :

- $p \Rightarrow_{R\text{-}i} q = sup \{ u, \mid \top (p, u) \leq q \}$,

où $\top$ est une t-norme. Il y a une infinité de R-implications dont la forme générale est :

- $p \Rightarrow_{R\text{-}i} q = \begin{cases} 1 & \text{si } p \leq q \\ f(p, q) & \text{sinon} \end{cases}$

les plus courantes sont données dans la table 2.2.

Nom	**Définition**	**Norme associée**
Gödel	$p \Rightarrow_{Gö} q = \begin{cases} 1 & \text{si } p \leq q \\ q & \text{sinon} \end{cases}$	$\top(x, y) = min (x, y)$

Goguen	$p \Rightarrow_{Gg} q = \begin{cases} 1 & \text{si } p \leq q \\ q/p & \text{sinon} \end{cases}$	$\mathsf{T}(x, y) = x.y$
Lukasiewicz	$p \Rightarrow_{Lu} q = \begin{cases} 1 & \text{si } p \leq q \\ 1-p+q & \text{sinon} \end{cases}$ $= \min(1-p+q, 1)$	$\mathsf{T}(x, y) = \max(x + y - 1, 0)$

Table 2.2 : Les trois principales R-implications

Exemple 2.2 :

Soient E, F deux ensembles flous.

E = {0.1/a1, 0.9/a2, 1/a3, 0.7/a4} et F = {1/a1, 0.6/a2, 0.8/a3, 0.7/a4}.

L'inclusion booléenne de E dans F n'est pas satisfaite. Cependant, l'inclusion graduelle nous permet de calculer les degrés d'inclusion de E dans F en appliquant la formule *(2.3) :*

- *min (1, 0.6, 0.8, 1) = 0.6,* avec l'implication de Gödel,
- *min (1, 2/3, 0.8, 1) = 2/3,* avec celle de Goguen,
- *min (1, 0.7, 0.8, 1) = 0.7,* avec celle de Lukasiewicz.

Les R-implications sont plus appropriées pour la modélisation d'une inclusion puisque le résultat est maximal (i.e. égal à 1) quand la conclusion est supérieure (ou égale) à l'antécédent. Ceci est conforme avec le principe décrit dans la formule *(2.2).*

2.4 Apport du flou dans les bases de données

Les systèmes de base de données classiques permettent de traiter des requêtes de type booléen. Cependant, les utilisateurs expriment leurs besoins avec des expressions vagues et imprécises pour lesquels la modélisation[1] avec une requête booléenne ne retourne pas souvent leurs préférences. Pour étendre la possibilité de ces systèmes à prendre en compte les préférences dans les requêtes, la théorie des ensembles flous semble être une solution bien adaptée.

[1] Modélisation : au sens traduction en un langage de requêtes.

Formellement, la théorie des ensembles flous offre un moyen pour exprimer la préférence par rapport à un élément *x* grâce à la fonction d'appartenance, plus $\mu_F(x)$ est proche de 1, plus l'élément *x* est préféré et plus $\mu_F(x)$ est proche de 0, moins *x* est préféré [BOS 04]

Sur ce est développé la thématique de l'interrogation flexible, qui fera l'objet du troisième chapitre de notre travail.

Chapitre 3

Requêtes Flexibles

3.1 Introduction

Dans le présent chapitre, nous nous intéressons aux requêtes flexibles. Différentes approches existent permettant d'exprimer les requêtes flexibles, toutefois la théorie des sous – ensemble flous semble être le moyen le plus expressif et le plus adapté [BOS 04].

Nous présenterons donc les concepts de base de cette approche, l'algèbre relationnelle étendue aux relations floues et un aperçu du langage SQLf. L'aspect évaluation de ces requêtes est aussi abordé dans ce chapitre.

3.2 Objectifs et motivations

Une requête flexible (i.e une requête floue) est une requête traduisant une notion de préférence. Les conditions d'une telle requête ne sont plus exprimées par des prédicats booléens, mais remplacés par des prédicats flous. De ce fait, le problème n'est plus de décider si un élément satisfait ou non une condition, mais dans quelle mesure il l'a satisfait. Ce qui induit un ordre sur les réponses [YES 11].

Les requêtes flexibles étendent alors les requêtes booléennes en introduisant les préférences dans les critères de recherche. La requête « *trouver les appartements **pas trop chers** et situés **pas trop loin** du centre ville* » est un exemple de requête flexible dans laquelle on ne peut pas déterminer à partir de quel prix un appartement devient trop élevé, mais il s'agit plutôt d'effectuer une discrimination entre les prix qui sont parfaitement acceptables pour l'utilisateur, trop élevé, qui demeurent plus au moins acceptables (spécialement si l'appartement est proche du centre ville). Une façon bien adaptée pour modéliser les préférences (sur le prix et le lieu) exprimées dans cet exemple est l'utilisation de prédicats flous dont la satisfaction est une affaire de degrés et non en « tout ou rien». Le résultat obtenu n'est plus un ensemble « plat » d'éléments, mais un ensemble ordonné où chaque élément est affecté d'un degré de satisfaction par rapport à la requête considérée [YES 11].

Le but des requêtes flexibles est de fournir à l'utilisateur des réponses plus qualitatives par rapport à celles fournies par les SGDB classiques. Les réponses retournées sont des réponses approchées (approximatives) et triées en fonction de leur degré de

satisfaction alors qu'une requête booléenne aurait pu retournée une réponse vide [YES 11]. Ainsi, les requêtes flexibles peuvent s'avérer très pertinentes dans certains domaines d'application tels que les systèmes documentaires, les pages jaunes, les petites annonces, la recherches d'information sur le web, l'interrogation de bases de données multimédias, etc.

3.3 Concepts de base

Pour exprimer une requête floue, deux concepts y interviennent : *les prédicats flous* et les relations floues [BOS 04].

3.3.1 Prédicats flous

Un prédicat flou est modélisé par à un ensemble flou. Il exprime dans quelle mesure un argument donné satisfait le prédicat. Par exemple, si *" x"* est une variable exprimant l'âge et si *"jeune"* est un ensemble flou défini sur l'univers des âges , alors *" x est jeune "* est un prédicat qui retourne le degré d'appartenance de *" x "* dans l'ensemble flou *" jeune"* .
Les prédicats flous peuvent être de différents types :

- Prédicats simples comportant des adjectifs comme *"jeune", "grand", "important".*
- Prédicats modulés par des modificateurs linguistiques comme *"très", "trop", "relativement".*
- Prédicats quantifiés avec des quantificateurs flous comme *"la plupart", "environ la moitié"* et *"une douzaine".*
- Prédicats composés (conjonction et disjonction des prédicats atomiques ou simples).

3.3.2 Relation floue

Une relation floue (ou graduelle) est le résultat de l'application de prédicats flous à une relation booléenne. Formellement, une relation floue *r* de schéma *R* est un sous-ensemble flou du produit cartésien d'un ensemble de domaines *D1, D2, ...Dn* . Chaque n-uplet *t* de la relation est affecté d'un poids $\mu r(t)$ ($\mu r(t) \in [0,1]$) indiquant la mesure dans laquelle le n-uplet *t* appartient à la relation *r*. Seuls les n-uplets *t* tels que $\mu r(t) > 0$ sont effectivement stockée dans la relation. Notons aussi qu'une relation ordinaire n'est qu'un cas particulier tel que pour tout *t,* $\mu r(t) = 1$.

Exemple 3.1 : [YES 11]

Soit la relation usuelle *employé (num, nom, salaire, âge, ville-res)* d'extension donnée ci-dessous:

Num	Nom	salaire	Age	ville-res
17	Rami	35000	42	Alger
76	Maria	30000	27	Tipaza
26	Ryad	25000	37	Alger
12	Myriam	25000	39	Alger

A partir de la relation initiale *employé,* la relation *employé -bp* des employés bien payés est donnée ci-dessous :

Num	nom	salaire	Age	ville-res	degré μ
17	Rami	35000	42	Alger	0,6
76	Maria	30000	27	Tipaza	0,4
26	Ryad	25000	37	Alger	0,2
12	Myriam	25000	39	Alger	0,2

Qui peut être construite à l'aide du prédicat "*bien-payé* " défini par la fonction donnée en Figure 3.1 :

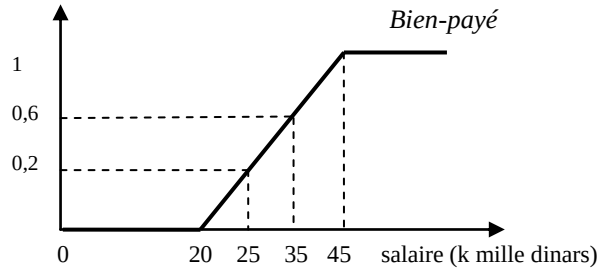


Figure 3.1 : La fonction d'appartenance d'un prédicat graduel

3. 4 Extension de l'algèbre relationnelle

Les opérations usuelles de l'algèbre relationnelles peuvent être étendues aux relations floues, d'une part en les considérants comme des ensembles flous (pour ce qui est des

opérations ensemblistes), d'autre part en introduisant des prédicats graduels au lieu des seuls prédicats booléens [BOS 08]. Soient r et s deux relations floues définies sur le même ensemble d'attributs X.

Les opérations ensemblistes sont définies comme suit :

- *Union* : $\mu_{union}(r, s)(t) = \perp(\mu_r(t), \mu_s(t))$
- *Intersection* : $\mu_{inter}(r, s)(t) = \top(\mu_r(t), \mu_s(t))$
- *Différence* : $\mu_{differ}(r, s)(t) = \top(\mu_r(t), 1 - \mu_s(t))$, *avec* $r - s = r \cap \bar{s}$

Le produit cartésien de deux relations floues r et s définies respectivement sur l'ensemble de domaines X *et* Y est donné par *:*

- $\mu_{prod}(r, s)(tu) = \top(\mu_r(t), \mu_s(u))$

Les opérations relationnelles sont définies comme suit :

- *Selection* : $\mu_{select}(r, cond)(t) = \top(\mu_r(t), \mu_{cond}(t))$, *où cond : prédicat graduel*

- *Projection* : $\mu_{project}(r, Y)(u) = max_v\ \mu_r(uv)$,

  où Y est un sous –ensemble de X avec X ensemble d'attribut de r et u est l'une de ses valeurs, tandis que v prend ses valeurs dans X-Y.

- *Jointure* : $\mu_{join}(r, s, A, B, \theta)(tu) = \top(\mu_r(t), \mu_s(u), \mu_\theta(t.A, u.B))$,

  où A (resp. B) est un sous-ensemble de X (resp. Y) avec X ensemble d'attribut de r (resp. s), A et B sont compatibles, θ est un comparateur binaire (éventuellement graduel), et t. A (resp. u. B) désigne la valeur du composant A (resp. B) du n-uplet t (resp. u).

- *Division* : $\forall x \in supp(project(r, X))$,

  $$\mu_{div}(r, s, A, B)(x) = \min_{a \in supp(s)} (\mu_s(a) \Rightarrow_f \mu_r(a, x))$$

  L'extension de la division classique aux relations floues s'appuie sur une adaptation de l'expression usuelle de la division (voir chapitre 1) :

1. l'implication usuelle est remplacée par une implication dite floue notée $\Rightarrow_f$, et le quantificateur universel est interprété par le *minimum* .
2. la division est restreinte aux seules valeurs de X apparaissant dans la relation dividende, i.e., aux valeurs x appartenant au support de la projection de la relation

floue *r* sur *X*, noté *supp (project (r, X))* où le *support* est la relation usuelle définie par: $supp(r) = \{t \mid \mu_r(t) > 0\}$.

où $\top$ et $\perp$ désignent respectivement une *norme triangulaire* (généralisant la conjonction) et une *co-norme triangulaire* (généralisant la disjonction) (voir chapitre 2).

Exemple 3.2 : [YES 11]

Soit la base de données comportant les relations : *emp (nom, salaire, âge, dep)* et *dept (nd, budget)* décrivant les employés (et notamment le département où ils travaillent) et les départements d'une compagnie. La requête recherchant les couples nom d'employé et budget de département tels que le département a un budget *moyen* et l'employé est *jeune*, ayant un salaire *très élevé* et travaillant dans ce département, peut s'exprimer de la façon suivante :

*project (join (select (dept, budget = "**moyen**")),*
*(select (emp, âge = "**jeune**" and salaire = "**très élevé**"),*
{dep}, {nd}, =),
{nom , budget}).

nom	Salaire	âge	dep
Rami	12500 (0,1)	38 (1)	23
Maria	14500 (0,9)	42 (0,4)	23
Ryad	13000 (0,7)	37 (1)	23
Myriam	12500 (0,3)	39 (0,8)	17

Extension de la relation *(emp)*

nd	budget
17	200 (1)
23	150 (0,6)

Extension de la relation *(dept)*

où les degrés associés à l'attribut salaire (respectivement âge, budget) expriment *dans quelle mesure* le salaire (respectivement l'âge, le budget) est *très élevé* (respectivement *jeune*, *moyen*), on obtient le résultat :

{0.3/< Myriam, 200>, 0.6/<Ryad, 150>, 0.4/<Maria, 150>, 0.1/<Rami, 150>}.

Ceci est l'exemple typique d'une requête floue décrivant une préférence sur le budget, l'âge et le salaire. Le résultat obtenu est un ensemble de tuples pondérés et ordonnés.

3.5. Aperçu sur le langage SQLf

En parallèle à l'algèbre relationnelle étendue, une extension du langage SQL, appelée SQLf, a été proposée dans [BOS 95]. Le langage SQLf permet d'interroger que des bases de données usuelles contenant des relations strictes, les relations floues sont uniquement le résultat de requêtes floues.

Dans ce qui suit, un petit aperçu de ce langage est fourni :

3.5.1 Bloc de base

La structure en trois clauses : "***select***", "***from***" et "***where***" du bloc de base SQL est conservée dans SQLf :

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.

Les paramètres "*n*" et "*t*" du bloc "*select*" limitent le nombre des réponses en utilisant un critère quantitatif (les "*n*" meilleures réponses) ou un critère qualitatif (les données qui satisfont la requête suivant un seuil supérieur à "*t*"). Le mot-clé "*distinct*" permet, comme en SQL classique, d'éliminer les doublons du résultat. Avec ce mot-clé, c'est le tuple de plus fort degré d'appartenance qui est retourné (il est le plus représentatif de la requête) [BOS 04].

Exemple 3.3 :
Soient les relations *emp* de schéma *emp (emp# , e-nom, salaire, emploi , âge, dep-travail)* et *dept* de schéma *dept (dep#, d-nom, budget).*

emp#	e-nom	salaire	emploi	âge	dep-travail
76	Maria	30000	Ingénieur	40 (0.25)	5
26	Ryad	20000	Secrétaire	24 (1)	3
12	Myriam	25000	Technicien	39 (0.3)	3
55	Liliane	25000	Comptable	30 (0.75)	1

Extension de la relation *(emp)*

dep#	d-nom	budget (k dinars)
1	Réseaux	50000 (1)
3	Composants	15000 (0.65)
5	Bureautique	4000 (0.1)

Extension de la relation *(dept)*

La requête cherchant à trouver « les noms des employés satisfaisant au-delà du degré 0.6 la condition : être jeune et travailler dans un département à budget élevé » est exprimée comme suit :

select distinct 0.6 e-nom
from *emp, dept*
where dep-travail = dep# and âge = "*jeune*" and budget = "*élevé*"

Supposons la fonction d'appartenance du prédicat flou *" jeune"* (voir la Figure 3.2) :

$$\mu_{jeûne} = \begin{cases} 1 & \text{si âge} \leq 25, \\ 0 & \text{si âge} \geq 45, \\ \text{décroit linéairement} & \text{si } 25 < \text{âge} < 45 \end{cases}$$

et la fonction d'appartenance du prédicat flou *"élevé"* (voir la Figure 1.3) :

$$\mu_{élevé} = \begin{cases} 0 & \text{si budget} \leq 2000, \\ 1 & \text{si budget} \geq 22000, \\ \text{croit linéairement} & \text{si } 2000 < \text{budget} < 22000 \end{cases}$$

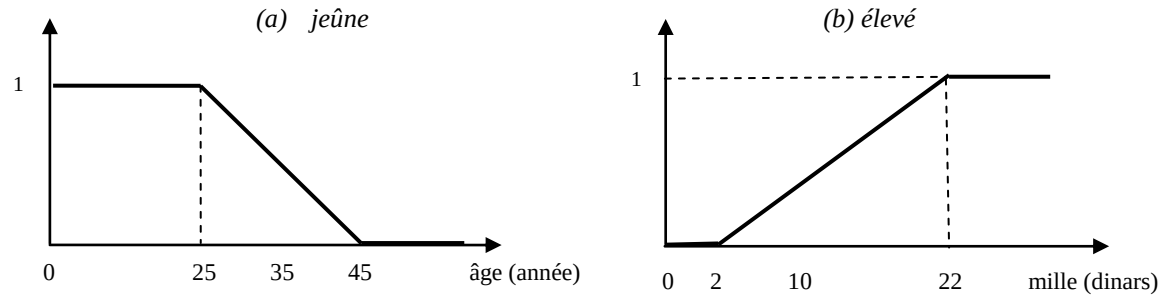


Figure 3.2 : Les fonctions caractéristiques *: (a) « jeûne»* et *(b) « élevé »*

Ainsi, la relation résultante est :{ 0.65/ <Ryad >, 0.75 /<Liliane >} puisque seuls ces deux employés se voient associer un degré supérieur au seuil exigé 0.6.

3.5.2 Sous-requêtes

Outre les prédicats simples présentés dans la section précédente, SQLf permet l'utilisation de prédicats construits par imbrication d'un autre bloc de base appelé sous-requête. Les opérateurs utilisés pour exprimer une sous –requête sont principalement : l'appartenance "*in*", la non-vacuité " *exists* " , et la quantification universelle et existentielle avec un opérateur de comparaison " θ *any* etθ *all* " . Ainsi, pour "*A in (select B from …)*", l'opérateur "*in*" retourne le degré d'appartenance de la valeur de *A* du n-uplet courant à l'ensemble de valeurs retournées par la sous –requête *(select B from …)*. [YES 11]

Exemple 3.4

La requête de l'exemple précédent peut être exprimée en utilisant une sous-requête comme suit :

select 0.6 e-nom
from *emp*
where âge = *"jeune"* and dep-travail ***in*** *(* ***select*** dep#
from *dept*
where budget = *"élévé").*

Avec l'extension *dept* donnée précédemment, la sous-requête retourne l'ensemble flou {1/<1>, 0.65/<3>, 0.1/<5>}, et le résultat final obtenu est identique au résultat de l'exemple 3.3 : 0.65 = min (1, 0.65) pour Ryad et 0.7 = min (0.75, 1) pour Liliane.

3.6 Evaluation des requêtes flexibles

La question d'évaluation des requêtes est un point important qu'on ne peut négliger lorsqu'on traite de nouvelles approches d'interrogation de base de données. Il est certes intéressant de proposer l'intégration des préférences dans les requêtes, mais sans que cela induise une perte en terme de performance. Les études d'évaluation alors s'intéressent à quantifier le coût additionnel dû à la prise en compte de la gradualité, autrement dit mesurer le surcout du caractère flou introduit dans les requêtes.

Il existe deux méthodes d'évaluation des requêtes flexibles :

1. La méthode par dérivation

Elle repose sur la traduction d'une requête floue en une requête booléenne SQL. Dans ce cas, le calcul des degrés de satisfaction liés aux relations floues manipulées sont effectués aux moyens de fonctions externes inclues dans la requête obtenue. Ce principe est appelé « principe de dérivation ».

2. La méthode algorithmique

Au lieu d'utiliser le langage SQL pour évaluer la requête, on peut faire appel à la méthode algorithmique dans laquelle le langage SQL est utilisé uniquement pour accéder aux données. Celle-ci repose donc sur la compilation de la requête floue en un algorithme d'évaluation codé dans un langage procédural. Les degrés de satisfaction, dans ce cas, sont calculés directement par le programme.

Les résultats des expérimentations menés sur l'étude des performances [BOS 04], en comparant l'évaluation du coût d'une requête floue et le coût d'une requête classique, obtenus sur des requêtes de type sélection, projection, jointure et division ont montré que le surcoût lié à la gradualité reste raisonnable (en général bien inferieur à 50%).

3.7 Conclusion

Nous avons décrit dans ce chapitre le principe de l'interrogation flexible à base de la théorie des ensembles flous, notamment par la présentation des langages de manipulation des requêtes étendues aux relations floues : l'algèbre relationnelle étendue et le langage SQLf.
Nous avons essayé de montré l'avantage d'offrir aux utilisateurs la possibilité d'exprimer leurs préférences par des termes vagues du langage naturels.

Compte tenue de ces études théoriques, nous avons voulu montrer dans notre travail la faisabilité d'appliquer l'approche d'interrogation flexible sur un cas d'étude réel. Ce qui a fait l'objet de la deuxième partie de notre travail.

Partie II

Un Système d'Interrogation Flexible de Réservation d'Hôtels en Ligne

Chapitre 4

Analyse et Conception

4.1 **Introduction**

Pour montrer un cas réel ou l'interrogation flexible s'avère très indispensable et pertinente, nous avons opté pour un système d'interrogation de base de données en ligne. Les utilisateurs n'ont pas une connaissance sur le contenu ou la structure interne de la base alors ces derniers expriment leurs besoins avec des termes vagues du langage naturels. Les requêtes classiques booléennes supportées par les SGBD actuels ne répondent pas à des requêtes naturelles. Elles doivent être converties. Nous avons déjà montré l'handicape de ces dernières lors de leurs conversion en une requête booléenne.

A cet effet, nous avons voulu apporté une contribution expérimentale à ce domaine très vaste de l'interrogation flexible en montrant qu'une requête flexible répond au mieux aux besoins des utilisateurs. Nous devons souligner que les études restent théoriques dans ce domaine. Très peu, même très rare d'expérimentations ont été appliquées.

4.2 Choix du domaine d'application

Notre choix s'est porté sur la réservation d'hôtels en ligne au niveau de la wilaya de Tizi-Ouzou. Nous partons de l'idée de satisfaire la recherche d'un utilisateur désirant trouver un hôtel selon les préférences qu'il exprime. Il s'agit alors de concevoir une solution web dynamique. Pour ce faire nous allons suivre la démarche suivante.

4.3 Description globale de la démarche à suivre

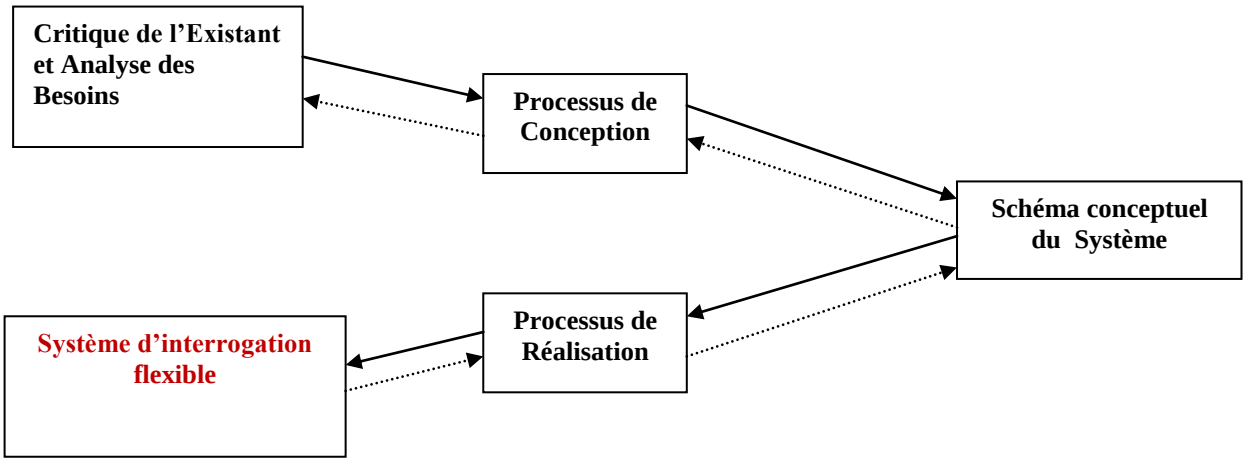


Figure 4.1 : Démarche globale

Afin de concevoir notre système, notre choix pour UML est argumenté du fait qu'il couvre toutes les phases de développement, de l'analyse de besoins à la réalisation.

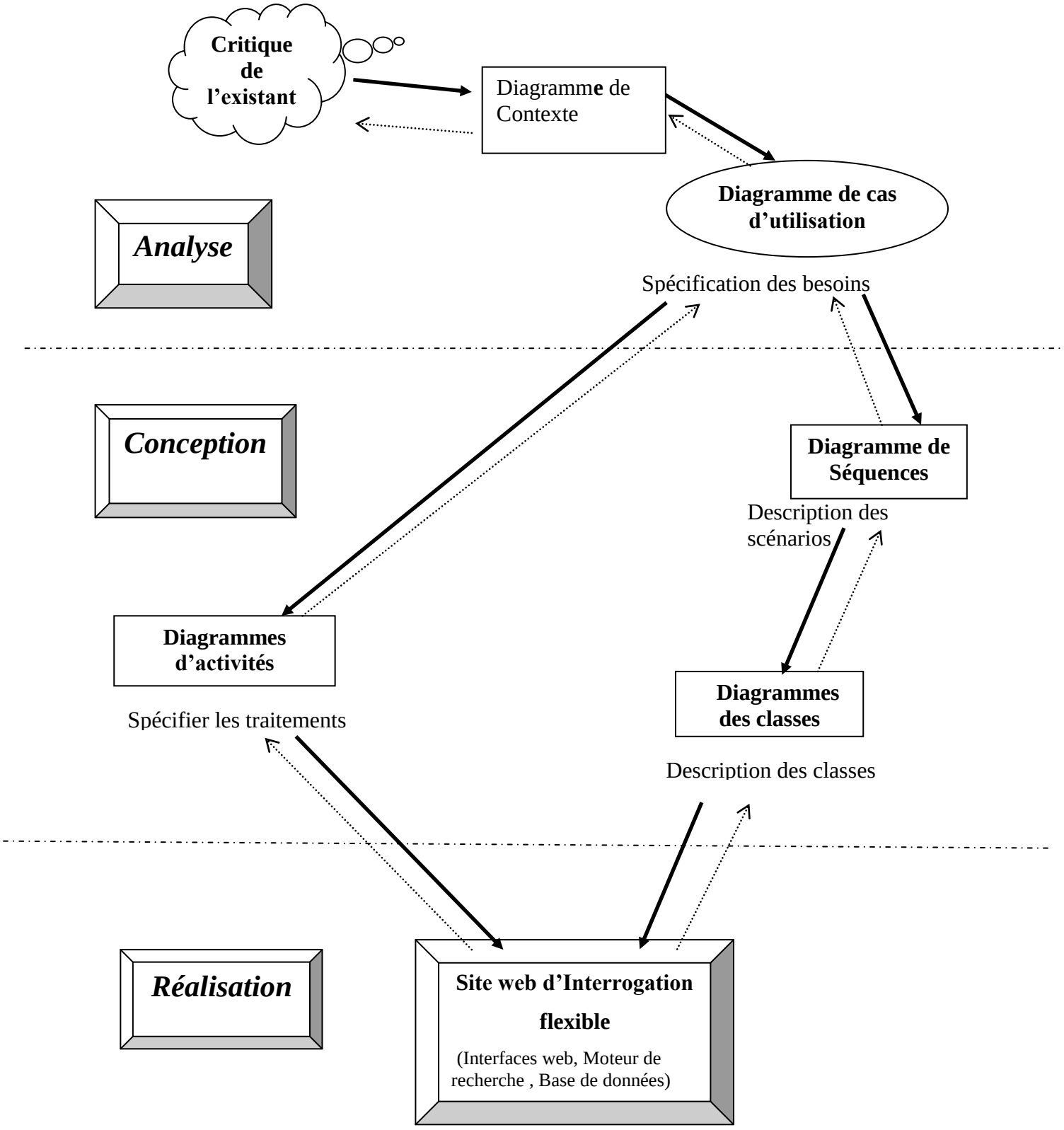


Figure 4.2 : Démarche globale de conception / Réalisation suivant les diagrammes UML

4.4 Description des phases

4.4.1 Phase d'analyse

- ***Critique de l'existant* :** consiste à consulter les applications disponibles sur le marché concernant la réservation d'hôtels, les moteurs de recherche basée sur l'interrogation booléenne et de les critiquer en faisant ressortir leurs manques.

- ***Analyse des besoins* :** consiste à étudier l'existant en recensant toutes les informations nécessaires sur le champ d'étude, notamment en termes d'expression des besoins des utilisateurs.

- ***Diagramme de contexte :*** il sert à délimiter le domaine d'étude en précisant les acteurs, les systèmes externes et leurs interactions avec notre système.

- ***Diagrammes de cas d'utilisation* :** les cas d'utilisation permettront de décrire les fonctionnalités de notre système et leurs interactions avec les utilisateurs. On peut décrire les interfaces à partir de ce niveau.

4.4.2 Phase de conception

- ***Diagrammes de séquences* :** ces schémas permettront de détailler tous les scénarios possibles d'un cas d'utilisation en précisant les objets de notre système et les acteurs.
- ***Diagrammes de classes :*** permettra d'établir un schéma conceptuel de la partie statique de notre système en décrivant les classes ainsi que les classes associatives.
- ***Diagrammes d'activité* :** permettront de modéliser les aspects dynamiques du notre système en donnant une représentation des différents processus faisant intervenir les acteurs concernés et manipulant les ressources nécessaires.

4.4.3 Phase de réalisation

A partir du diagramme de classe que l'on pourra dériver le modèle relationnel de la base de données à implémenter. Les diagrammes d'activité permettront de mettre en œuvre les interfaces logicielles nécessaires à l'exploitation de la base de données ainsi que le fonctionnement de notre moteur de recherche.

4.5. Phases d'Analyse

4.5.1 Etude de l'existant

Après une sortie sur le terrain au niveau de notre ville, nous avons recensé six hôtels : Amraoua, Ittourar, Les 3 Roses (ex :Concorde), le Palais Royale, Baloua et l'hôtel Lalla-Khedidja. Parmi ces hôtels, le palais Royale et l'hôtel Ittourar qui dispose d'un site web (www.hotel-ittourar.com) et offre le service de réservation en ligne. Dans ce dernier site, un client désirant effectuer une réservation doit sélectionner la date début et la date départ ainsi que le nombre de places puis communiquera ses coordonnées dans un formulaire. Après validation, il y a un message qui s'affiche et qui l'informe que ses données ont été bien enregistrées et qu'il doit patienter jusqu'à ce que l'agent de l'hôtel le contacte pour valider sa réservation.

Quant aux autres hôtels, on trouve seulement quelques pages ou l'on renseigne quelques informations comme par exemple : le nombre d'étoile, l'adresse, le numéro de téléphone. Celui qui désire effectuer une réservation, il n'a qu'à appeler et souvent ces numéros de lignes ne sont pas en marche puisque le plus part des ces informations ne sont pas mises à jours.

D'après notre petite enquête au niveau de notre ville, on a déduit les manques suivants :

- Un client potentiel ne peut pas effectuer une comparaison entre les hôtels,
- Les coordonnées de chaque hôtels sont pas mises à jours,
- Dans le cas d' Ittourrar, la procédure de réservation en ligne qu'il offre est un peu mal conçue . Un client ne reçoit aucune confirmation quant à sa réservation lancée, et il ne sait pas combien de temps pourrait –il attendre pour recevoir l'appel de l'agent d'hôtel lui validant sa réservation. Donc au final, cette procédure ne sert à rien. Il vaut mieux se déplacer ou appeler par téléphone pour faire sa réservation,
- Un client potentiel ne peut pas vérifier la disponibilité des chambres en ligne,
- Si l'agent d'hôtel effectue une réservation via le site ou par téléphone, et que ce dernier ne se présente pas, l'hôtel est perdant. La chambre a été réservée , mais n'a pas été réglée.

Sur le niveau national et International :

Sur le niveau national, il existe un annuaire qui recense les hôtels en mentionnant les coordonnées de chaque hôtel (Nombre d'étoiles, Adresse, email, Tél..).

Sur le niveau international, par exemple le site www.hotels.com. Dans ce site, un client désirant effectuer une réservation sélectionne un pays, une ville, date début, date départ, et le nombre de places. Le moteur de recherche retourne les hôtels selon l'ordre croissant des prix en affichant aussi le nombre de places libres. Le manque de ce site est qu'il limite les critères de sélections, et que le client par exemple ne peut pas effectuer une recherche selon l'approximation de ces hôtels envers lui ou bien envers des places connues comme de l'aéroport.

Quand on sélectionne notre pays l'Algérie, aucun résultat n'est retourné. On peut déduire que la culture de réservation d'hôtel en ligne n'est pas encore instaurée chez nous.

Sur le fameux moteur de recherche Google (www.Google.com) , si on émet la requête suivante : « *je cherche l'hôtel le plus proche de la maison de la culture et le moins cher dans la ville de Tizi-Ouzou* ». Google retourne un bon nombre de résultats qui ne sont pas pertinents. Il retourne des sites qui parlent sur les hôtels, d'autre sites qui parlent sur la ville de Tizi-Ouzou ou bien la maison de la culture. Google n'est pas adapté à ce genre de requête puisqu'il travaille dans le contexte général, il a été conçu pour effectuer des recherches dans des sites internet ou bien des fichiers et rarement dans des bases de données et il n'utilise pas la géo-localisation dans ses recherches, donc cela est une faille pour ce système. Si on exploite cette faille, on pourrait arriver développer une application qui aura un grand essor dans les années à venir.

4.5.2 Analyse des besoins

- Chaque hôtel est caractérisé par les informations suivantes : nom hôtel, adresse, nombre d'étoiles, possède ou pas un restaurant, coordonnées géographique (latitude et longitude dans Google Earth), et on suppose qu'il possède un compte bancaire (dans les systèmes bancaires internationaux comme MasterCard ou Visa).

- Chaque hôtel est composé d'un bon nombre de chambres, et chaque chambre est caractérisé par son numéro, son prix, son type, ainsi que son état (combien de fois elle est occupée dans le temps à partir du moment présent).

- Chaque type décrit une ou plusieurs chambres dans ces hôtels, et chaque type est caractérisé par : le numéro de type, nombre de places, nombre de lits, possède ou pas une salle de bains ou bien une TV.

- Chaque hôtel est géré par un agent ou plusieurs agents d'hôtels, et ils sont caractérisés par : nom, prénom, adresse, e-mail, date naissance, lieu de naissance, numéro de téléphone et un mot de passe pour accéder à leurs espaces. L'agent d'hôtel a comme tâches principales :
  - Gérer les actions (supprimer une action, convertir une réservation vers une location,)

    Remarque : une action, soit c'est une réservation, soit une location.
  - Gérer les chambres (ajouter, supprimer ou modifier une chambre, louer une chambre),
  - Consulter l'historique des actions.

- Un visiteur de notre site peut effectuer les tâches suivantes :
  - Effectuer une recherche d'hôtels selon ses besoins ou ses critères en tapant une requête en langage naturel

    Exemples : *l'hôtel le plus proche et le moins chers* ; *l'hôtel le plus proche de la nouvelle gare routière possédant un restaurant, hôtel le plus loin du stade possédant une salle de bains'*.
  - Ajouter un hôtel
  - S'ajouter comme étant un agent d'hôtel.

- Un client (un visiteur ayant l'intention de réserver ou bien de louer) peut effectuer les tâches suivantes :
  - Effectuer une action (réservation ou location) . Avant celà, notre système doit vérifier les choses suivantes :

    - La disponibilité des chambres libres
    - Si les données saisies sont correctes (contrôles)
    - Si le client a assez de crédit

Remarques :

Gestion des règlements des actions et remboursements :

- Si l'action demandée est une réservation, le client doit payer 50% du prix de la chambre. S'il s'agit d'une location, il doit payer 100% , et dans les deux cas, le client reçoit un message de confirmation contenant un code pour l'action effectuée.
- Pour ouvrir droit à un remboursement, le client ne doit pas dépasser le délai qui est fixé à 24 heures avant la date d'arrivé à l'hôtel, sinon aucun remboursement ne lui sera accordé.
- Le client peut compléter sa réservation (payer les 50% restants du prix en espèce) au niveau de l'hôtel avant 14 :00 de la date prévue d'arrivée, sinon sa réservation sera annulée systématiquement.
- Le DBA (Data Base Administrator ou bien le propriétaire du site) gagne 5% pour chaque action effectuée par les clients.

4.5.3 Diagramme de contexte

D'après l'analyse des besoins, on déduit que notre système a quatre acteurs et trois systèmes externes.

Les acteurs :

- Un visiteur : toute personne s'intéressant à notre site, par exemple pour effectuer des recherches.

- Un client : c'est un visiteur ayant l'intention de réserver ou bien louer.

- Un agent d'hôtel : un utilisateur qui gère son hôtel

- **L'Administrateur :** le propriétaire de l'application (DBA)

Les systèmes externes :

- Les systèmes de paiement en ligne: comme MasterCard ou Visa

- La Géo-localisation : comme GPS pour localiser le client, les hôtels et les places spéciales pour calculer les distances entre eux.

-Le système de messagerie : comme Gmail, Yahoo, Hotmail pour que les clients puissent recevoir leurs mots de passe au cas d'oublis, recevoir les codes de réservation ou de location (une fois ces dernières son confirmées par notre système).

Le schéma suivant montre le diagramme de contexte :

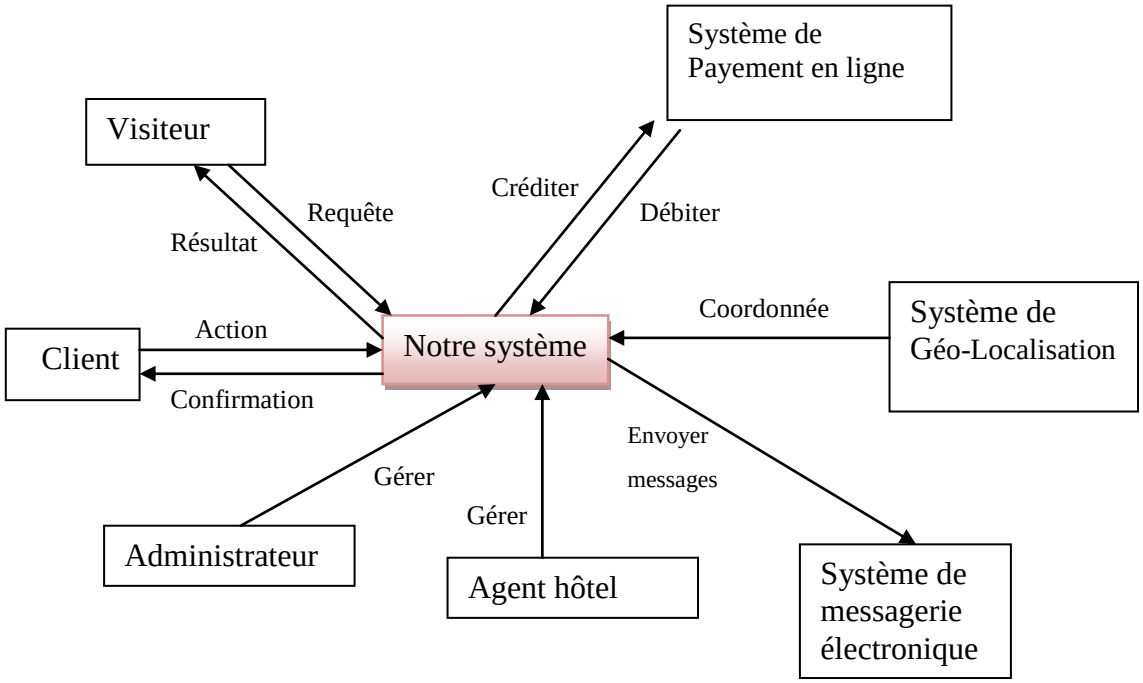


Figure 4.3 : Diagramme de contexte

Acteurs	**Description des besoins fonctionnels**
Visiteur	– Effectuer une recherche selon ses préférences
Agent d'hôtel	– S'authentifier – Vérifier les informations des clients – Ajouter une extension à l'hôtel – Faire une préinscription pour l'hôtel – Effectuer une réservation et/ ou une location – Annuler une réservation et /ou une location
Client	– Effectuer une réservation – Louer une chambre – Annuler une réservation – Convertir une réservation à une location – Annuler une location

Administrateur (DBA)	– Confirmer l'inscription d'un hôtel – Modifier les données d'un hôtel (attributs)

4.5.3 Les cas d'utilisations

Ce tableau montre la description préliminaire des cas d'utilisations :

Cas d'utilisation	**Acteur**
Ajouter une extension	Agent hôtel
Faire une préinscription	Agent hôtel/client
Effectuer une réservation	Agent hôtel /client
Effectuer une location	Agent hôtel/client
Confirmer l'inscription	Administrateur
Modifier un hôtel	Administrateur
Effectuer une réservation	Client
Effectuer une location	Client

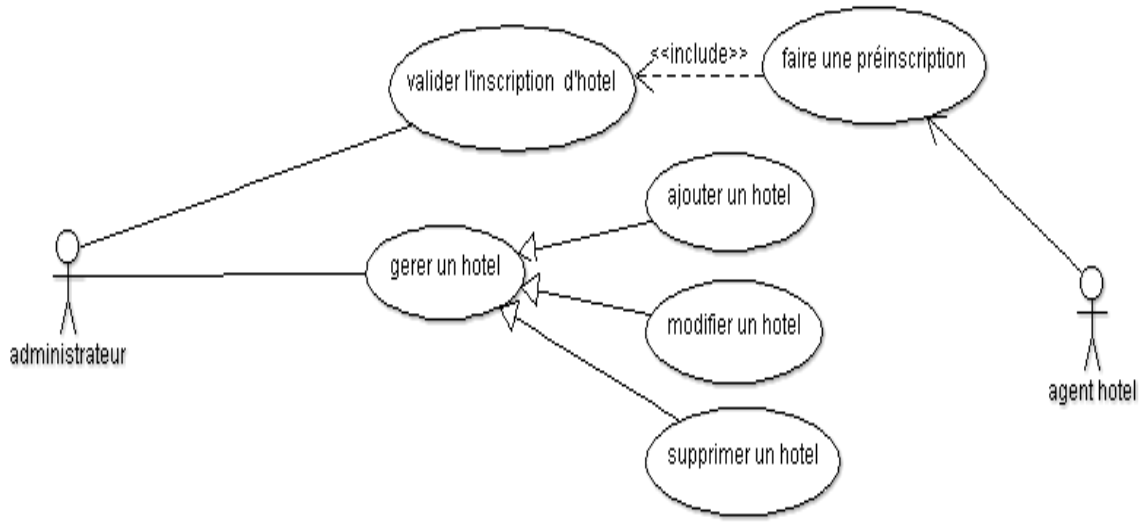


Figure 4.4 : Diagramme de cas d'utilisation pour « administrateur »

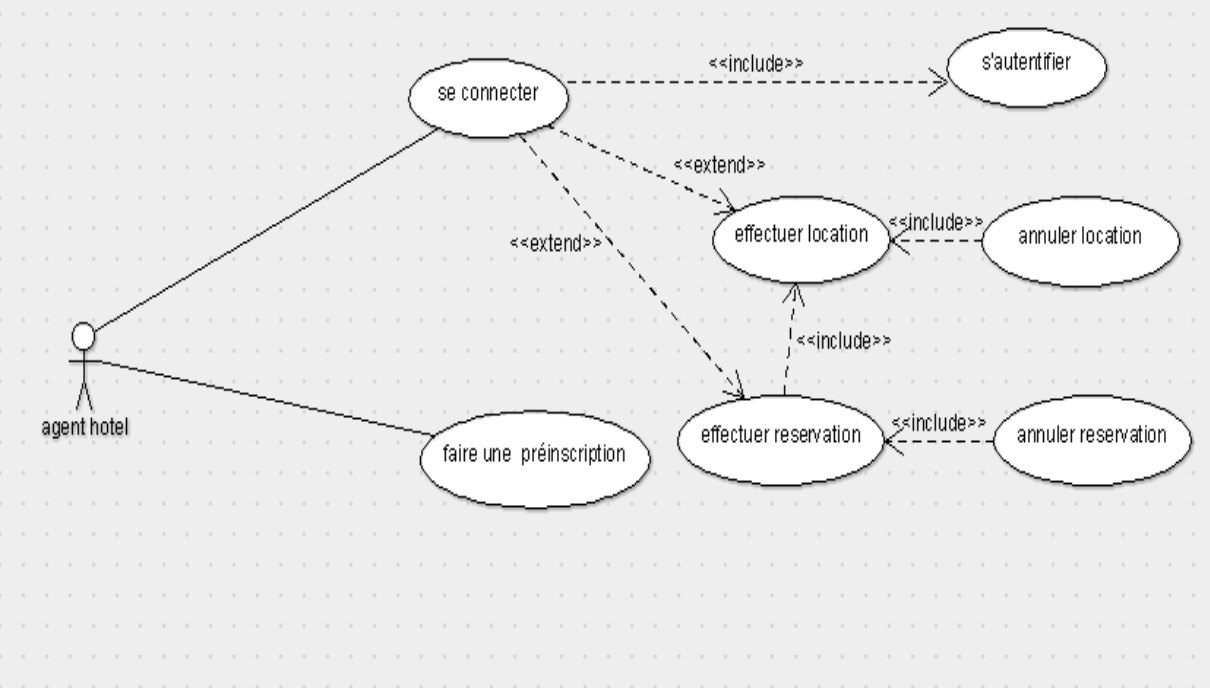


Figure 4.5 . Diagramme de cas d'utilisation pour « agent d'hôtel »

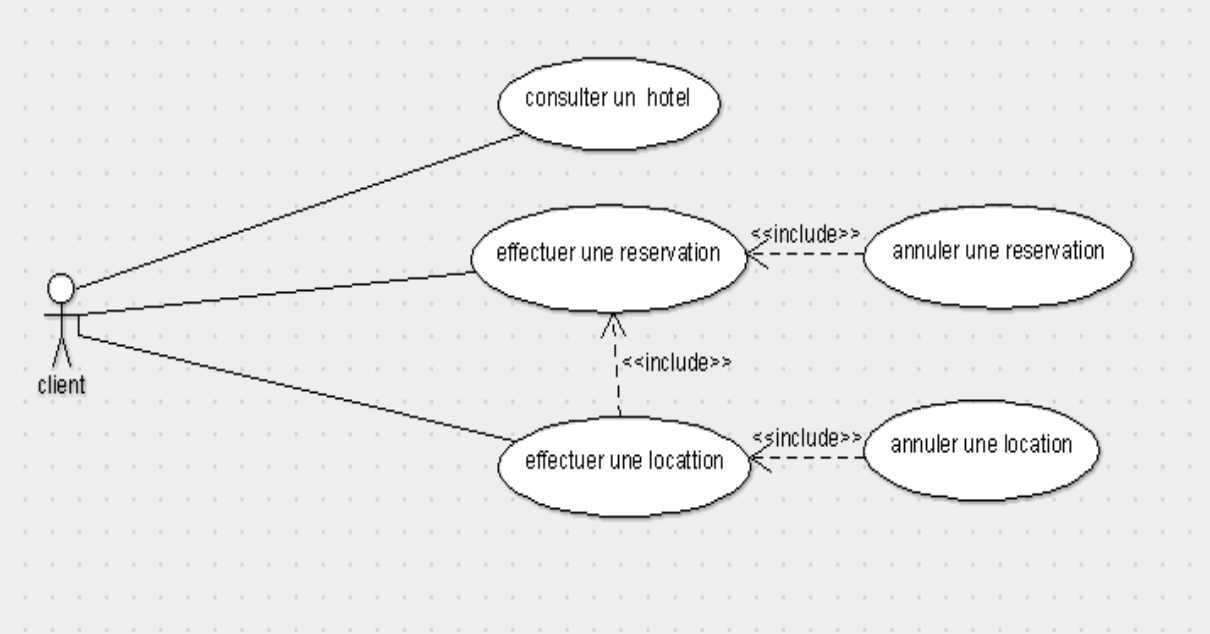


Figure 4.6 Diagramme de cas d'utilisation pour « client »

4.6 Phase de conception

A partir des diagrammes de cas utilisation, nous avons construit les diagrammes de séquences et d'activité qui montrent la vue dynamique du système. Le diagramme de classe quant à lui montre la vue statique de celui-ci.

4.6.1 Les Diagrammes de séquence

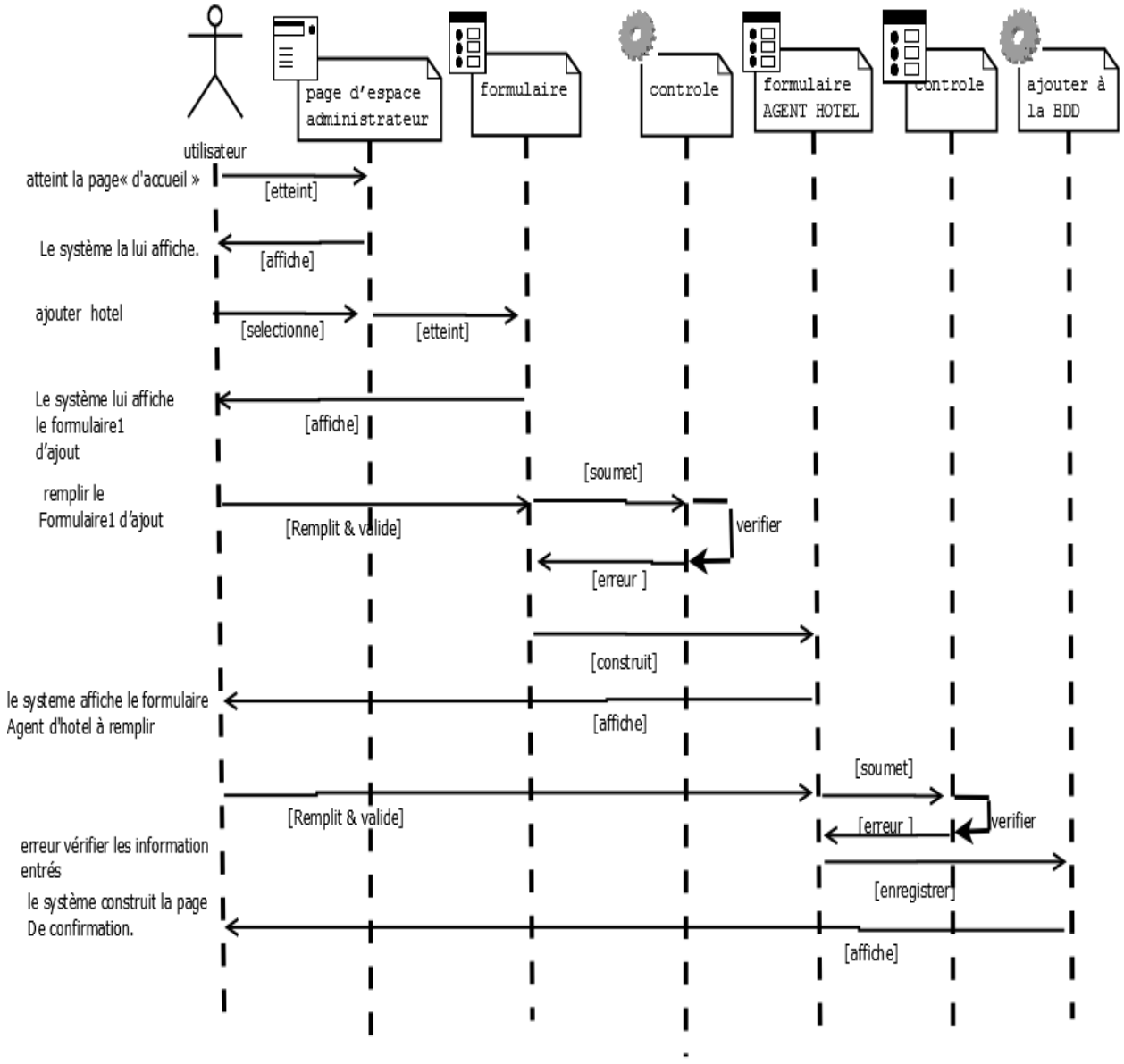


Figure 4.7 : Diagramme se séquence pour le cas d'utilisation « ajouter un hôtel »

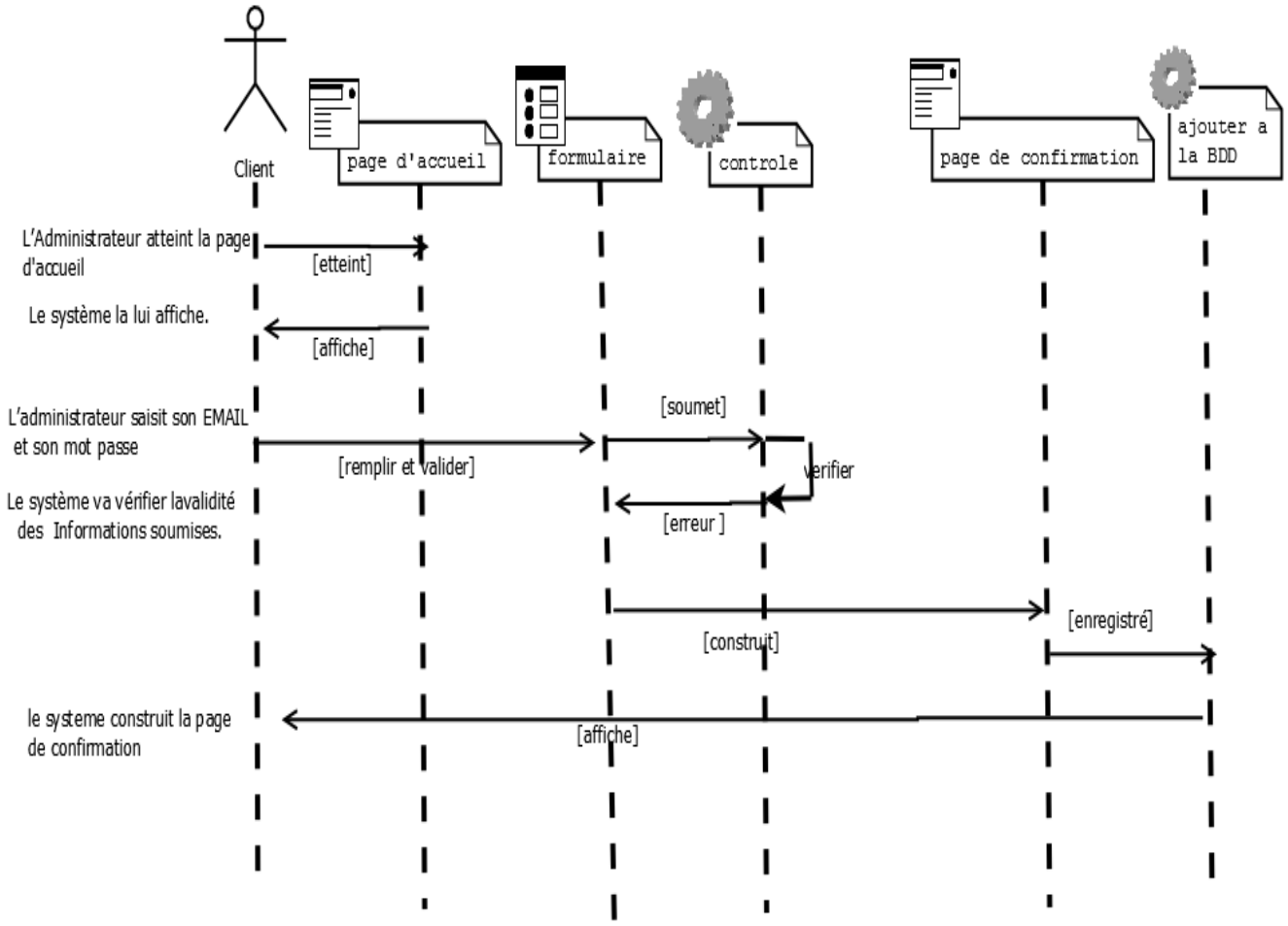


Figure 4.8 : Diagramme de séquence pour le cas d'utilisation « annuler réservation »

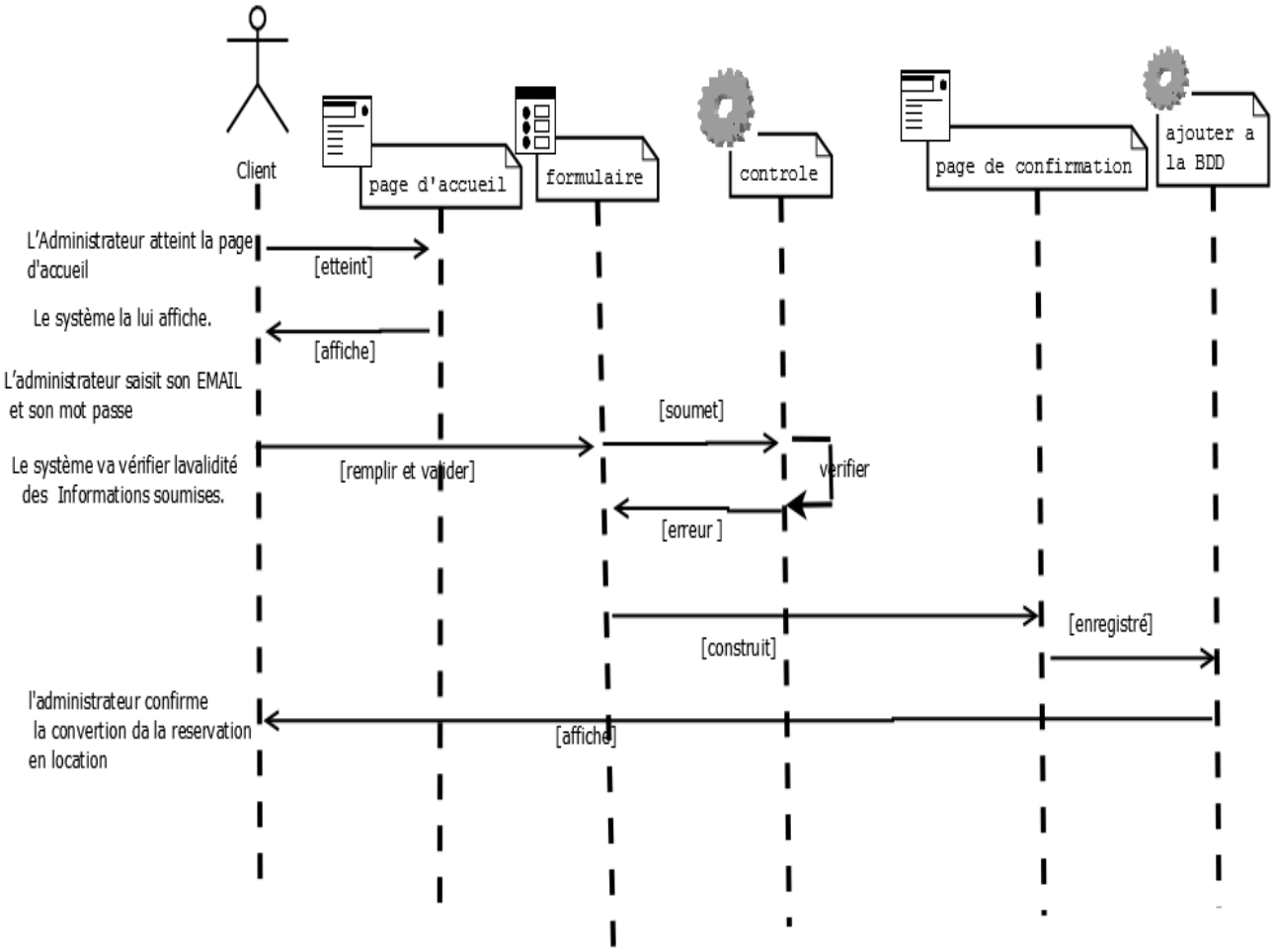


Figure 4.9 : Diagramme de séquence pour le cas d'utilisation « Convertir réservation »

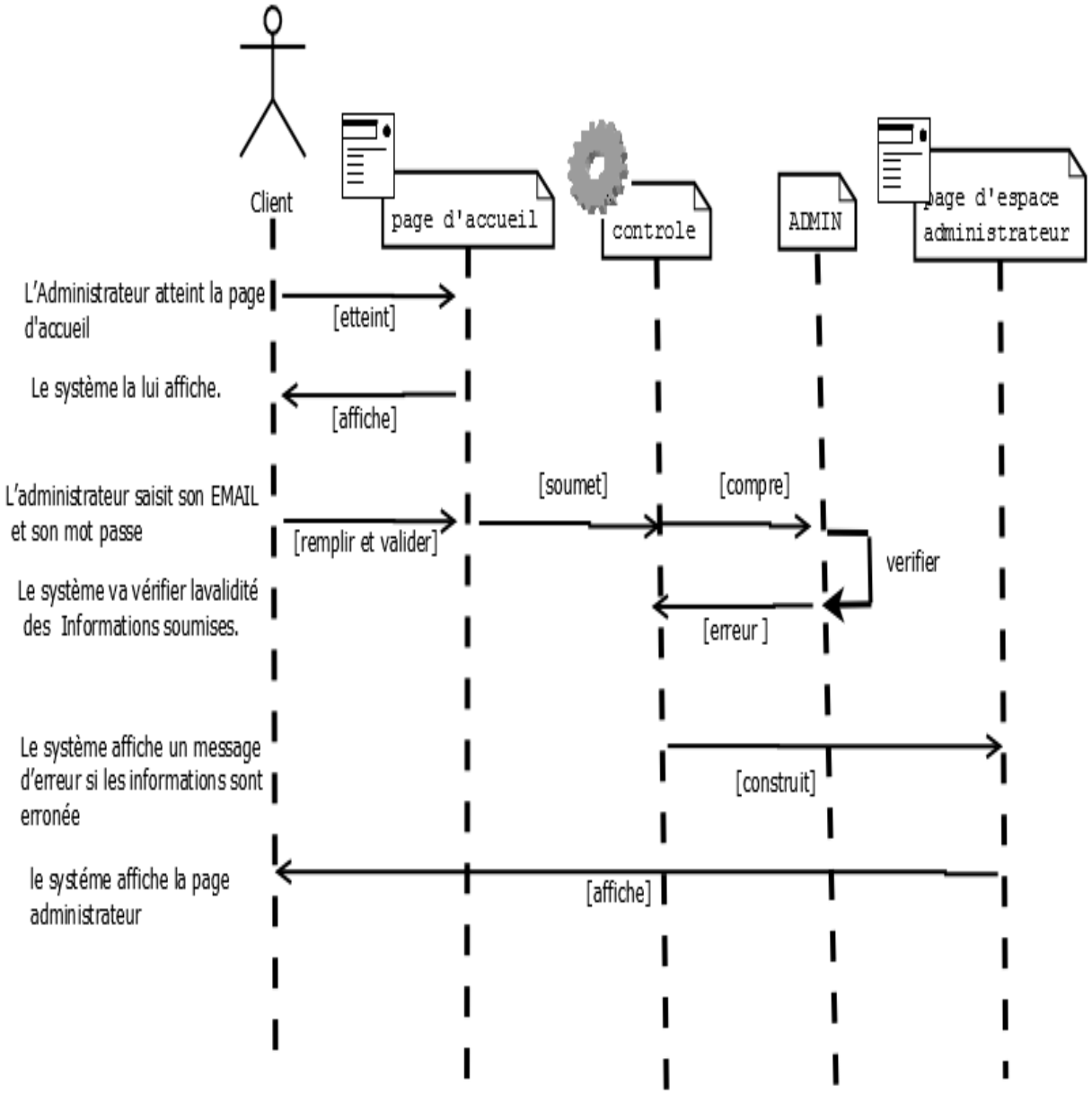


Figure 4. 10 : Diagramme de séquence pour le cas d'utilisation « authentification »

4.6.2 Les diagrammes d'activités

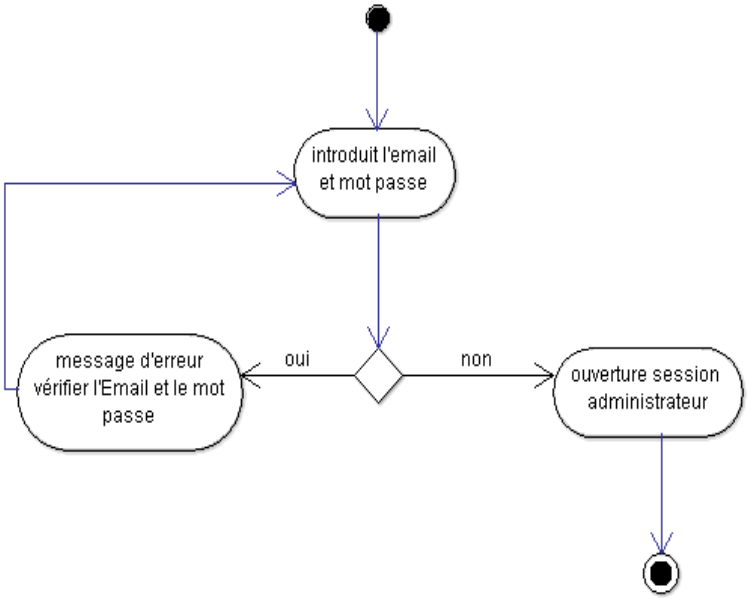


Figure 4. 11 : Diagramme d'activité « authentification»

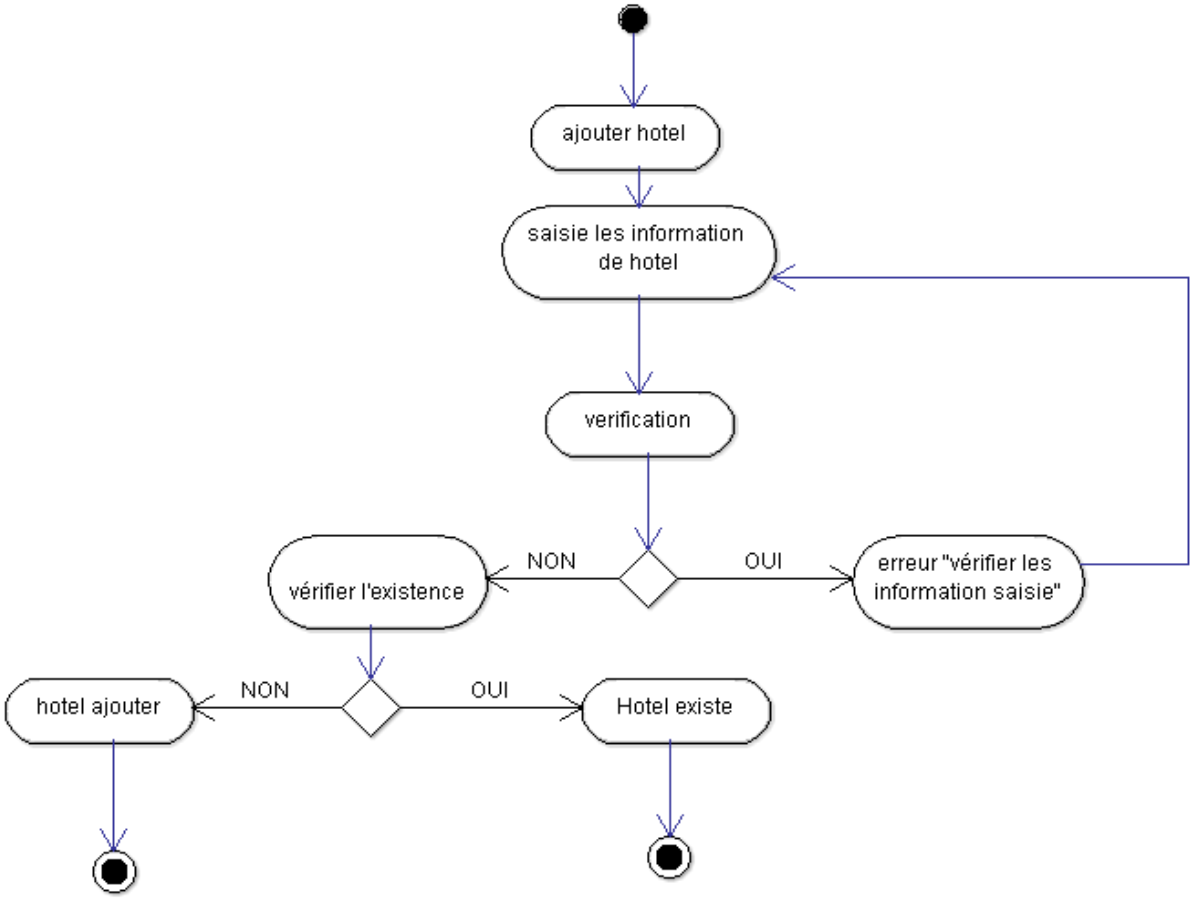


Figure 4 .12. Diagramme d'activité « Ajouter Hôtel »

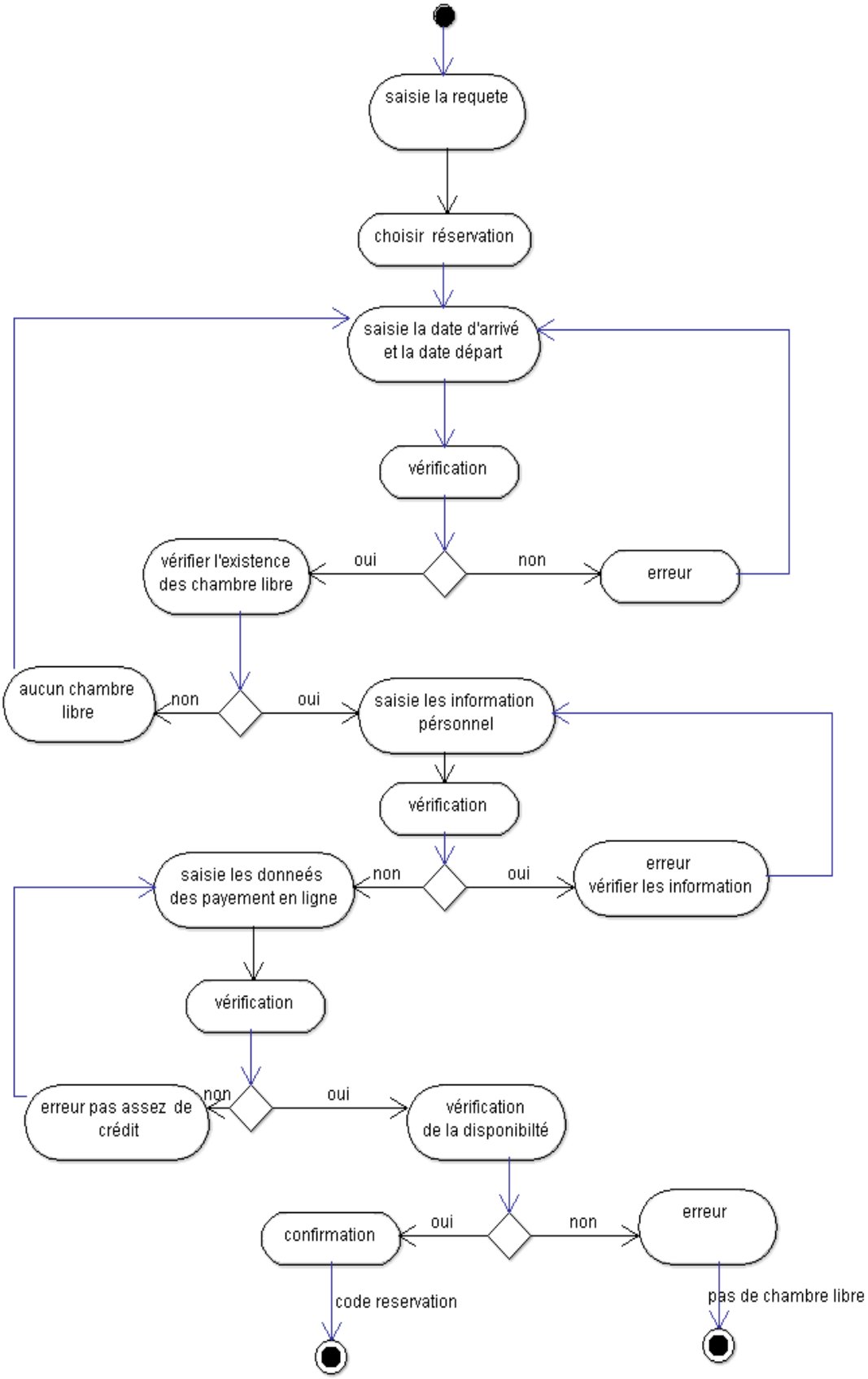


Figure 4. 13. Diagramme d'activité « effectuer réservations »

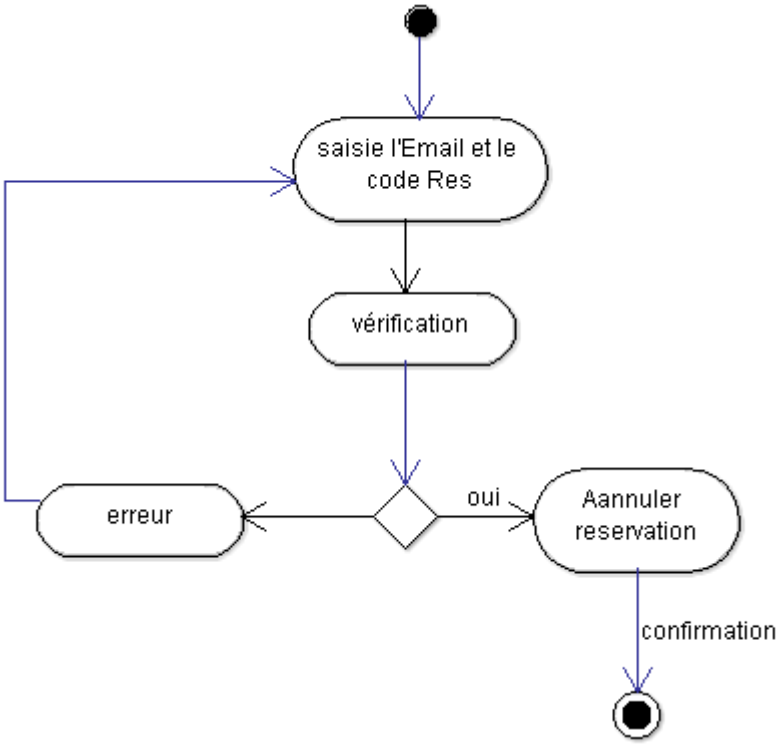


Figure 4. 14. Diagramme d'activité « annuler réservations »

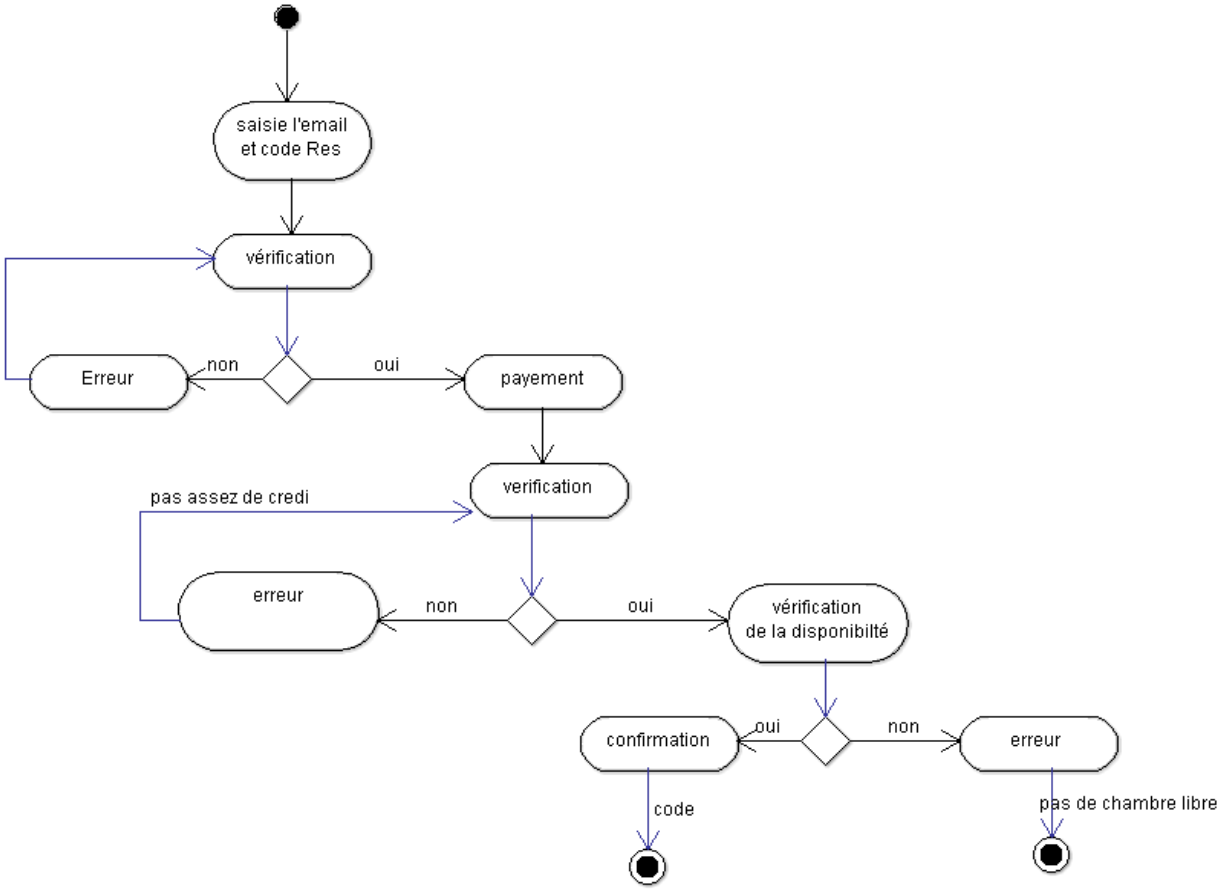


Figure 4. 15. Diagramme d'activité « convertir réservations »

4.6.3 Le diagramme de Classe

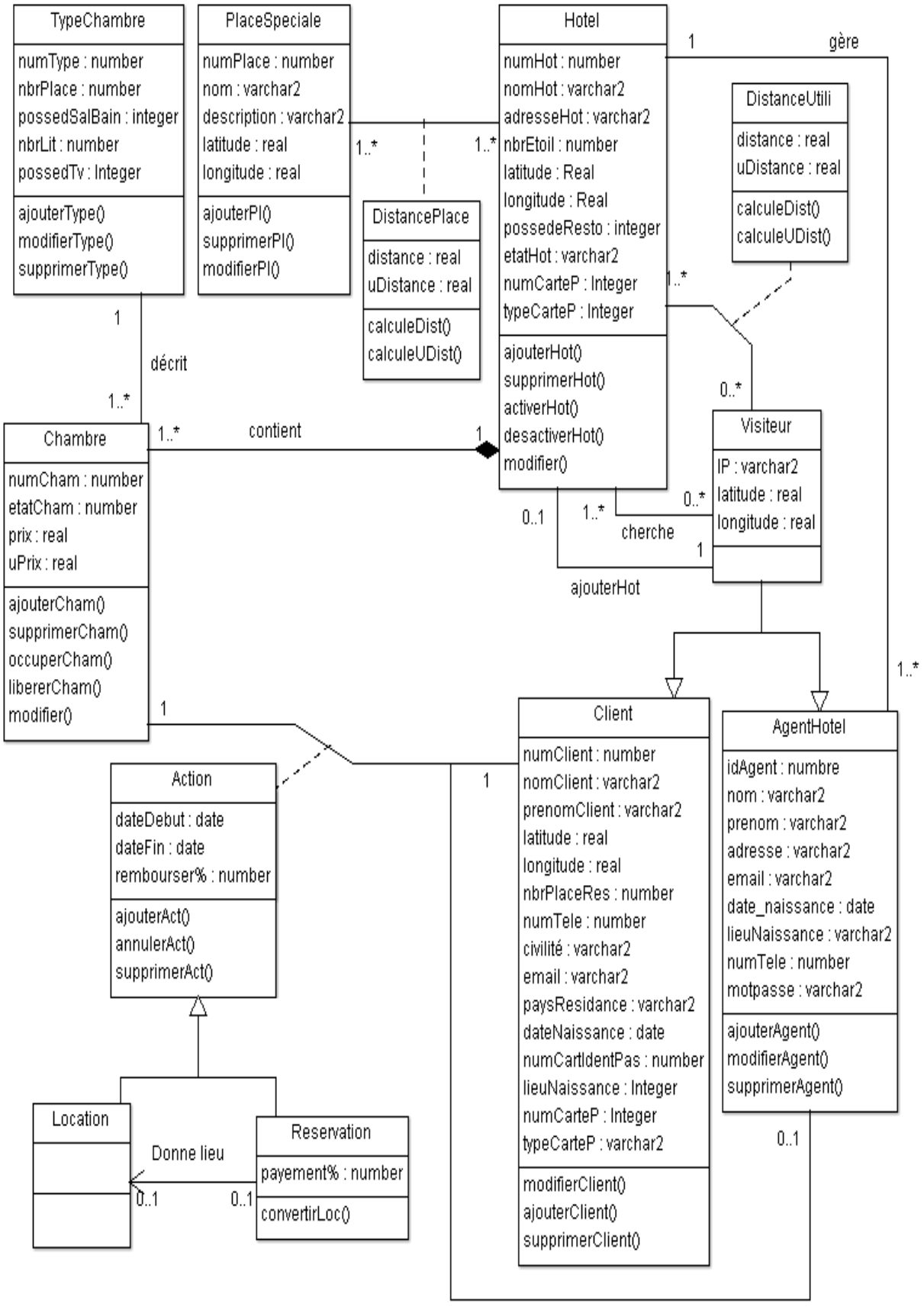


Figure 4.11 . Diagramme de classe général

4.7 Conclusion

A travers les deux phases d'analyse et de conception, nous avons pu tirer les principales fonctions, ainsi que le schéma conceptuel de notre future système. Nous avons pu aussi cerner la dynamique entre les différents acteurs et le système. Le chapitre suivant va porté sur l'implémentation de l'application.

Chapitre 5

Réalisation de l'Application

5.1 Introduction

Dans ce dernier chapitre consacré à la réalisation de l'application, nous allons commencer par présenter les outils et langages pour lesquels nous avons opté pour développer notre système. Beaucoup de travail a été fait en amant avant d'arriver au résultat final, notamment la création de la base de données, le calcul automatique des fonctions d'appartenances qui définissent les prédicats flous spécifiques à notre champ d'étude, le moteur de recherche qui est basée principalement sur une analyse lexicale, syntaxique et sémantique de la requête, ..etc.

5.2 Outils et langage de développement

Le tableau suivant montre les outils et langages utilisés :

Outil	**Version**	**Langages/ serveur**	**Description**	**Raisons d'utilisation**
Windows	Seven		Système d'exploitation	Plate-forme de développement
Oracle	10 g eXpress Edition	SQL	Langage de requête structuré	-créations des tables -définition des contraintes d'intégrités -manipulation des données -gestion des transactions -traitements internes par les triggers
		PL-SQL	Langage procédural	-Création des fonctions et procédures. -regroupement des fonctions et procédures dans des packages. -chargement des données à partir des tables via des curseurs
		Rigular Expression	Les expressions régulières	-l'analyse syntaxique -l'analyse sémantique

				-construction des requêtes booléennes
ArgoUML	0.34	UML (v2)	Langage de modélisation orienté-objet unifié (standardisé)	- Formalisation des diagrammes.
Macromedia Dreamweaver	8	HTML	Langage coté client	-construction des pages web statiques
		PHP	Langage coté serveur	-construction des pages dynamique -connexion à la base de données. -appel aux fonctions et procédures définies dans les package de PL/SQL
		CSS	Feuille de style en cascade	-pour le design des pages web
		JavaScript	Langage de script conté client	-manipuler les cartes et retourner les coordonnées d'un utilisateur (simuler la géo-localisation)
Xampp	1.7.3	Apache	Serveur internet	Exécuter le code PHP
		Oci8	Pilote oracle	Connexion entre PHP et Oracle
Notepad++	5.6.8		Editeur de code professionnel	Editer les codes : SQL, PL/SQL, Expressions Régulières
Excel	2007		Editeur pour les statistiques	Faire des statistiques pour choisir la meilleur norme et co-norme.

Google Earth	2012		Système de cartographie international	-Prise des cartes de la ville de Tizi-Ouzou. -Prise des coordonnées des hôtels et places spéciales

5.3 Mise en œuvre de la base de données

Le passage vers le modèle Relationnel donne lieu au modèle logique de données (MLD) suivant (voir l'annexe A pour les règles de passage) :

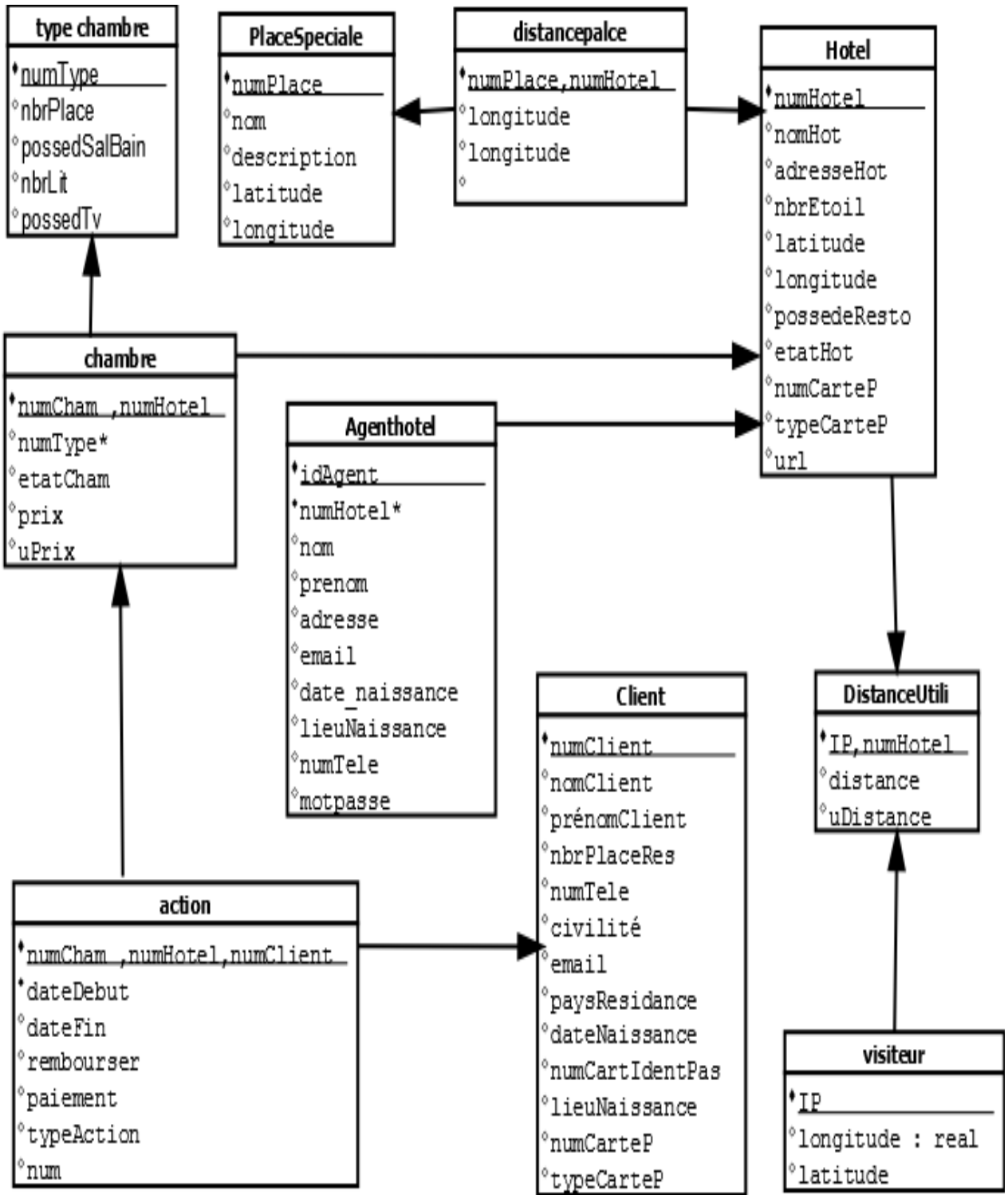


Figure 5.1 : Modèle logique de données (MLD)

5.3.1 Le schéma relationnel est donné ci-dessous :

- **Hôtel** (numHot , nomHot ,adresse ,nbrEtoil , latitude , longitude ,possedeResto,etatHot ,numCarteP , typeCarteP ,url)
- **Type Chambre (**numType ,nbrPlace ,nbrLit , possedSalBain , possedTv**)**
- **chambre (** numCham , numHot , numType* ,prix , uPrix , etatCham **)**
- **placeSpeciale (**numPlace , nom , description ,latitude , longitude **)**
- **distancePlace (** numHot, numPlace ,distance ,uDistance **)**
- **agentHotel (** idAgent, nom , prenom , adresse , email , date_naissance , lieu_naissance, numTele, motpasse , numHot* **)**
- **client (** numClient, nomClient ,prenomClient, nbrPlaceRes , numTele ,civilite , email , numCarte , typeCarte **)**
- **action(***numCham, numClient, numHot* , dateDebut, dateFin ,remboursement , payement , TypeAction **)**
- **visiteur (** ip ,latitude ,longitude**)**
- **distanceUtili (** numHot , ip ,distance ,uDistance **)**

- Les attributs soulignés désignent une clé primaire de la relation.
- Les attributs marqués par une étoile désignent des clés étrangères.

5.3.2 Le dictionnaire de données

Le tableau suivant représente le dictionnaire de données :

Nom de la rubrique	***Codification***	***type***	***longueur***
numHot	*numHot*	*N*	*3*
Nom Hôtel	*nomHot*	*A*	*30*
Adresse hôtel	*adresseHotel*	*N*	*30*
Nombre étoile	*nbrEtoil*	*N*	*1*
Possède restaurant	*possedeResto*	*A*	*3*
Etat hôtel	*etatHot*	*A*	*10*
Nombre de lit	*Nbr lit*	*N*	*1*
Possède une sale de bain	*possedSalBain*	*A*	*3*
Numéro de téléphone	*numTele*	*N*	*15*
Numéro de client	*numCl*	*N*	*12*

Tableau 5.1 : dictionnaire de données

5.4 Fonctions d'appartenance

On va étudier un exemple de la fonction d'appartenance du critère flou « bon-marché » (ou bien moins cher) puis en citera à la fin les autres critères flous.

Soit uPrix : X ⟶ [0,1]

x ⟶ uPrix(x) la fonction d'appartenance pour le critère « bon-marché»

La figure 5.2 montre la courbe de cette fonction :

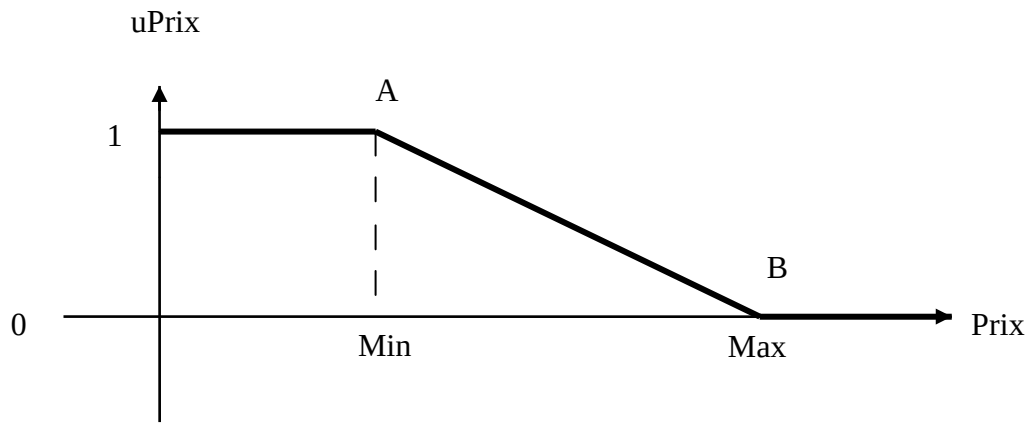


Figure 5.2 : fonction d'appartenance du prédicat bon marché

Tel que Min est le prix minimum des chambres d'un même type.

Max est le prix maximum des chambres d'un même type.

D'après le diagramme de la figure ci-dessus et la plage de valeurs de la variable prix ,on constate que uPrix prendra ses valeurs dans l'intervalle [0,1].

On constate bien que dans cet intervalle unité, uPrix est vu comme étant le segment [AB]

Calculons l'équation de la droite [AB] :

La forme générale de cette équation est : $\mathbf{y = \alpha x + \beta}$ **(1)**

tels que :

y=uPrix

x=Prix

Calculons α

$\boldsymbol{\alpha}$ représente la pente du segment [AB]

$$\boldsymbol{\alpha = \frac{\Delta y}{\Delta x} = \frac{y2-y1}{x2-x1} = \frac{0-1}{max-min} = \frac{-1}{max-min}} \qquad \textbf{(2)}$$

Calculons β

En remplaçons les coordonnées de **B** (Max, 0) dans l'équation (1)on obtient **:**

$$\mathbf{0 = \frac{-max}{max-min} + \beta}$$

donc $\boldsymbol{\beta = \frac{max}{max-min}}$ **(3)**

et en remplaçons les valeurs de $\boldsymbol{\alpha}$ et $\boldsymbol{\beta}$ dans l'équation générale on obtient :

$$\mathbf{uPrix = \frac{-prix}{max-min} + \frac{max}{max-min}}$$

$$\mathbf{uPrix = \frac{max-prix}{max-min}} \qquad \textbf{(4)}$$

Dans cette équation (4) on a deux problèmes :

- Le premier est que l'Hôtel ayant un prix Max aura Uprix=0 donc il sera suspendu, et puisque nous travaillons sur une dizaine d'hôtels alors nous voulons les prendre tous en considération.
- Le deuxième est que le trigger (déclencheur) qui calcule d'une manière automatique la valeur de la fonction d'appartenance se plante lors de la première insertion du prix d'une chambre, car dans cette première insertion le Max=Min donc il y'aura une

division sur 0. Pour remédier à cela, on a ajouté 100 DA à la valeur Max. Donc l'équation (4) devient :

$$\mathbf{uPrix} = \frac{\mathbf{max + 100 - prix}}{\mathbf{max + 100 - min}}$$

Ainsi, l'hôtel ayant un prix Max aura Uprix plus proche de zéro donc il est considéré dans les calcules, et le trigger ne se plantera jamais puisque lors de la première insertion : uPrix=100/100=1

Les fonctions d'appartenance pour les autres critères :

-la fonction d'appartenance pour le critère « cher » est uPrixCher

tel que **uPrixCher= 1-uPrix**

car uPrixCher est la fonciton inverse de la fonction uPrix.

-La fonction d'appartenance pour le critère « proche » est uDistance

tel que : $\mathbf{uDistance} = \frac{\mathbf{max+10-prix}}{\mathbf{max+10-min}}$

Pour éviter la division à zéro, on a ajouté 10 mètres à la valeur Max.

-La fonction d'appartenance pour le critère « loin » est uDistanceLoin

tel que : **uDistanceLoin=1-uDistance**

Les Modificateurs d'un critère flou :

On a utilisé les modificateurs flous suivants : « très » , « plus » et « moins »

Les modificateurs intensifient les critères flous entre eux en faisant la puissance de ces derniers.

Exemple : la valeur de « plus cher » est $(uPrix)^2$=uPrix*uPrix

Si l'hotel H1 a : uPrix=0.8 , alors $(uPrix)^2$=0.64 , et la différence entre uPrix et $(uPrix)^2$ est 0.16

Si l'hotel H2 a : uPrix=0.5 alors $(uPrix)^2$=0.25 et la différence entre uPrix et $(uPrix)^2$ est 0.25

D'après ces deux différences on constate que H1 est devenu plus important que H2 .

5.5 Choix de la norme et co-norme

Il existe une multitude de normes et de co-normes dont les principales sont citées dans le chapitre 2 sur les ensembles flous. Le choix du couple (norme / co-norme) dépend du contexte d'utilisation et de la pertinence souhaitée, et la **figure 5.3** montre les statistiques qu'on a faites pour le choix de la norme et de la co-norme convenables à notre cas d'étude.

H7 : =SI(G7=F7;A7;SI(G7=F8;A8;A9))

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Zadeh	distance	prix	uDistance	uPrix	min(B,C)	ordre	meilleur	diff	max(B,C)	maxmax	meilleur	diff
2	Hotel1	1440	1900	0.6	0.9	0.6	0.65	Hotel3	0.03	0.9	0.9	Hotel1	0.2
3	Hotel2	1388	4150	0.62	0.65	0.62	0.62	Hotel2	0.02	0.65	0.7	Hotel3	0.05
4	Hotel3	1310	3700	0.65	0.7	0.65	0.6	Hotel1	0.025	0.7	0.65	Hotel2	0.125
5													
6	probabliste	distance	prix	uDistance	uPrix	xy	max	meilleur		x+y-xy	max	meilleur	
7	Hotel1	1440	1900	0.6	0.9	0.54	0.54	Hotel1	0.085	0.96	0.96	Hotel1	0.065
8	Hotel2	1388	4150	0.62	0.65	0.403	0.455	Hotel3	0.052	0.867	0.895	Hotel3	0.028
9	Hotel3	1310	3700	0.65	0.7	0.455	0.403	Hotel2	0.069	0.895	0.867	Hotel2	0.047
10													
11	Lukasiewicz	distance	prix	uDistance	uPrix	max(x+y-1,0)	ordre	meilleur		min(x+y,1	max	meilleur	
12	Hotel1	1440	1900	0.6	0.9	0.5	0.5	Hotel1	0.15	1	1	Hotel1	0
13	Hotel2	1388	4150	0.62	0.65	0.27	0.35	Hotel3	0.08	1	1	Hotel2	0
14	Hotel3	1310	3700	0.65	0.7	0.35	0.27	Hotel2	0.115	1	1	Hotel3	0
15													
16	Hamacher	distance	prix	uDistance	uPrix	xy/(k+(1-k)(:	ordre	meilleur		(x+y-xy-(:	max	meilleur	0.5
17	Hotel1	1440	1900	0.6	0.9	0.551020408	0.55102	Hotel1	0.071	0.945205	0.94521	Hotel1	0.081
18	Hotel2	1388	4150	0.62	0.65	0.431708623	0.48021	Hotel3	0.049	0.833438	0.86408	Hotel3	0.031
19	Hotel3	1310	3700	0.65	0.7	0.480211082	0.43171	Hotel2	0.06	0.864078	0.83344	Hotel2	0.056
20													

figure 5.3 statistique pour trouver la meilleur norme et co-norme

A	B	C	D	E	F	G	H	I
Zadeh	distanc	prix	uDistance	uPrix	min(B,C)	maxmin	meilleur	différence
Hotel1	1440	1900	v=(3000-B2)/(2600)	0.9	v=MIN(D2:E2)	v=MAX(F2:F4)	v=SI(G2=F2;A2;SI(G2=F3;A3;A4))	v=G2-G3
Hotel2	1388	4150	v=(3000-B3)/(2600)	0.65	v=MIN(D3:E3)	v=SI(G2=F2;MAX(F3;F4);SI(G2=F3;MAX(F2;F4);MAX(F2;F3)))	v=SI(G3=F3;A3;SI(G3=F2;A2;A4))	v=G3-G4
Hotel3	1310	3700	v=(3000-B4)/(2600)	0.7	v=MIN(D4:E4)	v=MIN(F2;F3;F4)	v=SI(G4=F4;A4;SI(G4=F3;A3;A2))	v=MOYENNE(I2;I3)

Figure 5.4 : formules utilisées pour la norme Min(x,y) de Zadeh

Discussion des statistiques:

1. choix de la norme :

La figure 5.4 montre les formules utilisées dans les statistique de la figure 22 pour la norme min(x,y) Zadeh, et pour les autres normes et co-normes, on suit les mêmes formules en changeant seulement celle de la norme min(x,y).

Toutes les normes sauf celle de Zadeh retournent les mêmes résultats, les meilleurs hôtels sont classées ainsi: Hotel1,Hotel3 puis Hotels2 et donc ce résultat est pertinent.

La norme de Zadeh bien qu'elle soit simple, nous retourne un résultat non pertinent, car elle s'intéresse seulement aux MINs des critères entrés, mais pas à la relation entre eux, cette relation est très utiles pour notre cas d'études, donc cette normes est rejetée, cependant cette norme est utile si on veut retourner des tuples dominants pour une requête qui retourne des milliers de tuples, et pour notre cas, on retourne seulement une dizaine de tuples.

Mais qu'elle est la meilleur entre les 3 autres normes restantes ??

Pour faire ce choix, on s'intéressera à la relation entre les tuples retourné, plus que la différence entre ces tuples est assez grande plus que les tuples pertinents sont mis en valeur et facile à sélectionner, par exemple pour la norme probabiliste, la différence entre le tuple Hotel1 et hôtels 3 est : I7=G7-G8=0.085 , hotel3 et hotel2 est : I8=G8-G9=0.052 , la moyenne entre c'est différence est : I9=MOENNE(I7,I8)=0.069.

Si on classe les normes selon la moyennes des différences de plus grand au plus petit on obtient :

Lukasiewicz=0.115

Probabiliste=0.069

Hamacher=0.6

Mais, celle de Lukasiewicz a un inconvénient majeur, c'est que qu'on a deux critère x et y tels que x+y<=1 , alors la valeur du la norme max(x+y-1,0)=0,et on ajoutant un autre critère z=0.99 on obtient : max(max(x+y-1,0)+z-1,0)=max(0+z-1,0)=0, alors cela va entrainer l'absorption du tuple à la valeur zéro , donc il n'est pas utile pour notre cas.

Donc la norme Probabiliste est la meilleur pour notre cas et d'ailleurs elle est la plus simple que celle de Hamacher.

2. Choix de co-norme :

Toutes les co-normes ont retourné le même résultat sauf celle de Lukasiewicz qui retourne un résultat aléatoire puisque elle absorbe les tuples à la valeur 1 dès qu'il y a deux critère dont leurs somme est 1.

Et en classant les autres selon la moyenne des différences on obtient :

Zadeh=0.125

Hamacher =0.056

Probabiliste=0.047

La meilleur est celle de Zadeh et d'ailleurs elle est la plus simple.

5.6 Choix de la formule de calcule de la distance entre deux points sur la carte

La formule qu'on a choisie pour le calcule de la distance entre deux points sur la carte est la suivante :

$$\textbf{Distance= R}\sqrt{(\textbf{latA}-\textbf{latB})^2+\left(\cos\left(\frac{\textbf{latA}+\textbf{latB}}{2}\right)*(\textbf{lonB}-\textbf{lonA})\right)^2}$$

tel que : R= 6368 x10^3 mètres : est le rayon moyen de notre Terre.

latA est latitude de A, latB est latitude de B (en degré à virgule décimale)

lonA est longitude de A ,lonB est longitude de B (en degré à virgule décimale)

Pour plus d'informations consulter l'annexe B.

5.7 Paiement en ligne

La figure suivante montre les transactions entre les clients, les Hôtels et le DBA.

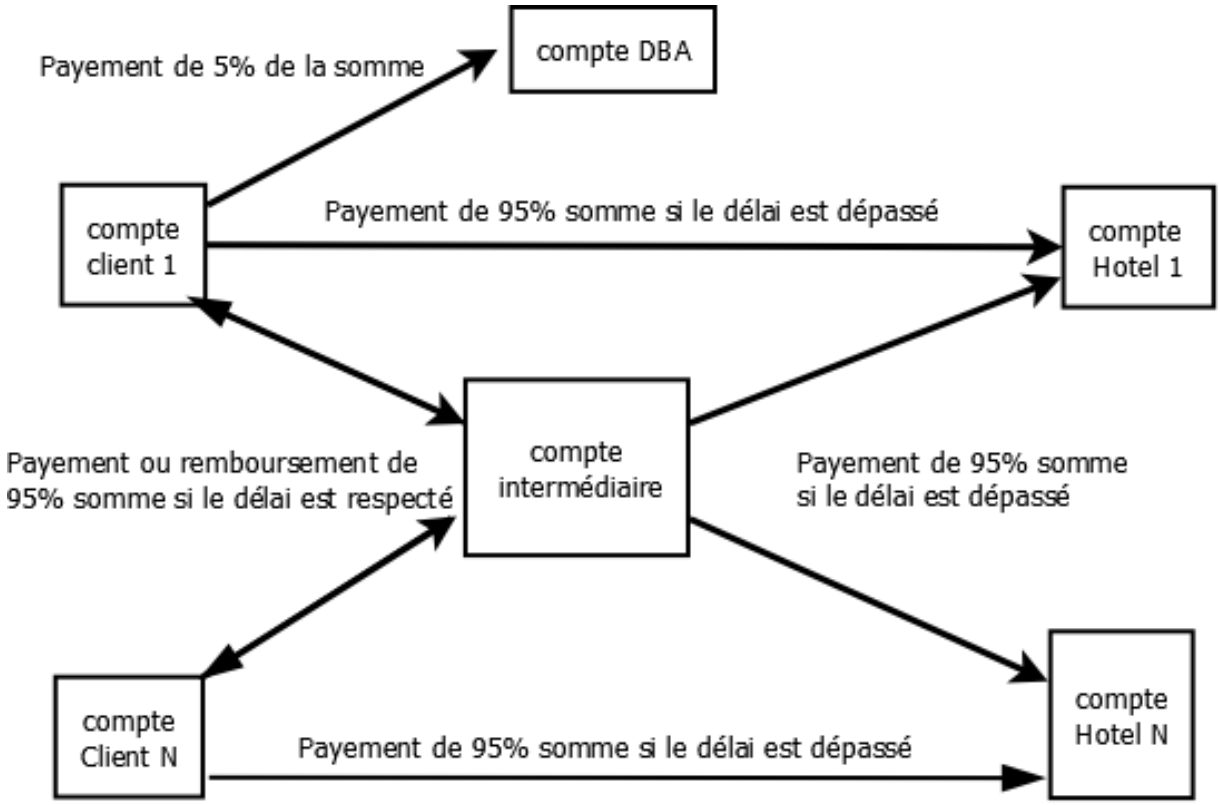


Figure 5.5 : Transactions de paiement en ligne

On a utilisé un compte intermédiaire pour pouvoir rembourser de l'argent aux clients si le délai est respecté. Sans ce compte, on sera obligé d'utiliser le code du compte des hôtels pour pouvoir les rembourser. Ce qui est impossible.
Le délai est fixé à 24 heures avant la date d'arrivée à l'hôtel.

5.8 Fonctionnement du Moteur de Recherche

Le moteur de recherche constitue le noyau de notre travail. Il est basé sur le principe de dérivation d'une requête exprimée en langage naturelle vers une requête booléenne en passant par le flou.

Exemple 5.1 :

Soit la requête naturelle R1 suivante : « ***hôtel le plut porche de la garre routières*** »

Remarque : *on n'a fait des erreurs exprès dans la requête pour que notre système les corrige.*

Dans ce qui suit, on va faire la transformation de R1 en requête booléenne R2 en passant par le flou. R2 est sous la forme suivante :

« **SELECT DISTINCT** *Affichage* **FROM** *Tables* **WHERE** *Critères* **ORDER BY** Pertinence **DESC**; »

Tels que :

- **Affichage :** ce sont les attributs que la requête doit retourner comme résultat.
- **Tables :** ce sont les tables que la requête va manipuler.
- **Critères :** ce sont les critères flous et booléens ainsi que les critères des jointures.
- **Pertinence :** résultat final des critères flous après avoir appliquer les normes et co-norme.

Le schéma suivant montre cette transformation.

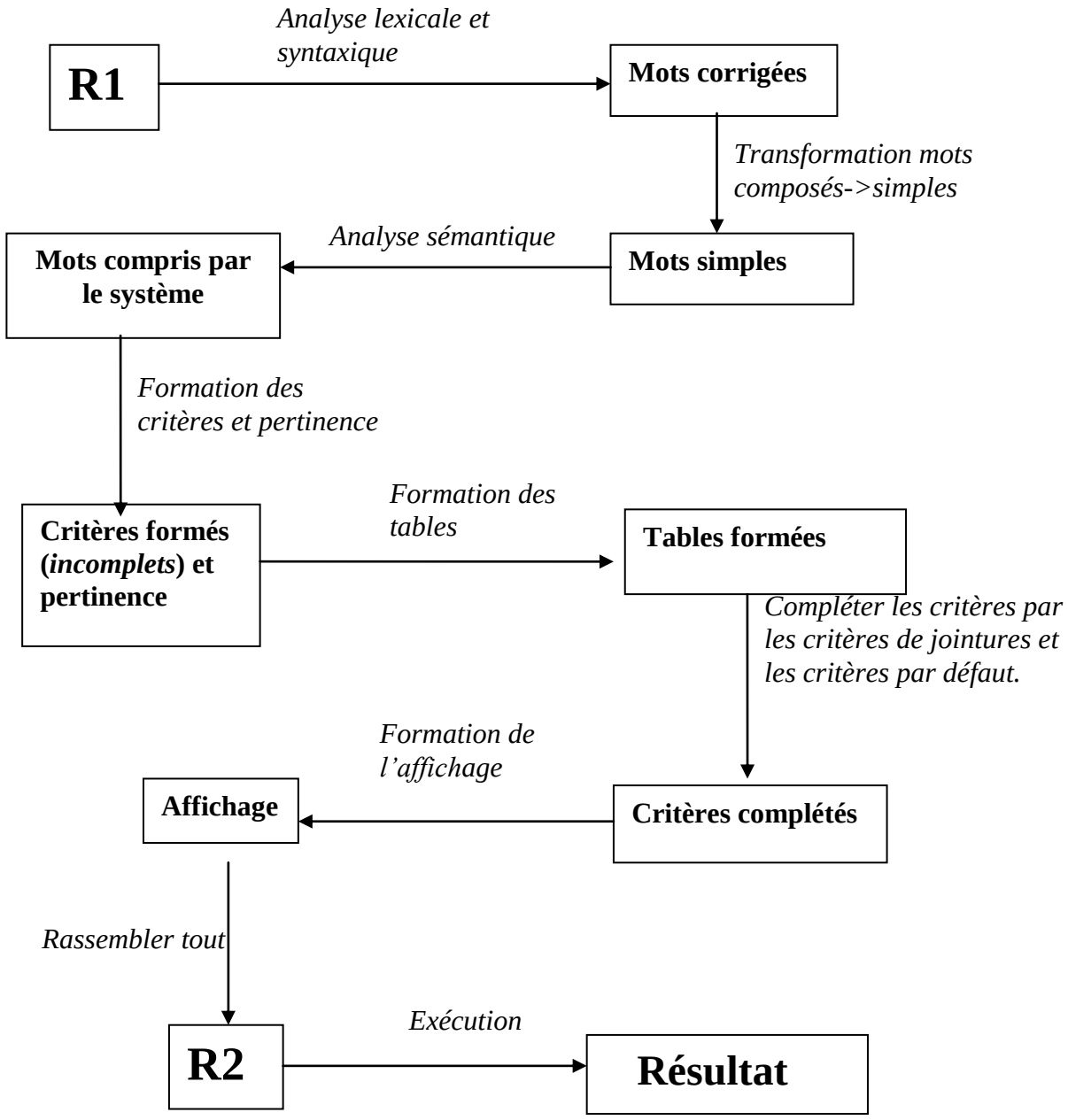


Figure 5.6 : Transformation de la requête naturelle en requête booléenne

Description des phases :

Soit les valeurs initiales suivantes :

R1= *' hôtel le plut porche de la garre routières'*

Affichage= 'h.nomHot,h.adresseHotel,h.nbrEtoil, h.possedeResto, tc.numType '

Tables = 'hotel h, typeChambre tc, chambre ch'

Critère= 'h.etatHot="actif"';
Pertinence='' ;(ou bien vide)

Phase1 : Analyse lexicale : il s'agit de découper R1 en un ensemble de termes (ou bien de mots), le séparateur c'est l'espace (le blanc) :
R1 devient l'ensemble suivant : (**hôtel, le, plut, porche, de, la, garre, routières**)

Phase2 Analyse syntaxique : il s'agit de corriger les erreurs d'orthographe (d'ordre au plus un seul caractère) selon un dictionnaire qu'on a défini. Les termes qu'on n'a pas pu corriger seront supprimés et on les considère comme des termes insignifiants.
Cette analyse se base sur la ressemblance entre ces termes et les mots du dictionnaire en considérant les termes et les mots comme étant un ensemble de caractère.
On dit que :
Mot1 ressemble à Mot2 si et seulement si longueur (Mot1)+longueur(Mot2)-2*nbr<=1
Tel que **nbr** est l'ensemble des caractères en commun entre mot1 et mot2 (ou bien l'intersection en terme de caractère).
Avec ce mécanisme, on peut corriger les erreurs de frappes, la permutation des caractères, pas de différence entre le pluriel et le singulier, ajouter, supprimer ou modifier un caractère…
R1 devient l'ensemble : **(plus, proche, la, gare, routière)**

Phase3. Transformation en mots simples : il s'agit de transformer les mots composés (pas vraiment au sens littéraires) en mots simples en utilisant une table de correspondance.
Pour cela, on délimite d'abord les mots de R1 par les tiré 6 (-)
Donc R1= -plus-proche-la-gare-routière-
Puis R1 deviendra :**-plus-proche-la-gareroutière-**

Phase4 .Analyse sémantique : ils s'agit de traduire les termes par d'autres termes compris par notre système en se basant sur un dictionnaire de synonymes propres à notre cas d'étude.
Ainsi, R1 devient : **-plus-proche-garemultimodale-**

Phase5. Formation des critères : il s'agit de former les critères (mais incomplet) à partir des termes de R1 en se basant sur une table de correspondance.
Ainsi , le terme « plus » n'a pas de critère puisque c'est un modificateur.

Le critère du terme « proche » est dp.uDistance>0 ;

Le critère du terme « garemultimodale » est ps.nom='garemultimodale' ;

Ainsi : **Critère**= 'h.etatHot="actif"' and dp.uDistance>0 and ps.nom="garemultimodale"';

Et **Pertinence**=(dp.uDistance)*(dp.uDistance), puisque il y a un modificateur (plus).

Phase 6. Formation des tables : il s'agit de déduire les tables à partir des alias des critères précédents et inclure les tables intermédiaires.

Tables = 'hotel h, typeChambre tc, chambre ch, distancePlace dp, placeSpeciale ps'

Phase 7.compléter les critère par des critères de jointure et les critères par défaut : c'est à partir des tables formées précédemment.

Critère= 'h.etatHot="actif"' and dp.uDistance>0 and ps.nom="garemultimodale" and tc.numType=ch.numType and ch.numHot=h.numHot and dp.numHot=h.numHot and ps.numPlace=dp.numPlace and tc.possedTV= "non" and tc.possedSalBain="non" and tc.nbrPlace=1'

Phase 8. formation de l'affichage : compléter par la pertinence

Affichage= 'h.nomHot,h.adresseHotel,h.nbrEtoil, h.possedeResto, tc.numType, (dp.uDistance)*(dp.uDistance)'

La figure 5.7 suivante montre la :

Phase 9 : Rassembler tout : il s'agit de d'arriver à **R2** (requête dérivée)

Phase 10 : Exécution : pour retourner le **Résultat** par ordre de pertinence

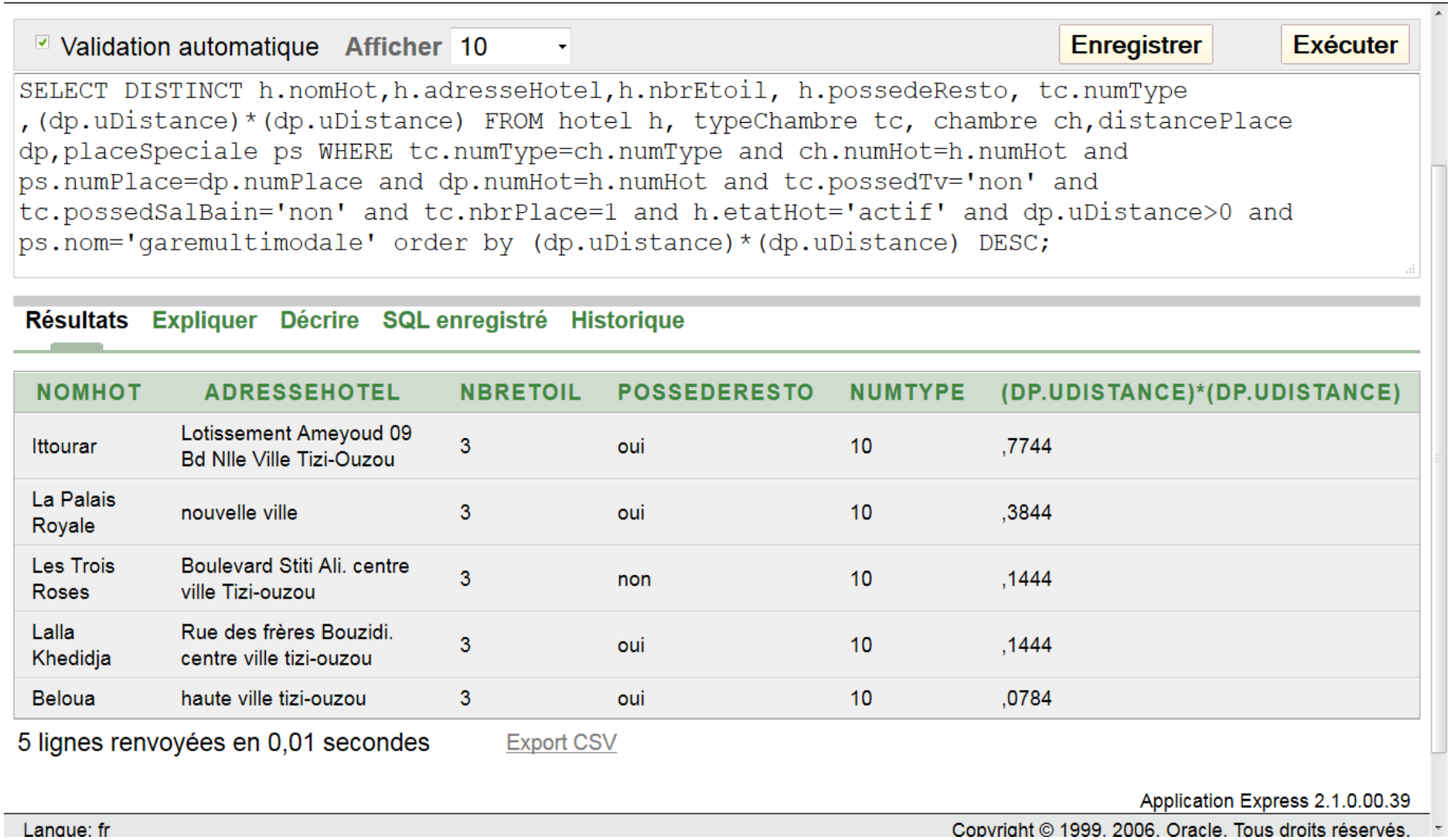


Figure 5.7 : La requête booléenne dérivée avec le résultat de son exécution

Voici un autre exemple un peu plus complexe :

R1=' je cherche l''hotel pour deu personn le plud pluz loinn du stade prremier noovembre ou le moins cheer posedant une sallle des bains est une tilivision' ;

La figure suivante montre la requête générée R2 :

```
SELECT DISTINCT h.nomHot,h.adresseHotel,h.nbrEtoil, h.possedeResto, tc.numType
,(1-dp.uDistance)*(1-dp.uDistance)*(1-dp.uDistance + ch.uPrix -(1-dp.uDistance * ch.uPrix))
FROM hotel h, typeChambre tc, chambre ch,distancePlace dp,placeSpeciale ps WHERE
tc.numType=ch.numType and ch.numHot=h.numHot and ps.numPlace=dp.numPlace and dp.numHot=h.numHot
and h.etatHot='actif' and tc.nbrPlace=2 and dp.uDistance<1 and ps.nom='stade1novembre' or
ch.uPrix>0 and tc.possedSalBain='oui' and tc.possedTv='oui' order by (1-dp.uDistance)
*(1-dp.uDistance)*(1-dp.uDistance + ch.uPrix -(1-dp.uDistance * ch.uPrix)) DESC
```

Figure 5.8 : La requête générée R2

Exemple du code source : ce code ci-dessous, fait l'analyse lexicale :

```
create or replace package com as--spécification
    function compar(x varchar2,y varchar2) return number ;
end com;
-----------------------------------------------------------------
create or replace package body com as
    function compar(x varchar2,y varchar2) return number is
    z varchar2(100);
    w varchar2(100);
    a varchar2(100);
    i varchar2(100);
    j varchar2(100);
    nbr number(2):=0;
    r number(2):=0;
    begin
        i:=x;
        j:=y;
        while length(i)>0 loop
            z:=regexp_substr(i,'[[:alnum:]]{1}');
            w:=regexp_replace(i,z,'',1,1);
            i:=w;
            if regexp_instr(j,z)>0 then
                nbr:=nbr+1;
                a:=regexp_replace(j,z,'',1,1);
                j:=a;
            end if;
        end loop;
        if length(x)+length(y)-2*nbr <=1 then
            r:=1;
        end if;
        return r;
    end compar;
end com;
```

Figure 5.9 : code source pour l'analyse lexicale

5. 9 Quelques Interfaces du site

Dans ce qui suit, nous allons présenter quelques captures d'écran de notre site.

En premier lieu, on présente un scénario de réservation, puis on présentera l'espace Agent Hôtel.

1. **Scénario de réservation :** un utilisateur se connecte et tape l'URL suivante : www.reservation-hotel.com , et la page suivante s'affiche.

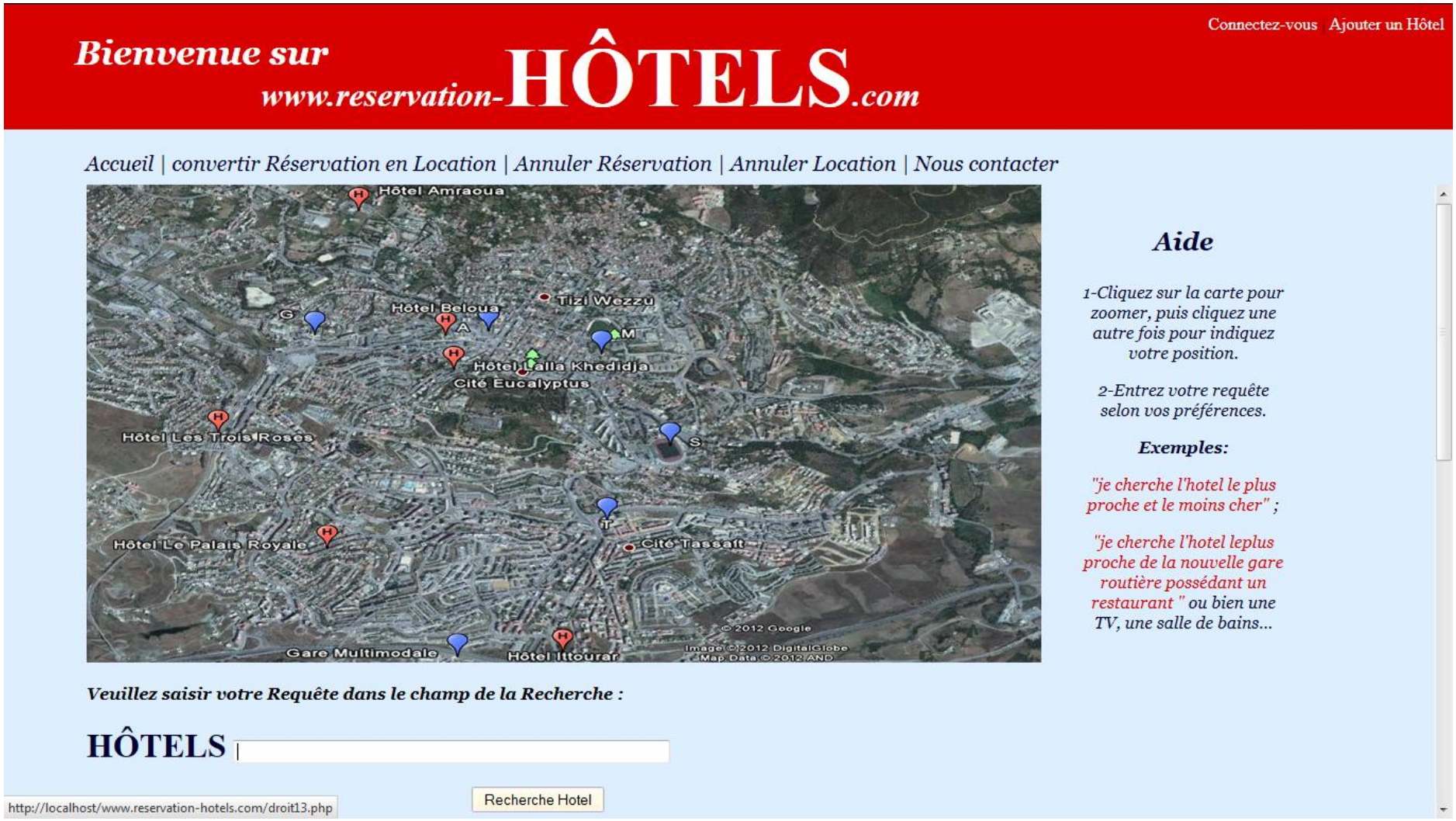


Figure 5.10 : Page d'accueil

C'est la Page d'accueil .Elle contient : un en-tête, une barre de menu, la carte de tizi-ouzou, un champ de recherche et une rubrique d'aide .

Par la suite, l'utilisateur saisit sa requête dans le champ de recherche et valide .La page Résultat s'affiche :

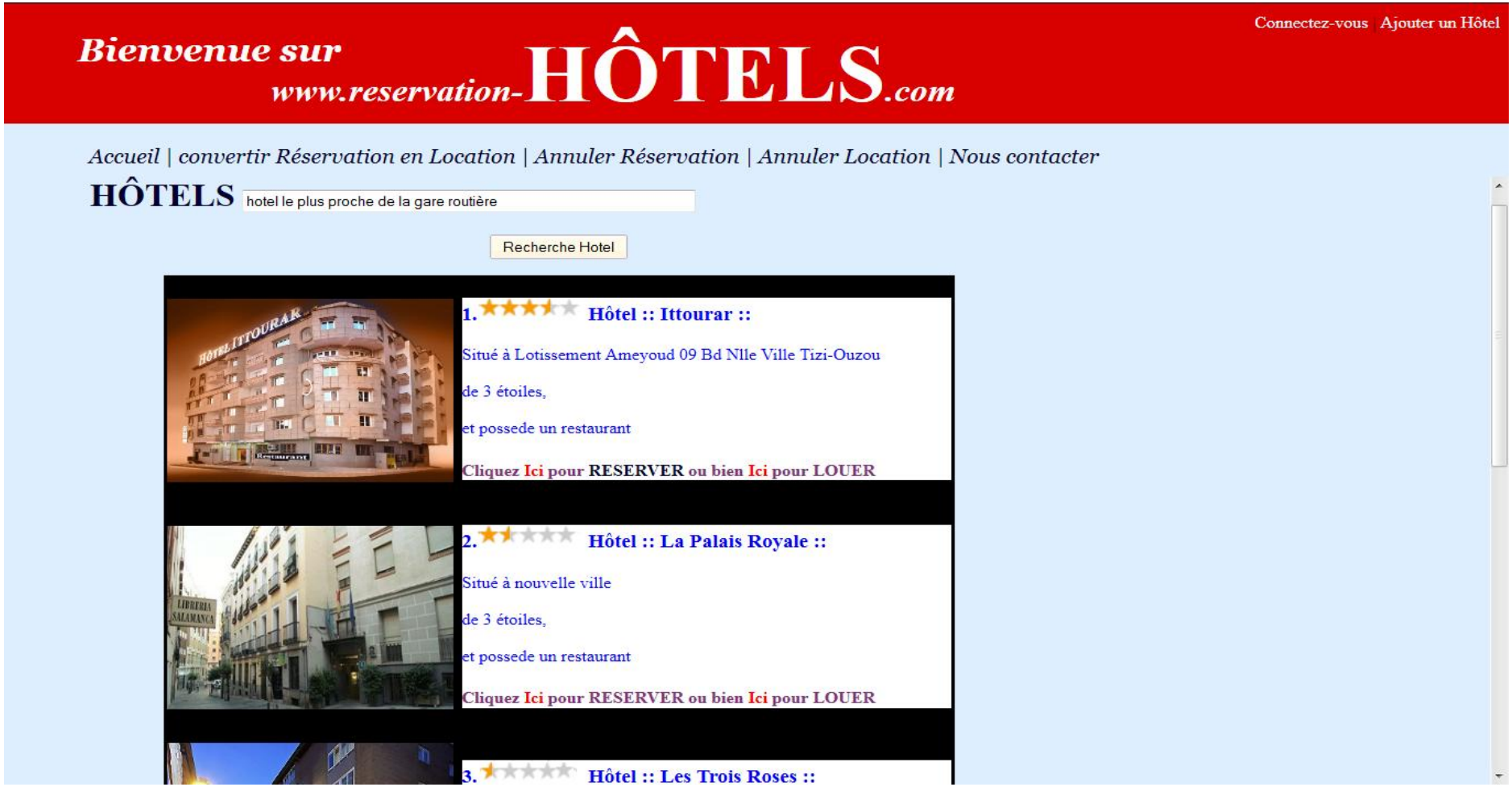


Figure 5.11 :Page résultat de la recherche

La recherche retourne des hôtels ordonnés selon leurs pertinences qu'on voit à travers les étoiles allumées, ainsi que quelques détails et deux lien : le premier pour la réservation et le deuxième pour la location. En cliquant sur le lien de réservation, la page suivante s'affiche :

Figure 5.12 : page date réservation

Le client choisit la date début et la date départ puis clique sur « vérifier disponibilité » et la page suivante s'affiche :

Figure 5.13 : Disponibilité des chambres

C'est la page qui indique le nombre de chambres libres. Le client clique sur « ici » pour continuer :

Figure 5.14 : page coordonnées du client

C'est le page où le client doit saisir ses coordonnées, et après avoir valider ce formulaire, la page Paiement s'affiche :

Figure 5.15 : page paiement en ligne

Après avoir affiché quelques règles, délais et détails pour la réservation, le client effectue la transaction de paiement.

Figure 5.16 : page de confirmation

C'est la page de confirmation de la réservation. Le client doit retenir le code pour le présenter une fois arrivé à l'hôtel. Il faut noter que ce code est envoyé au client par messagerie au cas de perte ou d'oubli.

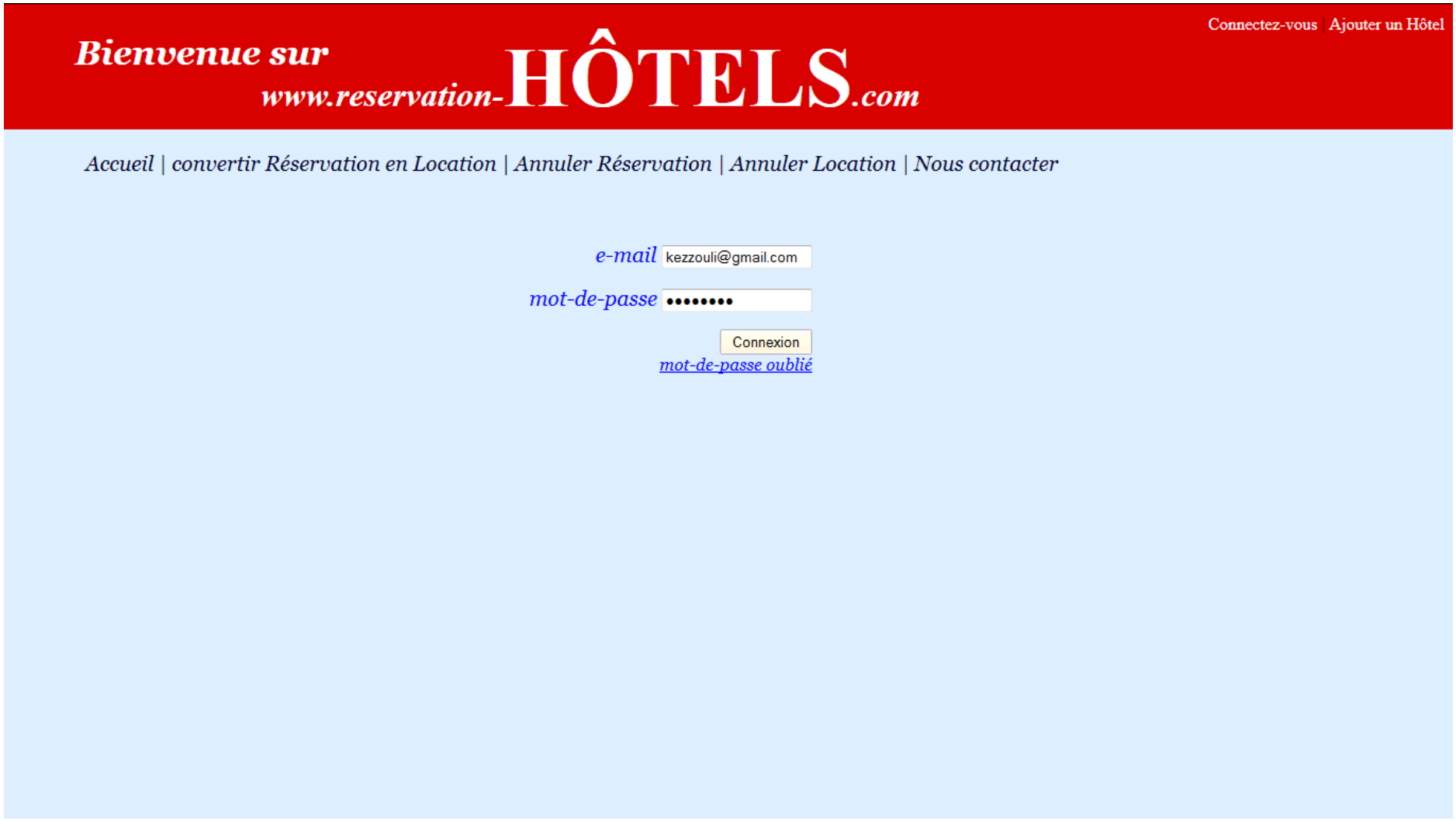
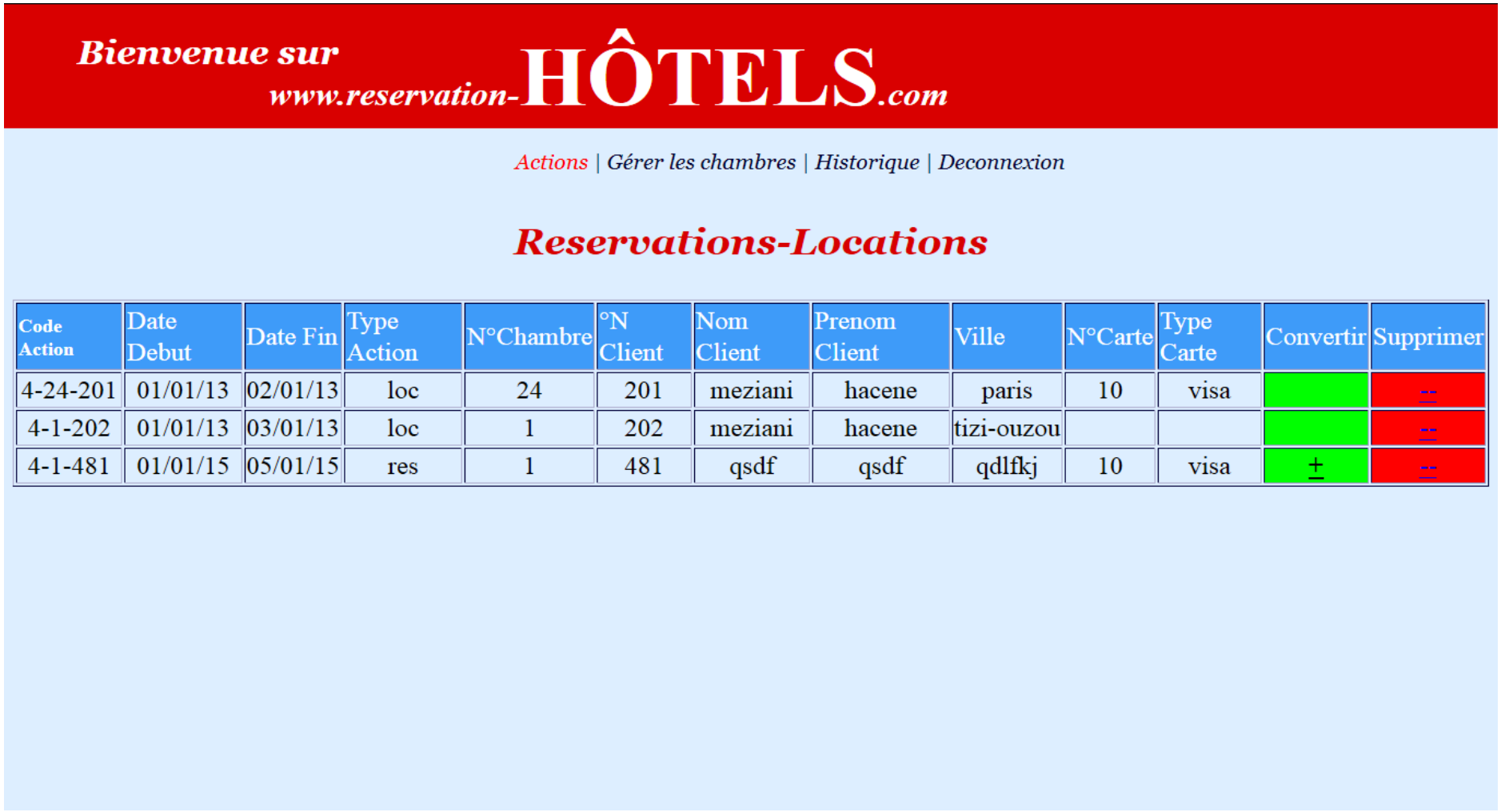


Figure 5.17 : Se connecter à l'Espace Agent Hôtel

En cliquant sur le lien « connectez- vous ». L'agent d'hôtel a accès à son espace . Il saisit son email et son mot de passe pour s'authentifier.

Figure 5.18 : espace agent hôtel

Cet espace est composé d'un en-tête, une barre de menu et une photo de l'hôtel



Code Action	Date Debut	Date Fin	Type Action	N°Chambre	°N Client	Nom Client	Prenom Client	Ville	N°Carte	Type Carte	Convertir	Supprimer
4-24-201	01/01/13	02/01/13	loc	24	201	meziani	hacene	paris	10	visa		--
4-1-202	01/01/13	03/01/13	loc	1	202	meziani	hacene	tizi-ouzou				--
4-1-481	01/01/15	05/01/15	res	1	481	qsdf	qsdf	qdlfkj	10	visa	+	--

Figure 5.19 :pages Actions(Réservation / Location)

Dans cette page l'agent d'hôtel peut gérer les actions. Il peut convertir une réservation à une location ou bien supprimer une action.

5.10 Conclusion

Dans ce dernier chapitre nous avons présenté les détails techniques qui nous ont permis d'implémenter notre système d'interrogation flexible de réservation en ligne. Le travail n'a pas été simple. Nous avons touché à plusieurs domaines : le flou, l'analyse lexicale , syntaxique et sémantique, le géo localisation , le paiement en ligne.

Conclusion et Perspectives

Notre mémoire a trait aux requêtes flexibles exprimant des préférences via des prédicats flous. De telles requêtes permettent de satisfaire au mieux les besoins des utilisateurs. Notre objectif était sur un premier plan de mener une étude théorique sur ce domaine de recherche assez récent. Sur un deuxième plan, nous avons voulu montrer la faisabilité de l'interrogation flexible sur un cas réel. Notre choix s'est porté sur la réservation d'hôtel en ligne.

La théorie des ensembles flous constitue le fondement scientifique de notre travail . Elle est utilisée dans divers domaines de l'informatique, par exemple dans le domaine des systèmes d'information et de connaissance. Dans le cas de notre étude sur la flexibilité et la gradualité dans les bases de données, la théorie des ensembles flous semble être un moyen très intuitif et très proche de la raison de l'être humain lui permettant d'exprimer ces préférences.

Notre travail est très satisfaisant dans la mesure où les résultats répondent aux besoins et retournés selon un ordre de pertinence par rapport aux préférences exprimés dans la requête.

En perspective, on pourrait faire une estimation de coût des requêtes, explorer des requêtes plus complexes telle que la division par exemple. Sur le plan pratique, on envisage d'intégrer notre application à des appareils de téléphonies mobiles équipés d'un GPS et de l'étendre pour couvrir tout le territoire national (pourquoi le monde entier). Encore, on souhaiterait lancer une petite publicité pour attirer le public à notre application.

Bibliographie

[BRO 04] F. Brouard , « La division relationnelle, mythe ou réalité ? », Article publié sur le site : http://sqlpro.developpez.com/cours/divrelationnelle/ , le 18/04/2004.

[COD 70] E.F Codd, «A Relational Model of Data for Large Shared Data Banks», communication of ACM, vol.13, 1970. http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

[GAB 01] P. GABRIEL, «Introduction générale à la logique floue et à la théorie des sous-ensembles floues», 2001. http://ace.montefiore.ulg.ac.be/elap/documents/fuzzy/Transparents_doc/Transparents_fuzzy_logic_1.pdf

[GOM 98] F. Gomide et W. Pedrycz , « An Introduction to Fuzzy Sets : Analysis and Design », MIT Press, 1998 . http://books.google.fr/books?id=wzWZVGVdlLIC

[HAS 05] M.A Ben Hassanine, « Contribution à l'implémentation d'une base de données floue sous un système de gestion de base de données relationnel», Mémoire de magister, soutenue devant l'Ecole nationale d'ingénieurs de Tunis, 2005.

[KAU 77] A. Kaufmann, « Introduction à la théorie des sous-ensembles flous – Tome I : Eléments théoriques de base », Edition Masson, 2ème édition, 1977.

[BOS 95] P. Bosc et O. Pivert, « SQLf : A relational database language for fuzzy querying », IEEE Transactions on Fuzzy Systems, 3, pp.1-17, 1995.

[BOS 04] P. Bosc , L. Liétard et O. Pivert et D. Rocacher , « Base de données , Gradualité et Imprécision dans les bases de données : Ensembles flous, requêtes flexibles et interrogation de données mal connues», Editions ellipses, 2004.

[HAS 05] M.A Ben Hassanine, « Contribution à l'implémentation d'une base de données floue sous un système de gestion de base de données relationnel», Mémoire de magister, soutenue devant l'Ecole nationale d'ingénieurs de Tunis, 2005.

[GOO12] http://www.dean.usma.edu/math/people/Heidenberg/JOMA_GPS/Class-Materials/Geographic%20Distances%20ICTCM/Geographic%20Distances.pdf

[YES 11] Y. YESLI , « Contribution à l'étude de requêtes flexibles : mise en œuvre d'un opérateur de division tolérant », mémoire de magister , soutenue en juin 2011 à l'ESI.

Annexe A : UML

1. Les diagrammes d'UML:

UML 2.0 comporte treize types de diagrammes représentant autant de vues distinctes pour représenter des concepts particuliers du système d'information. Ils se répartissent en deux grands groupes:

Diagrammes structurels ou diagrammes statiques (UML Structure)

- diagramme de classes (Class diagram)
- diagramme d'objets (Object diagram)
- diagramme de composants (Component diagram)
- diagramme de déploiement (Deployment diagram)
- diagramme de paquetages (Package diagram)
- diagramme de structures composites (Composite structure diagram)

Diagrammes comportementaux ou diagrammes dynamiques (UML Behavior)

- diagramme de cas d'utilisation (Use case diagram)
- diagramme d'activités (Activity diagram)
- diagramme d'états-transitions (State machine diagram)
- diagrammes d'interaction(Interactiondiagram)
- diagramme de séquence (Sequence diagram)
- diagramme de communication (Communication diagram)
- diagramme global d'interaction (Interaction overview diagram)
- diagramme de temps (Timing diagram)

Ces diagrammes, d'une utilité variable selon les cas, ne sont pas nécessairement tous produits à l'occasion d'une modélisation. Les plus utiles pour la maîtrise d'ouvrage sont les diagrammes d'activités, de cas d'utilisation, de classes, d'objets, de séquence et d'états-transitions. Les diagrammes de composants, de déploiement et de communication sont surtout utiles pour la maîtrise d'œuvre à qui ils permettent de formaliser les contraintes de la réalisation et la solution technique.

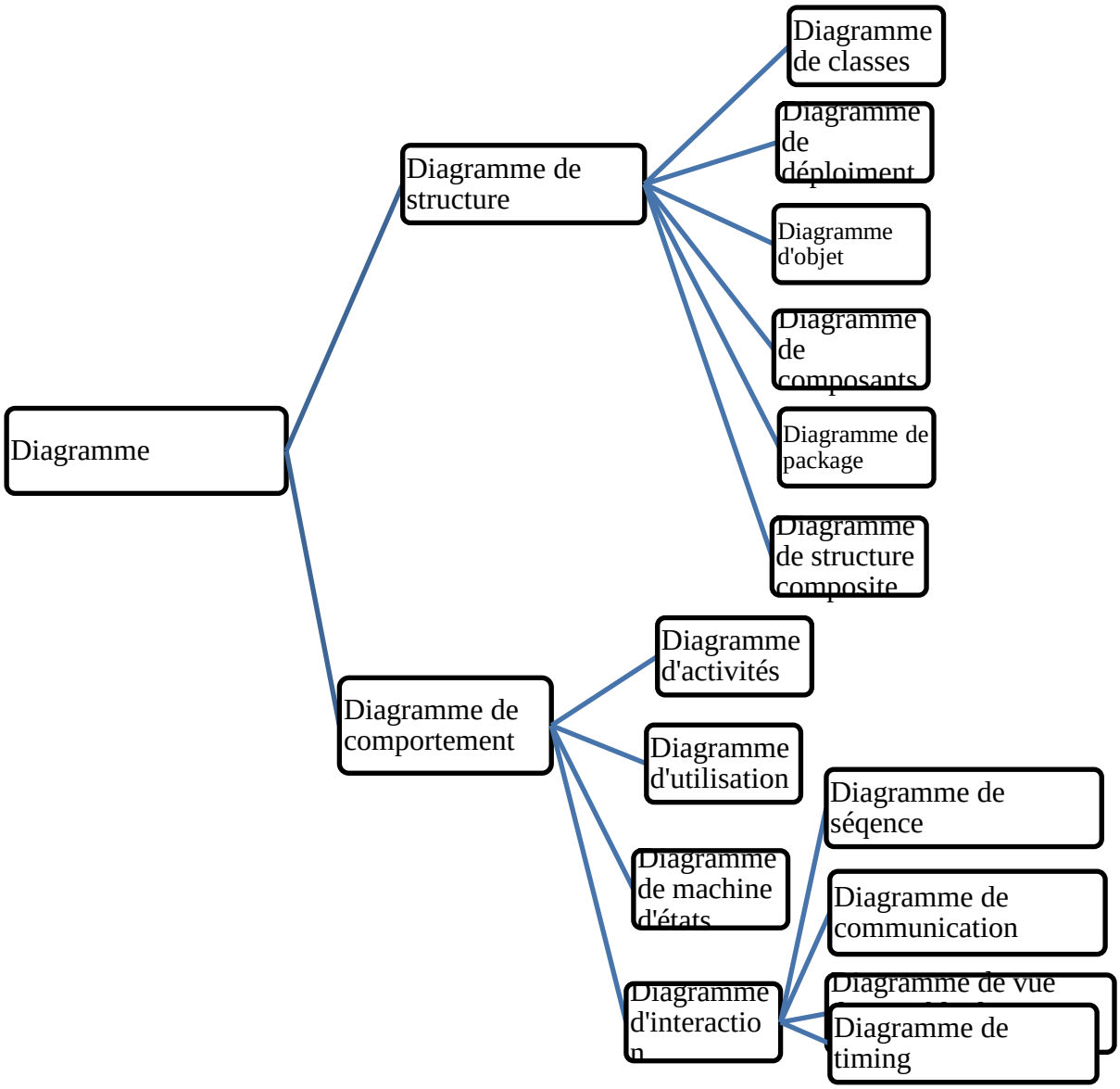


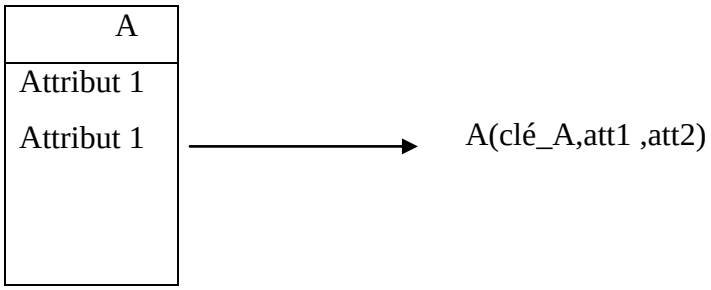
Figure : Classification des diagrammes UML (V 2.0)

2. Règles de passage du diagramme de classe vers le modèle relationnel :

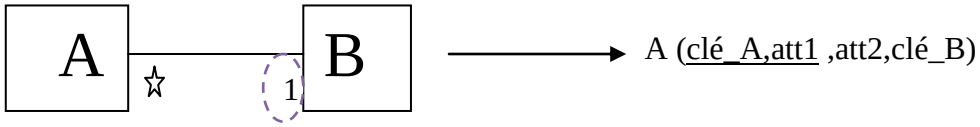
Un Schéma relationnel est un ensemble de relation en troisième forme normale .ce schéma peut être obtenu à partir d'un schéma conceptuel tel Qu'un diagramme de classe**.**

Pour cela il faut :

- Traduire chaque classe par une relation à laquelle on ajoute une clé,

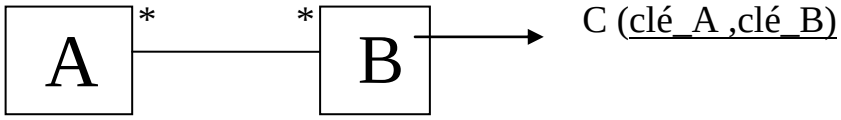


- Traduire les associations :
- Cela qui ont un bout avec une multiplicité maximum inférieure ou egale à 1 :



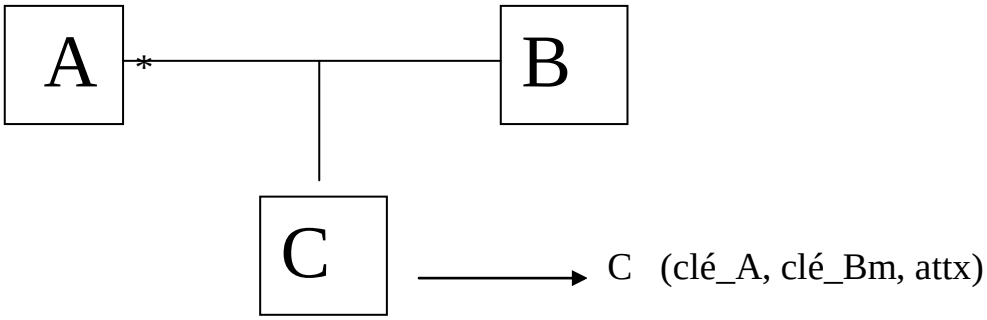
Peuvent être traduire en migrant la clé de B migre vers A en clé étrangère

Celles qui ont les bouts avec une multiplicité maximum * :



Nous ajoutons une relation C ayant une clé composée des clés de A et B.

Classes associations :



Nous ajoutons une relation C ayant une clé composée des clés de A de A et B ,

Et pour attributs ceux de la classe associative.

La traduction de la relation d'héritage se fait en propageant la clé de la classe mère vers les calasses filles .

Référence :

Annexe B

Formule du calcule de la distance sur la carte

Soient deux points : **A (latA, lonA) , B (latB, lonB)** quelconques sur la Terre ;

Tels que : latA est latitude de A, latB est latitude de B (en degré à virgule décimale)

lonA est longitude de A ,lonB est longitude de B (en degré à virgule décimale)

La formule générale de la distance entre ces deux points est :

Distance : R arcos(sin(latA)sin(latB)+cos(latA)cons(latB)cos(lonA-lonB))

Tels que R est le rayon moyen de la Terre et les latitudes et les longitudes sont convertis en radian.

Cette formule donne des résultat accéptable pour des longue distances (plus de 100km), mais elle donne des faux résultats pour les courtes distance, ce qui n'est pas pratique pour notre cas d'étude, puis qu'on travaille seulement dans la carte de la ville de Tizi-ouzou dont la longeur ne dépasse pas les 10 klm.

Dans les petite surface (comme la carte de tizi-ouzou) la courbure de la Terre est négligeable, alors on peux consédrer cette carte comme une surface plate, et dans ce cas, la foumule qui nous donne une aproximation acceptable de la distance entre deux points est celle basée sur le théorème de Pythagore comme on l'a montrer dans la figure suivante :

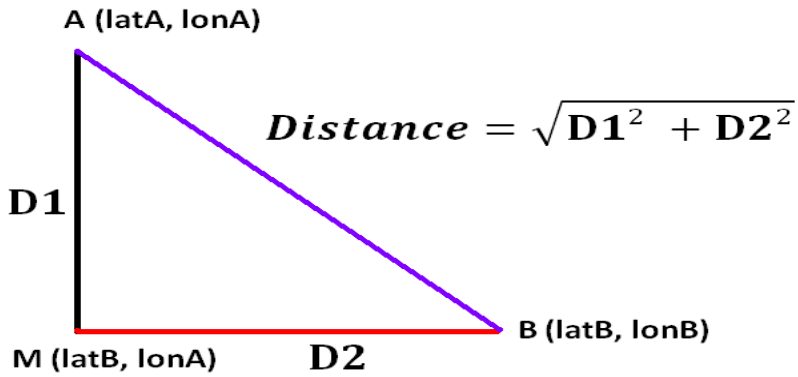


Figure : Théorème de Pythagore

Calculons D1 et D2 :

d'abord cherchant le valeur du rayon R :

notre planète a deux rayons : rayon polaire=6357km et rayon équatorial=6378km.

et R= moyenne(rayon polaire , rayon équatorial) = 6367,5= 6368 (arrondi)

donc **R=6368 km**

Calculons D1 : distance entre deux points ayant la même longitude

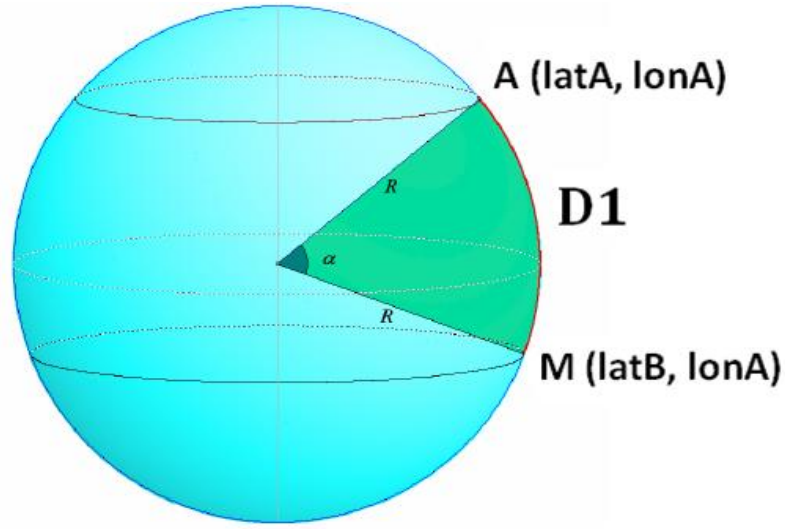


Figure : distance entre deux points ayant la même longitude

Selon la figure la formule de D1 est :

D1=R* α

α= |latA-latB| π/180 , tels que latitude et longitude sont en degré

donc **D1=R*|latA-latB| π/180.**

Calculons D2 : distance entre deux points ayant la même longitude :

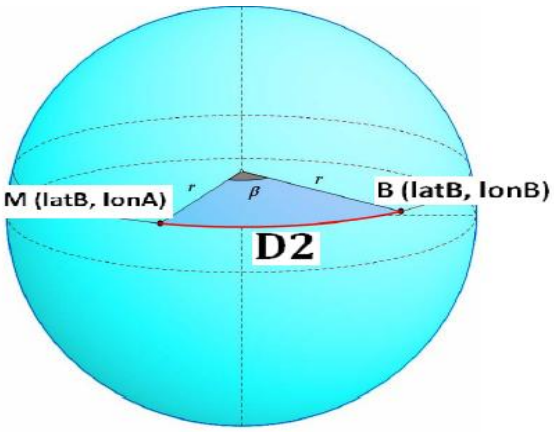


Figure : distance entre deux points ayants la même longitude

Dans ce cas **: β=|lonB-lonA|**

Mais r = R cos(latB $*$ π/180) comme c'est illustré dans la figure suivante :

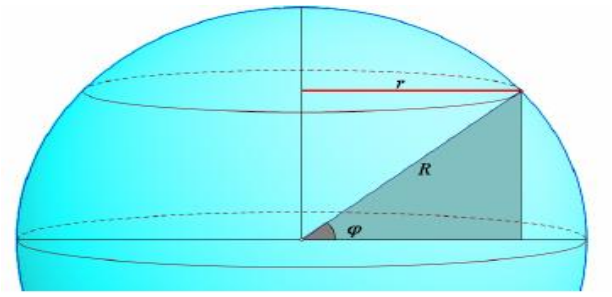


Figure : illustration de r=Rcos(ϕ$*$ π/180)

Tel que **ϕ=latB,**

Donc D2= Rcos(latB $*$ π/180)* β* π/180

D2= Rcos(latB $*$ π/180)*|lonB − lonA| * π/180

Maintenant calculant la distance entre A et B :

Distance=$\sqrt{D1^2 + D2^2}$

$$\textbf{Distance= R}\frac{\pi}{180}\sqrt{(\textbf{latA} - \textbf{latB})^2 + \left(\cos\left(\frac{\textbf{latB}\pi}{180}\right) * (\textbf{lonB} - \textbf{lonA})\right)^2}$$

Mais il y a un petit manque dans cette formule, car on a deux latitudes différentes, donc pour calculer le petit r on doit calculer la moyenne entre ces deux latitudes pour avoir une bonne approximation de la distance.

La formule va devenir :

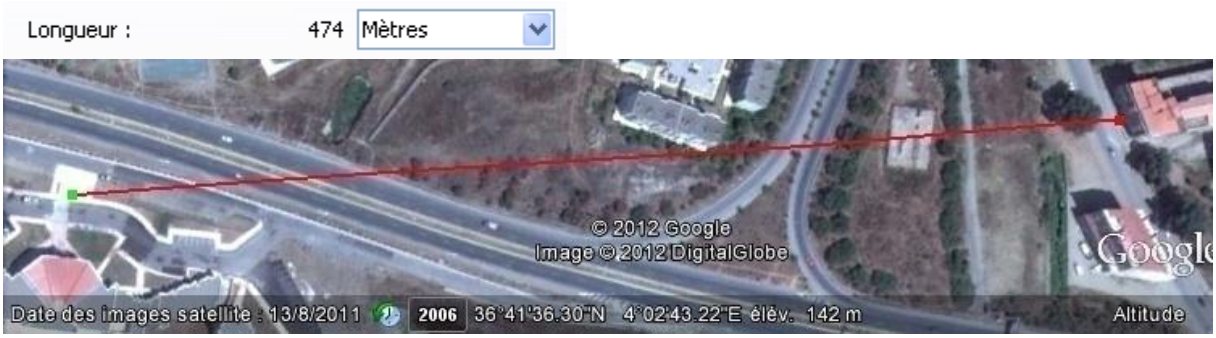
$$\textbf{Distance= R}\frac{\pi}{180}\sqrt{(\textbf{latA} - \textbf{latB})^2 + \left(\cos\left(\frac{(\textbf{latA}+\textbf{latB})\pi}{360}\right) * (\textbf{lonB} - \textbf{lonA})\right)^2}$$

Formule simplifiée :

$$\textbf{Distance= R}\sqrt{(\textbf{latA} - \textbf{latB})^2 + \left(\cos\left(\frac{\textbf{latA}+\textbf{latB}}{2}\right) * (\textbf{lonB} - \textbf{lonA})\right)^2}$$

Et ça après avoir converti les latitudes et les longitudes du degré au radian

Exemple : calculant la distance entre la nouvelle gare-routière et l'Hôtel Itturra



Gare-routière=A (36° ,41’36,30’’N, 4°02’43,22’’E)

Hotel Ittourar = B (36°,41’37,35’’N, 4°03’02 .23’’E)

La longueur trouvée par notre formule est: 472 mètres

Et celle données par Google Earth est : 474, une différance de 2 mètres,

Alors, notre formule nous donne une bonne approximation

Référence :

[GOO12] http://www.dean.usma.edu/math/people/Heidenberg/JOMA_GPS/Class-Materials/Geographic%20Distances%20ICTCM/Geographic%20Distances.pdf