République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique
Université **M**'hamed **B**ougara - **B**oumerdès

Faculté des Sciences
Département d'Informatique

MEMOIRE

Pour l'obtention du diplôme de

MAGISTER EN INFORMATIQUE

***Option* : « *INFORMATIQUE FONDAMENTALE* »**

***Thème* :**

Un système à base d'Acteurs pour l'interrogation des BDD en langue naturelle.

Présenté et soutenu publiquement le 21.09.2006 par

BADAOUI Atika

Devant le jury composé de:

Mr MEZGHICHE M.	**Pr.**	**U.M.B.B**	**Président**
Mr ZEGGOUR D.E.	**Pr.**	**I.N.I**	**Promoteur**
Mr HIDOUCI W.K.	**Cc.**	**I.N.I**	**Co-Promoteur**
Mr HERZALLAH A.	**Cc.**	**U.M.B.B**	**Examinateur**
Mr AIT-AOUDIA S.	**Mc.**	**I.N.I**	**Examinateur**
Mr AHMED-NACER M.	**Pr.**	**U.S.T.H.B**	**Examinateur**

"نظام قائم على أساس نموذج "العامل"

لاستجواب قواعد المعطيات باللغة البشرية العادية"

ملـخـــص

منذ بداية عصر الإعلام الآلي، كان أحد أهم انشغالات المستخدم إمكانية تخزين كميات هائلة من المعطيات و استعمالها بطريقة منتظمة من أجل استعراضها، تغييرها أو استخلاص معلومات أخرى. مما جعل من قواعد المعطيات أداة لا غنى عنها في جميع نظم تسيير المعلومات,

في حين أن مستخدم قواعد المعطيات غالبا ما يكون غير متخصص في الإعلام الآلي، مما يجعل الحوار ما بين المستخدم والنظام صعبا. هذا مما يسبب في الحد من عدد المستخدمين و إبعادهم عن أدوات الإعلام الآلي.

الميزة الأساسية في استعمال اللغة البشرية العادية لمعالجة قواعد المعطيات تكمن في كون أن المستخدم لن يكون ملزما بمعرفة بنية القاعدة. بالإضافة إلى أنه من الممكن تأويل عبارات مضمرة (محذوف منها) بخلاف اللغات الرياضية.

إن إنشاء نظام لاستجواب قواعد المعطيات باللغة البشرية مع اعتبار المدلول، الاستجواب و إنتاج الأجوبة عملية معقدة نتيجة لتنوع مصادر المعارف المستخدمة من طرف النظام و الحاجة إلى تبادل المعلومات بين مختلف عناصره. هذه المقتضيات تستلزم اختيار منهج يسمح بتوزيع المعارف من جهة، و ضمان طرق ناجعة للتواصل بين مختلف تلك المصادر من جهة أخرى .

في مثل هذا المنهج، تعتبر مستويات المعالجة أنظمة فعالة متصلة فيما بينها و كل مستوى يؤخذ كتركيبة لمجموعة مواد حيوية.

يمثل نموذج "العامل" حلا مثاليا لتلبية مثل هذه الاعتبارات.

حسب تعريفه، "العامل" خبير يحيا في مجتمع و هو في اتصال مع باقي الخبراء لحل المشاكل المطروحة.

بحثنا هذا يندرج ضمن مشروع ACT 21 تحت طور الإنجاز على مستوى المعهد الوطني للإعلام الآلي بالجزائر و يتمثل في اقتراح هندسة لنظام يسمح باستخدام اللغة البشرية لتشغيل قاعدة البيانات. يتلقى النظام من المستخدم سؤالا باللغة العادية يترجمها إلى عبارة بلغة SQL و يحملها إلى الوحدة REQ المكلفة بتنفيذ طلبات محررة بلغة SQL.

Résumé

Dés les débuts de l'informatique, l'un des soucis majeurs de l'utilisateur fut de pouvoir stocker massivement des données et de pouvoir en disposer régulièrement afin d'en extraire de nouvelles informations, de les consulter et de les modifier. C'est ce qui a fait des bases de données un outil indispensable dans tous les systèmes de gestion de l'information.

Cependant, l'utilisateur de bases de données est bien souvent un non informaticien. Un dialogue homme-machine trop rigide syntaxiquement, rend parfois difficile l'emploi d'un tel système. Ceci a généralement pour effet de limiter le nombre d'utilisateurs potentiels, et de les éloigner d'outils informatiques qu'ils rejettent, car trop compliqués, ou trop difficiles à utiliser. L'avantage majeur d'une interface en langue naturelle réside dans le fait que l'utilisateur n'est plus obligé de connaître la structure de la base de données. En plus, une interface en langue naturelle peut interpréter des expressions anaphoriques et elliptique ce qui est impossible avec les langages formels.

La mise en place d'un système d'interrogation de BDD en langue naturelle en considérant la sémantique, l'interrogation et la génération de réponse est très compliquée. Cette complexité est due à la diversité des sources de connaissances que manipule le système et la nécessité d'une communication continue et d'un échange d'informations entre ses différents composants.

De telles exigences impliquent le choix d'une approche permettant d'une part la distribution des connaissances et d'autre part d'assurer une communication efficace entre les sources de connaissance. Dans une telle approche les niveaux de traitement seront vus comme des processus actifs communiquant entre eux, et chaque niveau sera considéré comme une composition d'un ensemble d'objets dynamiques. Le modèle acteur, se présente comme une solution idéale pouvant satisfaire de telles exigences. Par sa définition, un acteur est considéré comme un expert vivant en société et communiquant avec d'autres experts pour résoudre des problèmes.

Notre travail s'inscrit dans le projet SGBD Act21 en cours de développement à l'Institut National d'Informatique, il consiste à proposer une architecture d'un système autorisant la manipulation d'une BDD en langue naturelle. Il reçoit en entrée une question en langue naturelle formulée par l'utilisateur du système, la traduit en une requête SQL et la transmet au module REQ chargé de l'exécution d'une requête de type SQL.

Abstract

Since the beginning of computer science, one of the major concerns of users was to be able to stock data massively and to be able to get them regularly in order to extract new information, to consult them and to modify them. This made of databases an essential tool in all information management systems.

The major advantages of a natural language interface lies in the fact that the user is not obliged to know the structure of the database. In addition, a natural language interface can interpret an anaphoric and elliptic expression which is impossible with formal languages.

The implementation of a database interrogation system in natural language taking into consideration semantics, interrogation and response generation is very complicated. This complexity is due to the diversity of knowledge resources manipulated by the system and the need of a continuous communication and exchanging information between its components.

Such requirements imply choosing an approach allowing knowledge distribution on one hand and to ensure efficient communication between knowledge sources, on the other hand. In such an approach, processing levels will be seen as active processes communicating between themselves, and each level will be considered as a composition of a set of dynamic objects.

The actor model is given as an ideal solution that could satisfy such requirements. By its definition, an actor is considered as and expert living within society and communicating with other experts to solve problems. Each one of these experts can be itself a society of more elementary experts. The model is thus distributed (or repartitioned). Knowledge and behaviours are diffused among actors, performing tasks in parallel, in an independent manner.

Our work is linked to the Act21 DMS project being developed at Institut National d'Informatique. The work consists of proposing an architecture for a system allowing manipulating a database in natural language form. It takes as an input a question written by the user of the system in natural language, translates it into an SQL query and send it to the REQ module responsible for executing an SQL-like query.

Table des matières

Liste des figures

Liste des abréviations

Abréviation	Détail
ABCL	**A**n object **B**ased **C**oncurrent **L**anguage
ASK	**A S**imple **K**nowledgeable system
B.D.D	**B**ase **D**e **D**onnées
CAT	**CAT**égorie
CYK	**C**ocke **Y**ounger **K**asami
DATALOG	**Data**base dialog
EUFID	**E**nd-**U**ser **F**riendly **I**nterface to **D**ata management
HPSG	**H**ead-driven **P**hrase **S**tructure **G**rammar
IA	**I**ntelligence **A**rtificielle
IHM	**I**nterface **H**omme **M**achine
ILNBDD	**I**nterface en **L**angue **N**aturelle pour les **B**ases de **D**onnées.
L.N	**L**angue **N**aturelle
MARGIE	**M**emory, **A**nalysis, **R**esponse **G**eneration **I**n **E**nglish
MASQUE	**M**odular **A**nswering for **QU**eries in **E**nglish
PEP	**P**arallel **E**xpert **P**arser
SAM	**S**cript **A**pplier **M**echanism
SGBD	**S**ystème de **G**estion de **B**ases **de D**onnées
SN	**S**yntagme **N**ominal
SQL	**S**tructured **Q**uery **L**anguage
ST	**S**tructure de **T**rait
STEP	**S**chema **T**uple **E**xpression **P**rocessor
TAL(N)	**T**raitement **A**utomatique de la **L**angue **(N**aturelle**)**
TEML	**T**emporal **E**vent **M**atching **L**anguage
UML	**U**nified **M**odeling **L**anguage
WWW	**W**orld **W**ide **W**eb

Chapitre 1 Introduction générale

1.1 Position du problème

Dés les débuts de l'informatique, l'un des soucis majeurs de l'utilisateur fut de pouvoir stocker massivement des données et de pouvoir en disposer régulièrement afin d'en extraire de nouvelles informations, de les consulter et de les modifier. C'est ce qui a fait des bases de données un outil indispensable dans tous les systèmes de gestion de l'information [Bouzeghoub et al., 2000].

Cependant, l'utilisateur de bases de données est bien souvent un non informaticien. Un dialogue homme-machine trop rigide syntaxiquement, rend parfois difficile l'emploi d'un tel système. Ceci a généralement pour effet de limiter le nombre d'utilisateurs potentiels, et de les éloigner d'outils informatiques qu'ils rejettent, car trop compliqués, ou trop difficiles à utiliser. En plus, l'hétérogénéité des populations d'utilisateurs de bases de données rend difficile l'emploi de langages de requêtes, du fait d'une part de l'apprentissage d'un tel langage (de type SQL ou autre), et d'autre part de la nécessité de connaître le schéma de la base (modèle de représentation des données) pour pouvoir établir les divers liens entre les données (tables de relations ou objets), afin de déterminer le chemin d'accès dans la base [Trigano, 2005].

L'avantage majeur d'une interface en langue naturelle réside dans le fait que l'utilisateur n'est plus obligé de connaître la structure de la base de données [Trigano, 2005]. En plus, une interface en langue naturelle peut interpréter des expressions anaphoriques et elliptiques ce qui est impossible avec les langages formels [Androutsopoulos et al., 1995].

Le but global d'une interface en langue naturelle est de traduire une question vers une requête codée dans un langage informatique d'interrogation de base de données, comme SQL par exemple. La traduction doit associer aux mots de la question d'une part les éléments de la base considérée, d'autre part les opérations à réaliser sur ces éléments [Bouchou et Maurel, 1999]. Cette traduction est précédée par un traitement linguistique de la question. Par traitement linguistique nous entendons les modules qu'il faut développer pour parvenir à la compréhension globale de la question. Nous distinguons :

- le niveau morphologique qui consiste à la détection des mots contenus dans le texte.
- le niveau syntaxique qui associe aux mots des formes grammaticales.
- le niveau sémantique qui s'intéresse au sens global de la phrase.

En outre, la flexibilité du langage naturel permet à l'utilisateur de formuler sa question de plusieurs manières.

En plus du traitement de la question, un système d'interrogation doit pouvoir interagir avec l'utilisateur. Cette interaction se traduit par la génération de réponse ou même l'échange d'un dialogue dans le cas d'une ambiguïté.

La mise en place d'un système d'interrogation de BDD en langue naturelle en considérant la sémantique, l'interrogation et la génération de réponse est très compliquée. Cette complexité est due à la diversité des sources de connaissances que manipule le système et la nécessité d'une communication continue et d'un échange d'informations entre ses différents composants.

De telles exigences impliquent le choix d'une approche permettant d'une part la distribution des connaissances et d'autre part d'assurer une communication efficace entre les sources de connaissance. Dans une telle approche les niveaux de traitement seront vus comme des processus actifs communiquant entre eux, et chaque niveau sera considéré comme une composition d'un ensemble d'objets dynamiques. Le modèle acteur, se présente comme une solution idéale pouvant satisfaire de telles exigences. Par sa définition, un acteur est considéré comme un expert vivant en société et communiquant avec d'autres experts pour résoudre des problèmes. Chacun des experts peut lui-même être une société d'experts plus élémentaires. Le modèle est donc distribué (ou réparti). Les connaissances et les comportements sont diffusés parmi les acteurs, qui effectuent des tâches en parallèle, de manière indépendante.

1.2 Le contexte de la recherche

Les avancés dans le domaine des machines parallèles et des réseaux à très haut débit laissent croire que les environnements répartis et parallèles formeront les nouveaux standards des architectures à venir. Le développement de logiciels doit donc s'adapter à ce type d'architectures pour bénéficier de leurs avantages. Beaucoup de travaux et de recherches sont consacrés, ces dernières années, au domaine de bases de données parallèles et plus particulièrement aux SGBD objets parallèles. Les modèles objet et acteur sont assez proches l'un de l'autre ce qui permet de les combiner et bénéficier de leurs avantages réunis (puissance de modélisation + parallélisme implicite). L'objectif est d'étudier cette intégration entre *objets* et *acteurs* dans le domaine des bases de données. L'idée est de concevoir un concept offrant la même puissance de modélisation que le modèle objet tout en étant fortement orienté vers les systèmes ouverts et parallèles. La solution proposée est la conception des *"Acteurs de Base*

de Données (DB-ACT)". Ceux sont des objets autonomes et actifs représentant une base de données (schémas, extensions, contraintes,….). Ils collaborent à la résolution des requêtes et des programmes d'application en parallèle. Act21 est un prototype de SGBD parallèle à base de DB-Act. Il est en cours de développement à l'Institut National d'Informatique d'Alger (I.N.I). Il est formé de quatre modules : STOCK, CAT, EXEC, REQ. Ce dernier offre aux utilisateurs un langage de requêtes déclaratif (type SQL) et prend en charge son exécution en générant les différents acteurs nécessaires à la résolution de la requête. Son rôle est de traduire les requêtes en un programme écrit en PACT et appelle l'interpréteur du module EXEC pour contrôler leur exécution [Hidouci et Zegour, 2001].

Notre travail consiste à proposer une architecture d'un système autorisant la manipulation d'une BDD en langue naturelle. Il reçoit en entrée une question en langue naturelle formulée par l'utilisateur du système, la traduit en une requête SQL et la transmet au module REQ.

1.3 Structure de la thèse

Nous commençons dans le deuxième chapitre par présenter les caractéristiques d'un système d'interrogation de BDD en langue naturelle. Nous mettrons l'accent sur les problèmes liés à un tel système tels que : la portée des quantificateurs, le traitement des conjonctions et disjonctions, le traitement des expressions anaphoriques et elliptiques. Nous présenterons à travers des exemples les différentes méthodologies permettant d'évaluer une question par rapport à la base de données. Ce chapitre expose également, d'autres phénomènes de grande importance telles que la portabilité du système d'interrogation, la génération de réponse et le traitement des requêtes de mise à jour.

Le troisième chapitre introduira le concept d'acteur, ses domaines d'application et parcours quelques modèles d'acteur très connus.

L'objectif du quatrième chapitre est de voir les apports du modèle acteur pour le traitement de la langue, en présentant d'une part, les apports du parallélisme et d'autre part, l'intérêt d'un traitement distribué. En fin de ce chapitre seront discutés quelques systèmes de traitement de la langue exploitant le paradigme d'acteur.

Dans le cinquième chapitre seront présentés quelques systèmes de TAL déjà développés. Le but est de montrer les techniques utilisées par les systèmes actuels. Il n'est pas possible de couvrir tous les systèmes, cependant nous allons considéré des systèmes qui nous paraissent intéressants pour notre recherche. Nous nous intéresserons aux systèmes d'interface en langue naturelle ainsi que ceux basés sur le paradigme d'acteur.

Le sixième chapitre est consacré à la conception dans lequel nous allons proposer une solution à base d'acteurs pour l'interrogation d'une BDD en langue naturelle en exploitant tous les concepts théoriques étudiés dans les chapitres précédents.

Chapitre 2 L’interrogation des BDD en langue naturelle

2.1 Introduction

Dés les débuts de l’informatique, l’un des soucis majeurs de l’utilisateur fut de pouvoir stocker massivement des données et de pouvoir en disposer régulièrement afin d’en extraire de nouvelles informations, de les consulter et de les modifier. C’est ce qui a fait des bases de données un outil indispensable dans tous les systèmes de gestion de l’information.

L'utilisateur de bases de données est bien souvent un non informaticien. Un dialogue homme-machine trop rigide syntaxiquement, rend parfois difficile l'emploi d'un tel logiciel. Ceci a généralement pour effet de limiter le nombre d'utilisateurs potentiels, et de les éloigner d'outils informatiques qu'ils rejettent, car trop compliqués, ou trop difficiles à utiliser. En plus, l'hétérogénéité des populations d'utilisateurs de bases de données rend difficile l'emploi de langages de requêtes, du fait d'une part de l'apprentissage d'un tel langage (de type SQL ou autre), et d'autre part de la nécessité de connaître le schéma de la base (modèle de représentation des données) pour pouvoir établir les divers liens entre les données (tables de relations ou objets), afin de déterminer le chemin d'accès dans la base.

L'avantage majeur d'une interface en langue naturelle réside dans le fait que l'utilisateur n'est plus obligé de connaître la structure de la base de données.

2.2 Présentation

Une interface en langue naturelle pour les BDD[Androutsopoulos et al., 1995] est un système qui permet à un utilisateur d'accéder aux informations stockées dans des BDD en utilisant des questions exprimées dans une langue naturelle (ex. Arabe, Anglais, Français,….). L'exemple ci-dessous est un extrait d'un dialogue entre un être humain et le système ASK [Bozenn et Frederic, 1983] (Les parties en caractères gras sont celles entrées par l'utilisateur) :

> List the destination and home port of each ship.

Ship	destination	home port
Ubu	New York	Naples
	Tokyo	-------
Maru	Oslo	Tokyo
Kittyhawk	Naples	Boston
	Boston	-------
Alamo	London	London
	New York	-------
……	……..	……..
……	……..	……..

> What cities are the home ports of ships whose destination is London?

Boston
London
Norfolk
> ……..
……..

Figure 2-1 : Exemple d'un dialogue dans le système ASK

2.3 Quelques systèmes très connus

Les premiers systèmes permettant l'accès aux BDD en langue naturelle datent des années 1960 [Zwigenbaum, et al., 2003], dont voici quelques exemples :

- Le système *Baseball (*de *Green)* qui permettait l'interrogation de résultats de baseball. Le système est capable de répondre aux questions de type

  Who did the Red Sox lose on July 5?

  On how many days in July did eight teams play? [Zwigenbaum, et al., 2003]

- Le système *LUNAR* créé par *Woods.* Est l'un des systèmes les plus connus dans le début des années soixante dix [Androutsopoulos et al., 1995]. Ce système permettait l'analyse de roches lunaires. Il est constitué de deux bases de données une pour l'analyse chimique et l'autre pour les références linguistiques. Le programme utilise un *Réseau de Transition Augmenté (ATN)* pour l'analyse syntaxique. Une démonstration de ce système a été réalisée au

cours de la seconde conférence annuelle des sciences humaines en 1971. Sa performance était très remarquable [Knowles et Mitrovic, 1999]**.** Le système peut répondre aux questions de type: *What is the average concentration of aluminum in high alkali rocks?*

How many Brescias contain Olivine? [Zwigenbaum, et al., 2003]

- Le système ***LIFER/LADDER*** fournit des statistiques sur les employés. Développé par *Hendrix* (1978). Ce système utilise une grammaire sémantique pour l'analyse de la question. LIFFER peut prendre en charge des requêtes portant sur une seule table et celles sur plusieurs tables incluant une simple jointure. Hendrix a démontré la capacité de LIFFER à répondre à des questions de type: [Knowles et Mitrovic, 1999]

What is average salary of match department secretaries?

How many professors are there in the CompSci department? [Zwigenbaum, et al., 2003]

- CHAT-80 [Warren et Fernando, 1982] est l'un des systèmes les plus connus dans les années quatre-vingt. Il est totalement implémenté en Prolog. CHAT-80 transforme une question (exprimée en Anglais) en expressions Prolog, qui seront évaluées par rapport à une BDD en Prolog. Le code de CHAT-80 a été largement utilisé et a constitué une base pour d'autres systèmes d'interrogation des BDD en langue naturelle (ex. MASQUE « **M**odular **A**nswering **S**ystem for **QU**eries in **E**nglish » [Androutsopoulos et al., 1993]). [Androutsopoulos et al., 1995].
- Les milieux des années quatre-vingt ont vu de nombreuses recherches dans ce domaine et plusieurs systèmes ont été implémentés. Une grande partie de ces recherches a été consacrée à la question de *portabilité*. Par exemple le système TEAM [Androutsopoulos et al., 1995] , [Grosz, 1983] a été conçu pour être facilement configurable par des administrateurs de base de données n'ayant pas de connaissances sur le système [Androutsopoulos et al., 1995].

Parmi les systèmes qui sont aussi apparus dans cette période on site : DATALOG, EUFID, LDC, TELY [Androutsopoulos et al., 1995].

2.4. Les avantages apportés par l'utilisation de la langue naturelle pour les BDD

Les langues naturelles en tant que langage d'accès aux données ont plusieurs avantages parmi les plus importants on citera [Mazlack et Feinauer] **:**

1. Un grand nombre d'utilisateurs de la BDD peuvent être peu disposés ou incapables d'apprendre et d'utiliser des langages machines formels pour interroger leurs BDD.
2. Les applications du langage naturel fournissent un milieu (environnement) de communication idéal.
3. Les utilisateurs potentiels connaissent déjà leur langage naturel ainsi peu de formation dans le langage d'interrogation serait nécessaire.
4. Les langages naturels sont des outils puissants pour l'expression de certains types des idées et de concepts non-mathématiques.
5. La flexibilité de la recherche documentaire est sensiblement améliorée quand les utilisateurs recherchent eux-mêmes les données.

Si les utilisateurs des données stockées par ordinateur, peuvent accéder aux données en employant le langage naturel, l'utilité de stocker les données en machine serait augmentée. Non seulement l'utilisateur occasionnel gagnerait un accès sans obstacles, mais un utilisateur expert pourrait gagner un accès plus facile car les nouvelles éditions (versions) de SGBD ne devraient pas être appris à chaque changement de système.

6. Une interface pour les BDD en langue naturelle peut supporter les expressions anaphoriques et elliptiques ce qui est impossible avec les langages formels [Androutsopoulos et al., 1995]. (pour un aperçu rapide sur le traitement des anaphores voir [Burros et DeRoeck, 1992]).

2.5. Problèmes liés à l'interrogation des BDD en langue naturelle

Bien qu'il y ait plusieurs systèmes permettant l'interrogation des BDD en langage naturel, beaucoup de problèmes dans ce domaine restent à résoudre [Moore,1982] :

2.5.1 La correspondance entre une question et la structure de la BDD

La correspondance entre une question en langage naturel et la BDD diffère nettement selon la manière dont la BDD est organisée. Par exemple, si *Département* est un attribut du fichier *Employé*, la requête *« Combien d'employés y a-t-il dans le département commercial ?? »*, doit compter le nombre d'enregistrements du fichier *Employé* ayant la valeur *Département* égale à *département commercial*. D'autre part, si l'information requise est stockée dans le champ N*ombreEmployé* du fichier *Département*, le système d'interrogation va seulement retourner la valeur de ce champ à partir de l'enregistrement *département commercial*. Un troisième cas se présente si les départements sont décomposés par exemple en services et le nombre d'employés pour chaque service est enregistré. Dans chaque cas, la requête en langage naturel est la même, mais la requête appropriée de BDD est radicalement différente [Moore,1982].

2.5.2 La portée des quantificateurs

Les quantificateurs (tels que : « tous », « chaque », « quelques »,…) dans les questions posent certains problèmes. Il est parfois difficile de déterminer la portée d'un quantificateur dans une phrase. Par exemple la question *(Has every student taken some course?)* peut avoir deux interprétations différentes :

1. $\forall$ *student* $\exists$ *course taken(student,course)*
2. $\exists$ *course* $\forall$ *student taken(student,course)*

Dans la première interprétation, chaque étudiant peut suivre différents cours, tandis que dans la deuxième tous les étudiants suivent un même cours.

Une méthode heuristique peut être utilisée pour résoudre une telle ambiguïté. Cette méthode s'opère de gauche à droite pour la détermination de la portée du quantificateur. Dans la phrase précédente *"every"* apparaît avant *"some"*. En conséquent, et se basant sur cette méthode, le quantificateur universel introduit par *"every"* précède le quantificateur existentiel introduit par *"some"* [Androutsopoulos et al., 1995].

D'autre approche associe un poids pour chaque quantificateur et celui ayant le plus grand poids sera sélectionné pour être le plus large. Dans l'exemple précédent, si *"every"* a un poids plus grand que celui de *"some"*, alors la première interprétation sera appropriée.

2.5.3 Les Conjonctions et disjonctions

Le mot *« et »* est souvent utilisé pour exprimer une disjonction plutôt qu'une conjonction. Cela invoque une ambiguïté difficile à résoudre. Par exemple dans la question *« quels sont les candidats habitant Boumerdes et Alger ? »* chaque candidat habitant *soit* Boumerdes *ou* Alger doit être compté. La possibilité pour que *et* indique une conjonction est rejetée, puisqu'un candidat ne peut pas habiter dans deux villes différentes [Androutsopoulos et al., 1995]. Dans ce contexte, **EUFID** [Templeton et Burger, 1983] détecte les cas où il est conceptuellement impossible que *et* indique une conjonction et le transforme en *ou* , (EUFID transforme un *et* en *ou* si la porté de la conjonction concerne deux phrases ayant leurs valeurs dans le même champ) [Templeton et Burger, 1983].

2.5.4 Problème des composants nominaux

Dans certains cas, un nom peut être modifié par un nom qui le précède. Le sens du nom composé sera difficile à définir. Par exemple, la phrase *« une grande entreprise »* peut signifier une entreprise avec un grand nombre d'employés ou une entreprise dont le chiffre d'affaire est élevé. Pour résoudre une telle ambiguïté certains systèmes exigent durant la phase de configuration, la définition des différents sens possibles pour chaque composant nom-nom ou nom-adjectif [Androutsopoulos et al., 1995].

Dans le système ASK, par exemple, le mot *long* dans le contexte d'une bibliographie en intelligence artificielle est définie (dans une phase dite *definition)* comme suit :

```
> definition : long: paper whose number of page exceeds 49
Defined.
> definition: long: book whose number of pages exceeds 800
Defined.
> What AI bibliography items are long?
There are 2 answers:
(1) long: paper whose number of page exceeds 49
Physical symbol systems
A general syntactic processor
(2) book whose number of pages exceeds 800
Human problem solving
> What long books were written in 1972?
book whose number of pages exceeds 800
Human problem solving
```

Figure 2-2 : Exemple d'une définition d'un modificateur dans le système ASK [Bozenn, Frederic, 1983]

2.5.5 Les anaphores

Les pronoms (ex. elle, ils), les déterminants possessifs (ex. son, leur), et certains groupes nominaux (ex. ces gens) représentent un phénomène linguistique dit *anaphore* [Androutsopoulos et al., 1995]. L'exemple ci-dessous est un extrait d'un dialogue avec le système ASK :

> *Is there a ship whose destination is unknown?*
>
> Yes
>
> *> What is it?*
>
> What is [the ship whose destination is unknown]?
>
> Saratoga

Figure 2-3 : « Exemple d'une anaphore dans le système ASK » [Bozenn et Frederic, 1983]

Le système a compris que "*it*" fait référence à " *the ship whose destination is unknown* ".

Le traitement des anaphores est un problème très connu en traitement de la langue. Voir [Charniak, et al., 1983] pour une approche probabiliste de résolution des anaphores, et [Hobbs, 1986] pour des méthodes relativement simples pouvant être utilisées pour la résolution des anaphores.

2.5.6 Les expressions elliptiques

Les gens emploient très fréquemment dans leurs discours des phrases incomplètes dont le sens sera complété par le contexte de discours, on parle des *ellipses*. L'exemple ci-dessous est un extrait d'un dialogue avec le système PARLANCE qui illustre ce phénomène [Androutsopoulos et al., 1995]:

> Does the highest paid female manager have any degrees from Harvad ?
Yes, 1.
>How about MIT?
No, none.
> Who is the manager of the largest department?

Name	**Dept.**	**Count**
Patterson	**045**	**40**

> The smallest department?

Name	**Dept.**	**Count**
Saavedra	**011**	**2**

Figure 2-4: « Exemple d'expressions elliptiques dans le système PARLANCE ».

2.5.7 Les formes négatives

Les questions en langue naturelle peuvent contenir des négations explicites (comme : ne pas, jamais, etc.), ou des négations implicites (tel que : seulement, sauf, autre que, etc.). Dans une question tel que *"What companies ship to companies other than Colonial?"*, le système EUFID interprète (auther than) comme l'opérateur ' !='. C'est la seule forme de négation implicite que peut traiter ce système [Templeton et Burger, 1983].

Méthodologies

Dans ce paragraphe nous allons répondre à la question suivante : ***comment sera évaluée une question exprimée en langue naturelle par rapport à la BDD ?.*** Plusieurs méthodes ont été appliquées à cet effet, nous les classons en deux catégories :

- Celles qui privilégient l'interprétation de la question vers un langage d'interrogation de BDD supporté par le SGBD (tel que SQL), et
- Celles qui utilisent une représentation conceptuelle directement projetée sur celle de la BDD.

2.6.1 L'interprétation de la question vers le langage d'interrogation

a) Les systèmes Pattern-matching (appariement du modèle)

Certains des premiers systèmes utilisaient la technique d'appariement du modèle dite *"pattern-matching"* [Androutsopoulos et al., 1995] pour générer des réponses aux questions des utilisateurs. Pour illustrer ce principe, considérons une table contenant les informations des pays :

Table-pays		
Pays	Capitale	Langue
Algérie	Alger	Arabe
Italie	Rome	Italien
……	…….	……..

Une primitive du système basé sur le pattern-matching peut utiliser des règles comme :

```
Pattern : ... 'capitale'... <pays>
Action : éditer CAPITALE des lignes où PAYS= <pays>
Pattern : ... 'capitale'... 'pays'
Action : éditer CAPITALE et PAYS de chaque ligne
```

La première règle indique que si la question contient le mot `'capitale'` suivi par un nom du pays (c-à-d un nom figurant dans la colonne `PAYS`), le système doit sélectionner les lignes qui contiennent le nom du pays, et édite la capitale correspondante. Si par exemple, l'utilisateur introduit la question : *'Quel est la capitale de l'Algérie ?'*, le système utilisera la première règle, et retourne *'Alger'*. La même règle permet au système de répondre à la question *'pouvez-vous me dire quelle est la capitale de l'Algérie ?'* ou *'Editer la capitale de l'Algérie'*, dans tous les cas la même réponse sera générée.

L'avantage principal de cette approche est sa simplicité : aucun module d'analyse ou d'interprétation est nécessaire, ainsi que la facilité d'implémentation du système. Cependant, la légèreté de l'approche pattern-matching l'a conduit à l'échec.

b) Les systèmes basés sur la syntaxe

Dans les systèmes à base de syntaxe [Androutsopoulos et al., 1995] la question est analysée (analyse syntaxique), et l'arbre syntaxique résultant est directement projetée en une expression dans un langage d'interrogation de BDD, un exemple typique de cette approche est LUNAR. Les systèmes à base de syntaxe utilisent une grammaire qui décrit les structures syntaxiques possibles de la question. L'exemple suivant montre une grammaire simple dans un système tel que LUNAR.

```
S   → NP VP
NP  → Det N
Det → "what"| "which"
N   → "rock"| "specimen"| "magnesium"| "radiation"| "light"
VP    → V N
V   → "contains"| "emits"
```

Figure 2-5: Une grammaire syntaxique

En utilisant la grammaire ci-dessus, le système pourra générer la structure syntaxique de la phrase *« which rock contains magnesium »* comme présentée dans l'arbre syntaxique suivante :

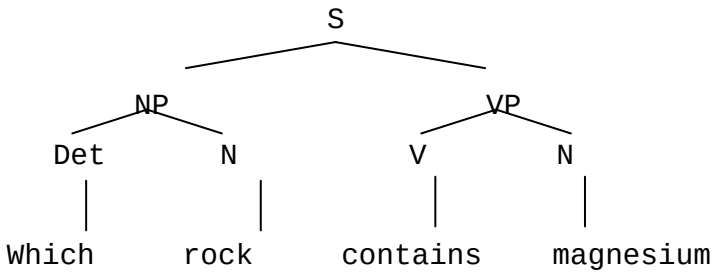

Figure 2-6: Arbre syntaxique de la phrase 'which rock contains magnesium' (exemple du système LUNAR)

A partir de cet arbre, le système élabore la requête ci-dessous (X est une variable)

```
(for_every X (is_rock X)
            (contains X magnesium) ;
              (printout X)
```

Qui sera ensuite, traitée par le système de base de données correspondant. Cette transformation (arbre syntaxique → requête de BDD) est effectuée par des règles et complètement basée sur les informations de l'arbre syntaxique. Dans l'exemple précédent, les règles appliquées sont :

- La projection de ***'which'*** est (***for_every X).***
- La projection de ***'rock'*** est ***(is_rock X).***
- La projection de ***NP*** est ***Det' N'***, avec ***Det'*** et ***N'*** sont respectivement, la projection du déterminant et du nom. Ainsi, la projection de sous-arbre ***NP*** est (***for_every X (is_rock X)).***
- La projection de ***'contains'*** est ***contains.***
- La projection de ***'magnesium'*** est ***magnesium.***
- La projection de ***VP*** est (***V' X N'***, avec ***V'*** et la projection du verbe, et ***N'*** est la projection du nom. Ainsi, la projection de sous-arbre ***VP*** est (***contains X magnesium).***
- La projection de ***S*** est (***NP' VP'*** ; ***(printout X))*** avec ***NP', VP'*** sont les projections des sous-arbres ***NP'*** et ***VP'.***

Il est généralement difficile d'imaginer toutes les règles de projection qui peuvent directement transformer l'arbre syntaxique en expression réelle du langage d'interrogation de BDD (ex. SQL).

c) Les systèmes à grammaire sémantique

Dans les systèmes de grammaire sémantique, le traitement de la question se fait aussi par la génération d'un arbre. La différence est que les catégories de la grammaire ne correspondent pas nécessairement aux concepts syntaxiques mais sémantiques. La figure ci-dessous représente un extrait de la grammaire sémantique utilisée par le système LUNAR:

```
S → Specimen_question| Spaceraft_question
Specimen_question → Specimen emits_info| Specimen contains_info
Specimen → 'which rock'| 'which specimen'
Emits_info → 'emits' Radiation
Radiation → 'radiation'| 'lignt'
Contains_info → 'contains' Substance
Substance → 'magnesium'| 'calcium'
Spacecraft_question → Spececraft Depart_info| Spacecraft Arrive_info
Spacecraft →  'which vessel'| 'which spacecraft'
Depart_info → 'was launched on' Date| 'departed on' Date
Arrive_info → 'returns on' Date| 'arrives on' Date
```

Figure 2-7 : « Un extrait de la grammaire sémantique utilisée par le système LUNAR »

On remarque que les catégories (ex. *Substance, radiation, Specimen_question)* ne correspondent pas aux constituants grammaticaux (ex. nom, phrase) mais représentent des informations sémantiques du domaine de connaissance. Elles sont choisies pour mettre en vigueur des contraintes sémantiques et faciliter la production des objets de la base de données à partir de l'arbre d'analyse.

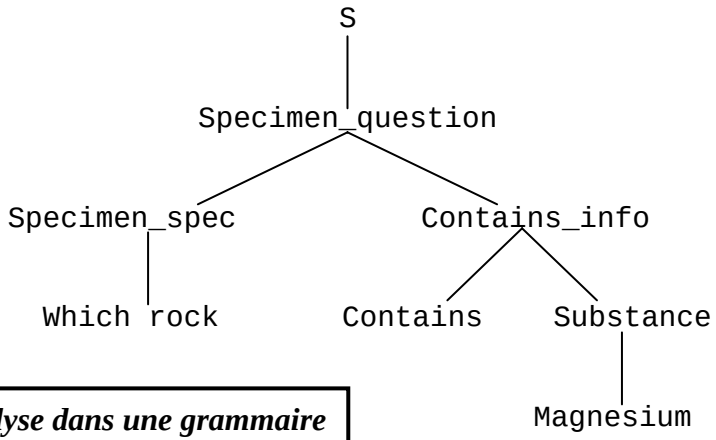

Figure 2-8: « Arbre d'analyse dans une grammaire sémantique »

Cet arbre sémantique représente la phrase *« which rock contains magnesium »*. L'existence du nœud **`Specimen_question`** conduit le système à consulter les tables contenant les informations sur les spécimens. De la même façon, le sous-arbre **`'Contains_info'`** oriente le système vers la recherche d'une table nommée **`'Contains_info'`**, et sélectionner les lignes dont la colonne **`Substance`** est `'magnesium'`. Les grammaires sémantiques sont utilisées dans les systèmes PLANES, LADDER, REL, EUFID.

Les systèmes basés sur cette approche sont très difficiles à appliquer sur plusieurs domaines de connaissance. Cela est dû au fait que la grammaire sémantique contienne des connaissances sémantiques difficilement installées dans un domaine bien spécifique.

d) L'interprétation vers une formule logique

Plusieurs systèmes d'interrogation de BDD en langue naturelle (tel que le système TEAM [Grosz, 1983]**,** MASQUE [Androutsopoulos et al., 1995]**,** [Bozenn et Frederic, 1983]), transforment la question de la langue naturelle en une requête logique intermédiaire. Le processus consiste à:

- Traduire un fragment significatif exprimé en langue naturelle vers une formule logique; (certains systèmes utilisent la logique extensionnelle typée incorporant le lambda calcul voir [Desarte et Thayse, 2001]);
- effectuer la traduction de cette dernière vers un langage d'interrogation de bases de données (tel que SQL);

2.6.2 L'utilisation d'une représentation conceptuelle

La représentation conceptuelle proposée par Sowa [Sowa, 1976] fournit une notation formelle intermédiaire entre l'être humain et l'ordinateur : elle décrit le sens des données (la question) d'après la vue de l'utilisateur, mais qui peut aussi être associée aux procédures d'accès aux données d'après la vue de l'ordinateur. Comme méthode de description formelle, elle a trois avantages :

(i) Elle peut supporter une projection directe sur une BDD relationnelle ;

(ii) Elle peut être utilisée comme une base sémantique pour la langue naturelle ;

(iii) Elle peut supporter des inférences afin d'obtenir des relations qui ne sont pas explicitement mentionnées dans la question.

2.7 La portabilité du système

Ces dernières années, une grande part des recherches sur les interfaces en langue naturelle pour les BDD, a été consacrée à la *portabilité*, c-à-d, la conception des interfaces pouvant être utilisées sur différents domaines de connaissance, par différents SGBD, ou même avec différentes langues. Nous discutons les différents niveaux de portabilité des systèmes:

2.7.1 Portabilité du domaine de connaissance

Une ILNBDD (Interface en langue naturelle pour une BDD) fournit une portabilité du domaine de connaissance, si elle peut être configurée pour une grande variété de domaines. Typiquement, à chaque fois que l'ILNBDD doit être reconfigurée pour un nouveau domaine de connaissance, le système doit être 'entraîné' avec de nouveaux mots et concepts utilisés, ainsi que la manière avec laquelle ces concepts sont reliés avec les informations stockées par la base de données.

Plusieurs ILNBDD imposent que cette configuration du domaine de connaissance soit effectuée par des personnes possédant quelques compétences :

Programmeur : dans certains systèmes une partie du code de l'ILNBDD doit être réécrit durant la phase de reconfiguration du domaine de connaissance. Ceci exige, par conséquent, que la personne chargée de la reconfiguration soit un programmeur, de préférence maîtrisant le code de l'ILNBDD.

Ingénieur de connaissance : d'autres ILNBDD fournissent des outils pouvant être utilisés pour la configuration du système avec un nouveau domaine, sans nécessiter aucune programmation. Ceci exige, cependant, que ces outils soient utilisés par un ingénieur de connaissance, ou une personne maîtrisant les techniques de représentation de connaissances de base, les bases de données, et les concepts linguistiques. Le dialogue suivant, entre *l'éditeur de domaine* du système MASQUE et l'ingénieur de connaissance (pour d'autres exemples plus détaillés voir [Bozenn et Frederic, 1983]), montre comment ajouter le verbe `"to exceed"` au domaine de connaissance du système :

```
editor> add verb
what is your verb ? exceed
what is its third sing. pres ? exceeds
what is its past form ? exceeded
what is its perfect form ? exceeded
what is its participle form ? exceeding
to what set does the subject belong ? numeric
is there a direct object ? yes
to what set does it belong ? numeric
```

Figure 2-9 : «L'ajout du verbe 'to exceed' dans le système MASQUE »

Administrateur de Base de données (expert de base de données) : dans le système TEAM [Grosz, 1983], trois sortes d'informations sont nécessaires pour l'adaptation du système à une nouvelle BDD : ***lexicales*** (les propriétés syntaxiques et sémantiques des mots utilisés dans la question)**,** ***conceptuelles*** (contenant les informations sur les relations, les objets qui servent comme argument aux prédicats), ***le schéma de la base de données.*** Ainsi, la configuration du système pour une nouvelle BDD doit être assignée à une personne ayant des connaissances sur les concepts de BDD. Ci-après deux figures illustrant l'ajout d'un fichier et d'un champ : (les réponses de l'administrateur sont en gras)

```
   File name : CHIP
(1) Fields- (ID MAKER WIDTH SPEED PRICE FAMILY)
(2) Subject- PROCESSOR
(3) Synonyms for PROCESSOR – CHIP
(4) Primary key- ID
(5) Identifying fields – MAKER ID
(6) Can one say Who are the processor?- YES NO
(7) Pronouns for file subject- HE SHE IT THEY
(8) Field containing the name of each file subject- ID
```

Figure 2-10: « Questions sur les attributs d'un fichier dans le système TEAM »

```
  Field – MAKER
 (1) Type of field – SYMBOLIC ARITHMITIC FEATURE
 (2) Are field values units of measure? YES NO
 (3) Noun subcategory- PROPER COUNT MASS
 (4) Domain of field value's reference- SUBJECT FIELD
 (5) Can you say Who is the CHIP-MAKER? YES NO
 (6) Typical value- MOTOROLA
 (7) Will values of this field be used as classifiers?
     YES NO
 (8) Will the values in this field be used alone as
     implicit classifiers? YES NO
```

Figure 2-11 : « Questions sur le champ (MAKER) dans le système TEAM »

2.7.2 Portabilité par rapport aux SGBD

Une ILNBDD fournit une portabilité de SGBD, si elle peut être facilement modifiée pour être utilisée avec différents systèmes de gestion de BDD. Dans le cas où le système génère une requête dans un langage largement supporté (ex. SQL), il sera possible de transférer l' ILNBDD à d'autres SGBD, avec seulement quelques modifications.

2.7.3 Portabilité du langage de programmation et du matériel

Dés les années 80, la capacité d'importer une ILNBD sur différents matériels et environnements logiciels (systèmes d'exploitation), est devenue un aspect important.

2.8 La génération de réponses

Dans un système d'interrogation de BDD en langue naturelle, la génération de réponse est aussi importante que la formulation et l'interprétation de la question. Certaines ILN pour des BDD relationnelles retournent simplement les tuples retrouvés par la requête de BDD, cette approche n'est pas toujours convenable. Si le système est incapable d'interpréter la question, aucune réponse n'est retournée. Dans ce cas, les raisons d'échec doivent être expliquées à l'utilisateur (ex. mot inconnu, syntaxe complexe, information introuvable dans la BDD, etc.) ;

2.9 La mise à jour en langue naturelle

Bien que les problèmes d'interrogation des BDD en langue naturelle aient été largement étudiés, il y a eu relativement peu de travaux sur le traitement des mises à jour de BDD exprimées en langue naturelle [Bozenn et Frederic, 1983].

Dans le système ASK [Bozenn et Frederic, 1983] les requêtes de mise à jour peuvent apparaître au cours d'un dialogue avec le système. Le dialogue suivant est tiré de [Bozenn et Frederic, 1983] .

```
  > What is the home port of the Tokyo Maru?
  Yokohama
  > Home port of the Tokyo Maru is Hong Kong.
  Yokohama has been replaced by Hong Kong as
home port of Tokyo Maru.
  > The cargo of the Tokyo Maru is coal.
  Coal has been added as the cargo of Tokyo
Maru.
  > What is the cargo of the Tokyo Maru?
  Grain
  Coal
```

Figure 2-12 : « Exemple d'une requête de mise à jour dans le système ASK »

ASK sait que chaque bateau doit avoir seulement un port. Par conséquent, dans **`'Home port of the Tokyo Maru is Hong Kong`**', il a remplacé `Yokohama` (préalablement connu comme le port de `'Tokyo Maru')` par `Hong Kong.` Contrairement, un bateau peut transporter plusieurs types de charges. Ainsi, dans **`'The cargo of the Tokyo Maru is`**

coal', ASK a ajouté `'coal'` aux types de charges pouvant être transportées par `'Tokyo Maru'`.

Cependant, satisfaire une requête de mise à jour en langue naturelle n'est pas toujours aussi directe [Androutsopoulos et al., 1995]. Les requêtes de mise à jour peuvent engendrer des effets indésirables ou inattendus[Kaplan et Davidson, 1983] [Kaplan et Davidson]. Ce-ci est due aux contraintes de BDD que l'utilisateur ignore [Androutsopoulos et al., 1995].

Soit l'exemple tiré et adapté à partir de [Kaplan et Davidson, 1983]

Table-employees

Employee	Salaire	Departement
Brahimi	25000	Finance
Ben ali	25000	Stock
Badis	35000	Ventes
Moussaoui	30000	Ventes

Table-departement

Departement	Directeur
Ventes	Si Youcef
Stock	Ait arbi
Finance	Salhi

Le dialogue suivant a eu lieu:

> lister les employés et leurs directeurs.

Employee	directeur
Brahimi	Salhi
Ben ali	Ait arbi
Badis	Si Youcef
Moussaoui	Si Youcef

> modifier le directeur de Badis de Si Youcef à Ait arbi.

Fait

> lister les employés et leurs directeurs.

Employee	Directeur
Brahimi	Salhi
Ben ali	Ait arbi
Badis	Ait arbi
Moussaoui	Ait arbi

On remarque que l'employé Moussaoui a aussi changé de directeur, passant ainsi de SiYoucef à Ait arbi, bien que cela n'était pas demandé. Ici, le système a modifié le nom du directeur du département de ventes de Si Youcef à Ait arbi.

2.10 Conclusion

Depuis les années 60, de nombreux systèmes d'interrogation de BDD en langue naturelle ont été réalisés et ont démontré des caractéristiques impressionnantes et constituent jusqu'aujourd'hui une référence pour toutes les recherches dans ce domaine (tel que LUNAR, BASEBALL, TEAM).

Actuellement, et avec l'avènement du Web, les interfaces en langue naturelle pour les bases de données tiennent une place de premier ordre dans les recherches en traitement de la langue.

Nous donnons les statistiques suivantes sur la répartition du budget en TLN (établies par le Clearinghouse for Natural Language Processing, Battelle Colunbus Laboratories) [Gilles,2000]**:**

- Interface en LN pour les bases de données (40%)
- Traduction automatique d'une langue naturelle en une autre (20%)
- Fabrication d'outils spécifiques pour systèmes intégrés (20%)
- Parcours de texte pour indexation automatique
- Génération de texte pour documents standardisés
- Reconnaissance de la parole.

Les problèmes auxquels doit faire face un système d'interrogation de BDD en LN, sont nombreux et lorsque nous devons réaliser un système entier, en considérant la sémantique, l'interrogation et la génération de réponse, la situation devient très compliquée.

Chapitre 3 Les modèles d'Acteurs

3.1. Introduction

Les systèmes parallèles et les systèmes distribués partagent un même modèle de calcul de base : des processus physiquement répartis qui fonctionnent d'une façon concurrente et interagissent entre eux afin d'accomplir une même tâche. Cependant, les systèmes parallèles et les systèmes distribués ont été développés comme des domaines de recherche distincts. En conséquence, chacun de ces deux modèles avait son propre environnement d'exécution.

Dans les systèmes parallèles, les processus (se communiquant fréquemment) sont supposés 'proches' l'un de l'autre -d'où le ratio calcul/communication, dans ce type d'applications, est plus petit que celui dans les applications distribuées. D'autre part, les systèmes distribués (réparties) concernent des processus largement 'dispersés' -la communication entre les processus est plus coûteuse que dans les systèmes parallèles.

Les tendances actuelles visent la convergence entre ces deux domaines. Ceci s'explique par deux arguments principaux: le premier est *architectural* (ex. les systèmes client/serveur adoptent de plus en plus l'architecture multiprocesseurs). Le second est le *potentiel d'Internet* (qui est vu comme une énorme machine parallèle répartie).

Le modèle Acteur fournit un modèle de calcul flexible qui supporte à la fois un traitement parallèle et réparti [Agha et Wooyoung,1999].

L'objectif de ce chapitre est d'introduire le concept d'acteurs et d'exposer quelques modèles basés sur ce paradigme.

3.2. Les acteurs

Un acteur peut être considéré comme un expert vivant en société et communiquant avec d'autres experts pour résoudre des problèmes. Chacun des experts peut lui-même être une société d'experts plus élémentaires. Le modèle est donc distribué (ou réparti). Les connaissances et les comportements sont diffusés parmi les acteurs, qui effectuent des tâches en parallèle, de manière indépendante [Gabrias, 1997].

Les acteurs communiquent entre eux par *envoi de messages* (le message est lui-même un acteur). Le *script* d'un acteur (analogue à la notion de *méthode* pour les objets) est un programme qui défini la réaction d'un acteur à la réception d'un message. En se basant sur ce script, l'acteur cible décide d'accepter le message (exécute son script local) ou de le rejeter. Dans le cas d'un rejet, l'acteur peut déléguer le message à d'autres acteurs (dits *proxy*), qui sont considérés plus aptes à la résolution du problème, c'est le *mécanisme de délégation.*

Un acteur se communique seulement avec ces *accointances*, c-à-d, les acteurs qu'il connaît. [Adriaens et Hahn, 1994]

3.3. Communications

L'activité [Labidi et Lejouad,1993] d'un acteur consiste à recevoir des messages et à élaborer des réponses. Un acteur peut *déléguer* les messages auxquels il ne peut pas répondre. Dans les paragraphes suivants, nous présentons les communications dans un modèle acteur.

3.3.1. Envoie de messages

Dans un monde d'acteurs, tout est acteur et le seul événement qui peut se produire est l'envoie de messages : un acteur émetteur transmet à un acteur récepteur un acteur message. Ceci se fait normalement d'une manière asynchrone, mais certains langages (comme ABCL) offrent aussi les communications synchrones ou anticipées (une communication anticipée permet à un acteur d'envoyer un message dont il n'utilisera le résultat qu'ultérieurement : l'acteur continue son activité et sollicitera le résultat au moment où il en aura besoin, et si le résultat n'est pas encore disponible, il se mettra en attente).

Les messages contiennent des *continuations* qui sont des informations supplémentaires permettant de désigner l'acteur auquel sera envoyé le résultat du message. Une continuation peut être un autre acteur ou un acteur crée spécialement pour recevoir ce résultat. [Labidi et Lejouad,1993]

3.3.2. Délégation

Il s'agit d'un mécanisme permettant à chaque acteur ne pouvant pas traiter un message de le *déléguer* à un autre acteur appelé *proxy* : le *proxy* prend en charge l'exécution du message et l'acteur délégant est immédiatement disponible pour exécuter d'autres messages. Le *proxy* connaît l'acteur délégant pour le cas où il aurait besoin d'informations supplémentaires. Pour éviter une régression à l'infini dans le processus de délégation, il y a des acteurs qui ne délèguent jamais : ce sont les acteurs primitifs. [Labidi et Lejouad,1993]

3.4. Quelques modèles d'acteur

3.4.1. Le modèle Act1

Le premier modèle d'acteur crée fut celui de Carl Hewitt [Lieberman,1990] du laboratoire de l'intelligence artificielle de MIT. Act 1 était écrit en MacLisp

Un *acteur* est un objet actif qui communique avec les autres acteurs par envoi de messages. Chaque message est également, lui-même, un acteur.

Les interactions entre acteurs étaient modélisées par des *"continuations "*.

Un acteur regroupe au sein d'une même entité [Amrouche,2004]:

1. *Des données locales,* appelées *accointances,* qui sont les autres acteurs qu'il connaît directement.

3. *Un comportement* : définit les actions que l'acteur entreprend au cours de son existence, le comportement est définit par un script, unique, qui filtre les messages reçus par l'acteur et active la partie appropriée du comportement.

- ***La synchronisation***

Act 1 dispose de plusieurs variétés de synchronisateurs [Lieberman,1990]:

- *ONE-AT-A-TIME*: est un acteur qui ne permet qu'un seul message à la fois. Lorsqu'un message est reçu, les autres messages venant sont obligés d'attendre la fin du calcul pour être traités. Un tel synchronisateur suppose l'existence d'une file d'attente de messages.
- GUARDIAN : c'est un synchronisateur plus général. GUARDIAN permet de retarder la réponse à un message, et de continuer à recevoir d'autres messages. GUARDIAN exige une gestion explicite des continuations.

La technique d'implantation d'acteurs synchronisateurs en Act 1 emploie les salles d'attente (*waiting rooms).* Ce sont des listes qui mémorisent les continuations afin de suspendre le calcul.

3.4.2. Le Modèle de Gul Agha

Dans ce modèle un acteur [Agha et Wooyoung,1999] peut être décrit par :

- *Une adresse* : boite aux lettres où se trouvent les messages reçus.
- *Un comportement* : les actions entreprises par les acteurs lorsqu'il traite le message courant.

L'exécution d'une méthode est atomique : une fois débutée, la méthode est exécutée sans interruption.

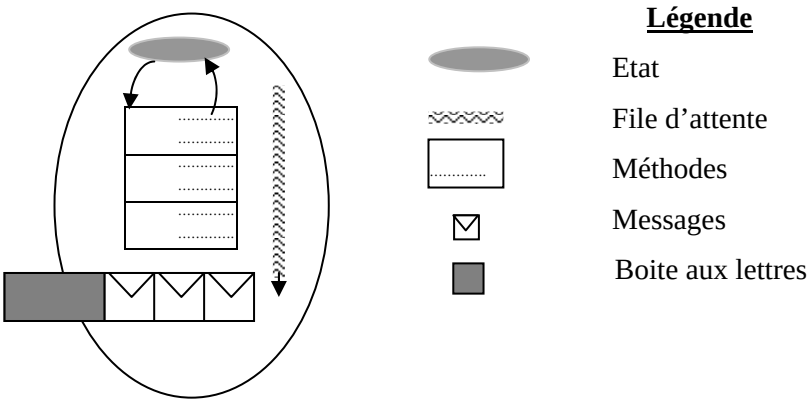

Figure 3-1: « un acteur encapsule des états, un ensemble de méthodes et une file d'attente ».

A la réponse à un message, un acteur peut envoyer des messages, créer des acteurs, ou changer ses données locales [Agha et Wooyoung,1999].

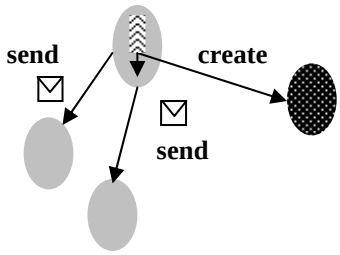

Figure 3-2 : « les primitives du modèle Acteur. Les acteurs peuvent envoyer de messages, créer de nouveaux acteurs et changer leur état local ».

3.4.3 Le modèle ABCL (An object Based Concurrent Language).

Langage développé par l'équipe de *Yozenawa* à **T**okyo **I**nstitute of **T**echnology.

- ***Quelques caractéristiques de ce modèle***

- Un acteur possède une mémoire locale, décrivant son état (valeur des accointances) et un script décrivant son comportement.
- Un acteur peut avoir un délégué à qui il envoi les messages non compris (n'appartiennent pas à son script).
- Il existe deux types de messages: *ordinaire* et *express*, ils sont rangés dans deux files différentes. A la réception d'un message ordinaire on le place dans la file des messages ordinaires. Il sera traité lorsque l'acteur termine le traitement de tous les messages précédents. A la réception d'un message express, il y a deux cas de figures :
  - *Si l'acteur traite un message ordinaire, il sera interrompu, il va traiter le message express. A la fin il reprend le traitement du message précédemment abandonné.*
  - *Si l'acteur traite un message express, le message sera mis dans la file des express* [Amrouche,2004]

3.5. Conclusion

Les systèmes d'acteurs sont caractérisés par la communication par envoie de messages et un traitement local : un acteur détient les informations locales, et son comportement est indépendant de celui des autres. Ils représentent une distribution à la fois des connaissances, des résultats partiels et des méthodes utilisées pour la résolution du problème [Labidi et Lejouad,1993].

Les langages d'acteur ont été à l'origine, développés dans un laboratoire de l'intelligence artificielle. Les systèmes d'intelligence artificielle sont complexes et exigent souvent d'exécuter plusieurs tâches en même temps. Les langages d'acteur avec leur capacité du traitement parallèle, sont une solution idéale pour l'exécution de telles applications. *PLASMA* et *Acte1* sont, par exemple, des langages développés et utilisés par de nombreux chercheurs de l'intelligence artificielle.

Chapitre 4 Le modèle Acteur et le TALN

4.1 Introduction

Les «*langages d'acteurs*» (Plasma, SmallTalk et Loops), permettent une programmation originale en introduisant une nouvelle structure de contrôle : la transmission de messages. Chaque objet, i.e. chaque élément du langage, est décrit sous deux aspects : un aspect statique, ensemble d'attributs pouvant prendre une ou plusieurs valeurs, et un aspect dynamique, ensemble de méthodes, comportements qui sont déclenchés à la réception d'un message

Ces langages succèdent aux langages relationnels, dont le plus connu est Prolog, qui devaient équiper la fameuse cinquième génération d'ordinateurs, prévue en particulier par les Japonais. Ces langages présentent des conditions nécessaires pour gérer des processeurs parallèles.

Ce type de langage, mis à l'honneur dans plusieurs familles de produits industriels (Xerox, Apple MacIntosh), est de plus en plus utilisé pour le TALN, tant à cause de son confort en tant qu'environnement de développement, que de sa puissance pour la représentation d'objets évoluant dans des micro-univers dynamiques, donc de la mise en oeuvre de modélisations de l'activité cognitive humaine. Il permet de modulariser les connaissances du système. Ainsi, chaque module (chaque objet) peut choisir la méthode la plus adaptée pour résoudre un problème.

L'objectif de ce chapitre est de montrer les apports du modèle acteur pour le TALN. Nous allons discuter d'une part, les lacunes du traitement séquentiel et d'autre part les apports du traitement parallèle de la langue. Nous donnerons l'intérêt d'un traitement distribué et présenterons quelques systèmes de traitement de la langue exploitant le paradigme d'acteur.

4.2 Le parallélisme et le traitement du langage naturel

4.2.1 Les lacunes du traitement séquentiel du langage naturel

Dans le traitement automatique de la langue naturelle, tous les chercheurs s'accordent à dire que les systèmes séquentiels sont incapables de supporter la grande variété des phénomènes linguistiques humains [Eiselt et Granger, 1994], [Nijholt,1994], [Adriaens et Hahn, 1994]. Ainsi, dans [Eiselt et Granger, 1994] les auteurs ont décrit trois phénomènes importants : l'indépendance fonctionnelle entre le traitement syntaxique et sémantique, l'interaction entre le traitement lexical et pragmatique, et les accès multiples du sens d'un mot.

- ***L'indépendance fonctionnelle entre les traitements syntaxique et sémantique***

Chomsky a noté que les gens peuvent attribuer une forme syntaxique correcte à des phrases sémantiquement anormales, comme :

Colorless green ideas sleep furiously.

Inversement, Charniack et Winograd ont noté qu'il est possible d'assigner un sens à des chaînes agrammaticales constituées des mots sémantiquement liées, comme par exemple :

Skid crash hospital. (Winograd)

Fire match arson hotel. (Charniack)

En donnant ces simples observations, il est difficile d'opposer l'hypothèse que le traitement syntaxique et sémantique *peuvent* fonctionner indépendamment l'un de l'autre. Cela ne veut pas dire que ces processus *doivent* fonctionner indépendamment ou qu'il n'y a aucune interaction entre eux. Les deux processus contribuent à une interprétation précise d'un texte.

Plusieurs systèmes de compréhension du langage naturel ignorent cette simple mais importante contrainte d'indépendance fonctionnelle de la syntaxe et de la sémantique. Des systèmes tels que Lunar de Woods 1970, ou bien SHRDLU de Winograd, 1973 exécutent en premier une analyse syntaxique suivie d'une analyse sémantique (l'analyse sémantique dépend d'une analyse syntaxique correcte).

D'autres groupes de systèmes tel que SAM de Cullingford 1978, PAM de Wilensky 1978, Ms. Malaprop de Charniack 1978 et ARTHUR de Granger 1980 emploient une approche à deux étapes pour le traitement du texte. La première étape convertit le texte entré en une représentation intermédiaire basée sur un bas-niveau sémantique (c-à-d lexical) et connaissances syntaxiques enregistrées comme définitions de mot. La représentation

intermédiaire sera ensuite traitée par une seconde étape utilisant des connaissances pragmatiques pour remplir la représentation finale d'une entrée. (Ces systèmes ne peuvent pas déterminer l'exactitude syntactique d'un texte sans une évaluation sémantique correcte).

- ***L'interaction entre les processus lexical et pragmatique***

Le second phénomène de compréhension de la langue et qui est rarement pris en compte par les systèmes computationnels (de calcul), est que les décisions d'inférence lexicale influencent sur les décisions d'inférence pragmatique, et que ces décisions pragmatiques à leurs tours, influencent immédiatement sur les décisions lexicales[1].

Les chercheurs semblent être dans l'accord qu'il y a une dépendance entre les deux niveaux de traitement.

- ***Les divers accès aux significations d'un mot***

Les approches d'accès lexical et de désambiguïsations se divisent en deux classes : *modèles à accès sélectif* et *modèles à accès exhaustif.*

Les modèles à accès sélectif suivent la prémisse que le contexte existant prédétermine, dans une certaine mesure, les significations à rapporter quand un mot ambigu est traité. Certains modèles psychologiques utilisent une recherche séquentielle dans laquelle les significations d'un mot sont examinées dans un ordre déterminé par leur fréquence d'utilisation, c-à-d, que la signification la plus fréquemment utilisée est d'abord sélectionnée et évaluée par rapport au contexte. Si cette signification est appropriée la recherche se termine, sinon la prochaine signification est appelée et évaluée, et ainsi de suite jusqu'à ce qu'une signification appropriée soit trouvée.

La limitation du modèle à accès sélectif est que dans certaines situations, la signification la plus appropriée du mot ambigu n'est jamais envisagée. Ce modèle suppose, que le sens approprié au contexte le plus fréquemment utilisé sera toujours la meilleure signification (ce qui ne peut pas être toujours correct). Cette limitation sera surmontée si toutes les significations d'un mot ambigu sont considérées en même temps, comme le postule le modèle *à accès exhaustif.*

---

[1] *Les inférences pragmatiques sont des connections entre les hypothèses escortées par un texte et les connaissances de l'univers, déjà enregistrées en mémoire. Autrement dit, ce sont des décisions au sujet duquel une inférence représente mieux une relation implicite entre événement ou état explicitement donnés dans un texte* [Eiselt et Granger, 1994].

Dans ce modèle toutes les significations du mot ambigu sont activées en parallèle indépendamment du contexte, et la signification la plus conforme au contexte sera choisie.

Cette approche semble être la meilleure explication du traitement lexical humain et a été incorporé dans plusieurs systèmes informatiques dans le domaine de la compréhension de la langue.

4.2.2 Les apports du traitement parallèle

Les machines parallèles offrent non seulement une vitesse plus augmentée mais aussi une nouvelle perspective en informatique (systèmes industriels, réseau de neurone, et autres applications de l'intelligence artificielle) [Minhwa et Dan, 1994]. Dans [Adriaens et Hahn, 1994] les auteurs citent trois considérations pour lesquelles l'appel au parallélisme devient adéquat:

- ***Puissance de calcul***

La nécessité d'incorporer le parallélisme dans les systèmes de TALN a été exprimée la première fois dans le secteur de reconnaissance et de compréhension de la parole. Mais il y a eu, également, des études du parallélisme dans le domaine de compréhension du texte, comme avec le mécanisme appliquant les scripts distribué SAM de Cullingford (voir aussi [Eiselt et Granger, 1994]).

Cela provient des expériences faites dans des projets traitant de grandes quantités de données, de vagues espaces de recherches et du degré élevé d'ambiguïté qui nécessitent l'intégration de diverses sources de connaissance afin de produire des interprétations raisonnables d'une entrée ambiguë.

- ***Exécution accélérée :***

Les études d'évaluation d'exécution considèrent que les résultats d'un programme parallèle sont nettement au-dessus de l'algorithme séquentiel le plus efficace**.** Dans les systèmes de TAL, deux types de parallélisme peuvent être définis : le parallélisme dans un seul composant (*parallélisme intra-composant* dans le processus morphologique, lexical ou syntaxique), et le parallélisme complet (*parallélisme inter-composant).* Les expériences dans le domaine du parallélisme intra-composant adaptent principalement les algorithmes d'analyse séquentiels bien connus en parallélisme (un aperçu du parallélisme d'un analyseur d'Early et Cocke-Younger-Kasami est donné par Nijholt (voir [Nijholt,1994]). Plusieurs expériences ont été réalisées pour démontrer que le temps d'exécution d'un analyseur parallèle est au-dessus de sa version séquentielle. Par exemple, Lozinskii et Nirenburg ont proposé un algorithme (nommé LCB), sa complexité est de $\theta((\log n)^4)$, n est la longueur de la phrase si n processeurs sont

utilisés. Grishman et Chitrao présentent un analyseur CYK et ont démontré que sa version parallèle s'exécute 5 à 6 fois plus rapidement que celle séquentielle (voir [Thompson,1994]).

- ***Plausibilité cognitive :***

L'attraction majeure pour que la communauté de TAL soit impliquée dans le parallélisme est peut être due soit à l'observation de correspondance entre l'architecture et le fonctionnement du cerveau humain avec les machines parallèles (comme le suggère le courant connexioniste) ou soit en reliant le comportement du traitement parallèle observable chez les être humains aux principes abstraits de calculs et considèrent ces relations comme critère d'adéquation entre les modèles computationnels et cognitifs du langage naturel (comme postulé par le courant psychologique qui mettent en exergue les activités parallèles dans le traitement d'une entrée linguistique). Par exemple, le système ATLAS (présenté dans [Eiselt et Granger, 1994]) simule la façon parallèle dont laquelle un humain analyse et interprète des textes.

4.3 Pourquoi un traitement distribué de la langue naturelle ?

Les réalisations des systèmes de compréhension des textes (des expériences similaires ont été faites pour la compréhension de la parole) ont identifié la nécessité d'une disponibilité simultanée et d'une interaction mutuelle des sources multiples de connaissance dans des conditions d'architectures distribués et des modes de traitements parallèles. Ceci coïncide avec les résultats des sciences cognitives qui conçoivent les textes comme résultat d'une forte interaction d'un traitement concurrent des niveaux syntaxique, sémantique et pragmatique du langage naturel.

Par conséquent, les architectures des systèmes de compréhension des textes ont reflété ce besoin évident d'une communication continue et d'un échange d'informations entre les différents composants du système en insistant sur des lignes de communication directe ou des structures de données communes pour l'envoie de messages et l'échange de données.

C'est ce type de *macrodistribution* dans les niveaux de représentation et dans la gestion des différentes sources de connaissance qui fait du traitement du langage naturel un cas véritable pour l'adaptation des modèles distribués [Hahn,1994].

Les systèmes d'acteurs fournissent une structure méthodologique excellente pour réaliser des grammaires (données) distribuées, d'une part et leurs implémentations en terme d'analyseurs distribués d'autre part [Hahn,1994].

4.4 Des systèmes TALN utilisant le modèle acteur

La première recherche concernant la distribution des connaissances linguistiques et leur traitement en parallèle est due à Kaplan (1973). Qui préconise une conceptualisation du processus de traitement de la langue et son implémentation comme une collection de processus parallèles se communiquant d'une façon asynchrone [Hahn,1994].

L'approche d'envoi de messages dans le TALN a été considérée par plusieurs chercheurs:

Yonezawa a présenté un analyseur orienté-objet parallèle pour une grammaire à contexte libre. Cet analyseur a été implémenté en utilisant le langage ABCL/1. Sa complexité étant de O(n*h), tel que ***n*** est la longueur de la phrase à analyser et ***h*** la hauteur de l'arbre syntaxique. L'approche de Yonezawa consiste à considérer les symboles terminaux et non-terminaux d'une grammaire comme des objets reliés dans un réseau et qui concourent à la construction de l'arbre syntaxique d'une phrase [Yonezawa et Ohsawa, 1994]**.**

Ne divergeant pas de l'approche de Yonezawa, Dekang Lin & Randy Goebel [Lin et Goebel,1994] ont présenté un analyseur d'une grammaire à contexte libre par envoi de messages. L'algorithme proposé est similaire à CYK et étant d'une complexité $O(|G|n^3)$ (où n est la longueur de la phrase et |G| et le nombre *d'occurrences* des symboles non-terminales dans l'ensemble des règles de la grammaire). L'analyseur est implémenté en C++.

L'analyseur *Expert Parallèle (PEP)* (M.Devos, G.Adriaens) ([Devos et Adriaens, 1994], [Small,1980]) est un modèle d'analyse du langage naturel basé sur une interaction parallèle des sources de connaissance actives (experts). Ces experts développent une analyse conceptuelle de fragment de texte en échangeant des messages contenant des informations de nature lexicale, syntaxique et sémantique/pragmatique.

PEP a été implémenté en *Flat Concurrent Prolog* en utilisant l'environnement de programmation *Logix.*

Le système *ParsTalk,* de Udo Hahn ([Hahn, 1996], [Hahn,1994], [Lohuizen, 1998]) est considéré comme une solution pour une base formelle appropriée d'un calcul concurrent basé sur le paradigme Acteur. Il conçoit une communication entre et à l'intérieur de différentes

sources de connaissance (grammaire et connaissances de discours) comme épine dorsale pour des tâches complexes de compréhension de la langue.

Remarque : ces systèmes seront présentés en détail dans le chapitre suivant.

4.5 Conclusion

L'ambiguïté de la langue naturelle due à sa richesse et à sa diversité d'expression rend le processus du traitement très complexe. La compréhension d'un texte constitue toujours un problème majeur pour toute application de TALN. La résolution de ce problème a identifié la nécessité d'une disponibilité simultanée et d'une interaction mutuelle des sources multiples de connaissances. Ceci a rendu évidente l'importance d'une communication continue et d'un échange d'informations entre les différents composants du système. De telles exigences, impliquent le choix d'une approche permettant d'une part la distribution des connaissances et d'autre part d'assurer une communication efficace entre les sources de connaissance. Dans une telle approche les niveaux de traitement seront vus comme des processus actifs communiquant entre eux, et chaque niveau sera considéré comme une composition d'un ensemble d'objets dynamiques. Le modèle acteur, se présente comme une solution idéale pouvant satisfaire de telles exigences. Par sa définition, un acteur est considéré comme un expert vivant en société et communiquant avec d'autres experts pour résoudre des problèmes. Chacun des experts peut lui-même être une société d'experts plus élémentaires. Le modèle est donc distribué (ou réparti). Les connaissances et les comportements sont diffusés parmi les acteurs, qui effectuent des tâches en parallèle, de manière indépendante. L'exploitation du modèle acteur pour le traitement de la langue a constitué le sujet de plusieurs projets de recherche dont l'analyseur de Yonezawa, PEP et ParseTalk sont les plus connus.

Chapitre 5. Etude de quelques systèmes

5.1 Introduction

Dans ce chapitre seront présentés quelques systèmes de TAL déjà développés. Le but est de montrer les techniques utilisées par les systèmes actuels.

Il n'est pas possible de couvrir tous les systèmes, cependant nous allons considéré des systèmes qui nous paraissent intéressants pour notre recherche.

Nous nous intéresserons aux systèmes d'interface en langue naturelle ainsi que ceux basés sur le paradigme d'acteur.

5.2. Les systèmes d'interrogation des BDD en langue naturelle

Nous allons présenter cinq systèmes d'interface de BDD en langue naturelle écrite. Le premier système, SystemX est le plus mature. Tamic-P se concentre sur l'interrogation de plusieurs BDD décentralisées. Pour les deux systèmes LOLITA et SCHISMA, plusieurs recherches sont en cours pour l'implémentation du parallélisme. Nous présenterons également PRICISE qui se veut une nouvelle théorie dans le domaine d'interrogation de BDD en langue naturelle.

SystemX (1988-1996)

Développeur	Popowich, Cercone, Fass
Organisation	Natural Language Laboratory at Simon Fraser University
Références	[Lohuizen, 1998], [Fass,1998]

Objectif Interface pour une base de données en langue naturelle (écrite).

Domaine/Couverture est apparu en deux versions :

SystemX(pré-90 :1988-1990) est une interface en langue naturelle pour une BDD concernant les informations du conseiller académique de l'université de Simon Fraser. Il assure une couverture totale de la langue anglaise et peut manipuler les formes passives, impératives, possessives, relatives et les quantifications.

SystemX(post-90 :1990-1996) la nouvelle version de SystemX est utilisée pour l'accès à quatre BDD différentes :

- Rogers CableSystems customer service operations information.
- National Science and Engineering Research Council grants information.
- Environment Canada information.
- Rogers CableSystems cable and pay-per view product information.

Architecture SystemX a une conception modulaire, contenant un module de compréhension de la langue naturelle ainsi qu'un module d'interrogation de BDD. Le module de compréhension de la langue naturelle contient trois composants : un étiqueteur lexical, un analyseur syntaxique et un interpréteur sémantique qui produit une forme logique. Le module d'interrogation de BDD contient un module d'interprétation de la forme logique en SQL.

Techniques l'analyseur lexical contient deux sous-systèmes : TEMPLATE et MORPHOS. TEMPLATE emploie la forme de certains mots pour identifier certaines catégories comme les

noms propres et les chiffres. MORPHOS est un analyseur morphologique qui emploie un ensemble de règles pour détecter la fin des mots et identifier leur racine. Un arbre syntaxique sera construit par l'analyseur en utilisant des règles grammaticales. Cet arbre sera envoyé à l'interpréteur sémantique qui contient un ensemble de règles sémantiques et un dictionnaire sémantique. Chaque entrée du dictionnaire est une structure contenant la description d'une entité de BDD en rapport avec chaque mot. Les informations de ces structures seront attachées à l'arbre d'analyse (l'arbre syntaxique) et les règles sémantiques seront alors appliquées pour construire une représentation sémantique. Un composant dit Pathfinder est chargé de la résolution des ambiguïtés qui n'ont pas pu être résolues par l'interpréteur sémantique. Pathfinder utilise des informations sémantiques contenues dans le modèle conceptuel de la BDD pour résoudre l'ambiguïté générée. Si Pathfinder est incapable de résoudre l'ambiguïté, il interroge l'utilisateur pour plus de précision.

La version post-90 de SystemX utilise une grammaire HPSG. Il contient deux analyseurs : un écrit en LISP et l'autre en PROLOG.

L'apprentissage plusieurs recherches ont été effectuées pour l'automatisation du processus de génération d'un modèle conceptuel de BDD. La création d'un modèle conceptuel est l'une de principales et grandes tâches que nécessite la personnalisation d'une interface en langue naturelle pour une nouvelle BDD.

Tamic-P (1997-1998)

Développeur	Hans Uszkoreit
Organisation	NPS (Rome) Coordinator,; Quinary SpA (Milano); Cap Gemini; Innovation (Boulogne); OFAI (Vienna); IRST (Trento); DFKI GmbH
Objectif	Interface en langue naturelle écrite pour plusieurs BDD décentralisées.
Statut	
Domaine	Tamic-P peut connecter à plusieurs BDD dans le but de collecter des informations concernant les pensions et la sécurité sociale.
Lien	http://www.dfki.de/pas/f2w.cgi?ltp/tamic-e
Référence	[Lohuizen, 1998]

LOLITA (1994---)

Développeur	R. Garigliano, S. Peyton Jones
Organisation	Lab for Natural Language Engineering, University of Durham; Rolls Royce; University of Glasgow
Statut	Version démonstration disponible (la version parallèle non encore disponible)
Lien	http://www.dur.ac.uk/ ~ dcs1cac/DEAR/DEAR.html http://www.dur.ac.uk/ ~ dcs0www3/lnle/system.html ftp://ftp.dur.ac.uk/pub/comp-sci/lolita (articles)
Référence	[Lohuizen, 1998], [Morgan, 1994]

Objectif Interface (Parallèle) pour les BDD.

Architecture Le noyau de LOLITA offre trois fonctionnalités principales :

1. La conversion d'un texte exprimé en langue Anglaise en une représentation sémantique interne (basée sur les réseaux sémantiques) ;
2. Une inférence dans le réseau sémantique ;
3. La conversion des portions du réseau sémantique en langue naturelle (l'Anglais).

Le système utilise 1500 règles de grammaire Anglaise et couvre aussi les langues Espagnole et Chinoise.

Performances Le parallélisme est prévu d'être exploité dans ce projet en utilisant Haskell (un langage de programmation parallèle), puisque le système entier est écrit en Haskell. La version parallèle n'est pas encore disponible, le projet est en progression.

SCHISMA (1995--)

Développeur	Anton Nijholt et. al.
Organisation	Parlevink subproject, Twente University; KPN Research
Lien	http://wwwseti.cs.utwente.nl/ schaake/schisma.htm
Référence	[Lohuizen, 1998]

Objectif Interface en langue naturelle.

Domaine Réservation et information du théâtre (hollandais).

Architecture Le système comporte un analyseur morphologique, un module de correction de fautes (MAF), un analyseur basé sur l'unification (PARS), un gestionnaire de dialogue et un générateur.

Techniques L'analyseur de grammaire d'unification est implémenté en C++.

Remarque Le système utilise une interface écrite, mais une attention est donnée pour l'utilisation d'une interface vocale. Des recherches relatives à l'analyse parallèle sont en cours de réalisation.

PRECISE (2003)

Développeur	Ana-Maria Popescu et al.
Organisation	Department of Computer Science & Engineering, University of Washington.
Statut	Expérimental
Lien	http://www.cs.washington.edu/research/nli
Reference	[Popescu et Etzioni, 2004], [Popescu et al., 2003]

Objectif Offrir un système d'interrogation portable.

Domaine PRECISE a été empiriquement testé avec de centaines de questions sur trois bases de données différentes concernant le domaine des restaurants, de l'emploi, et la géographie.

Techniques Le principe de PRECISE s'applique sur des phrases de type *wh-word*, autrement dit, des questions dans l'ensemble {*'what', 'which', 'where', 'who', 'when'}*. Dans une base de données relationnelle, chaque attribut possède un ensemble de *wh-word* compatible.

Le comportement de ce système est illustré par un exemple dans la figure ci-dessous :

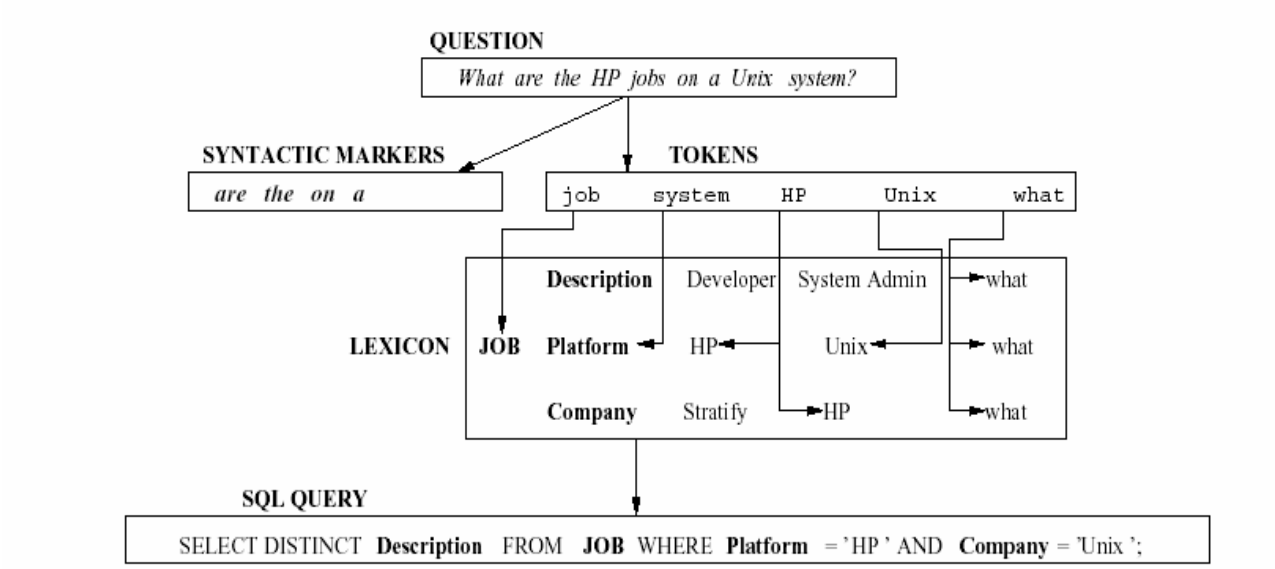

Figure 5-1 : « Transformation de la question 'What are the HP jobs on a Unix System' concernant une BDD contenant une seule relation, JOB, ayant les attributs Description, Platform, Company ».

La question *'What are the HP jobs on a Unix System'* se rapporte à une seule table **'job'** dont les attributs sont **Description, Platform, Company.** Un étiqueteur produit un étiquetage complet de la phrase : **(what, HP, job, unix, system)** et supprime quelques marqueurs syntaxiques comme **'the'** et **'a'**. PRECISE identifie l'ensemble des éléments correspondants à chaque étiquette[1]. Les étiquettes de **what, HP** et **unix** sont des valeurs, l'étiquette de **system** est un attribut et celle de **job** est une relation. Ces informations seront utilisées par le module *générateur de requête SQL* afin de générer la requête SQL correspondante et produire ainsi la réponse.

Architecture L'architecture de PRICISE se compose de six modules [Popescu et Etzioni, 2004] (voir figure 5-2):

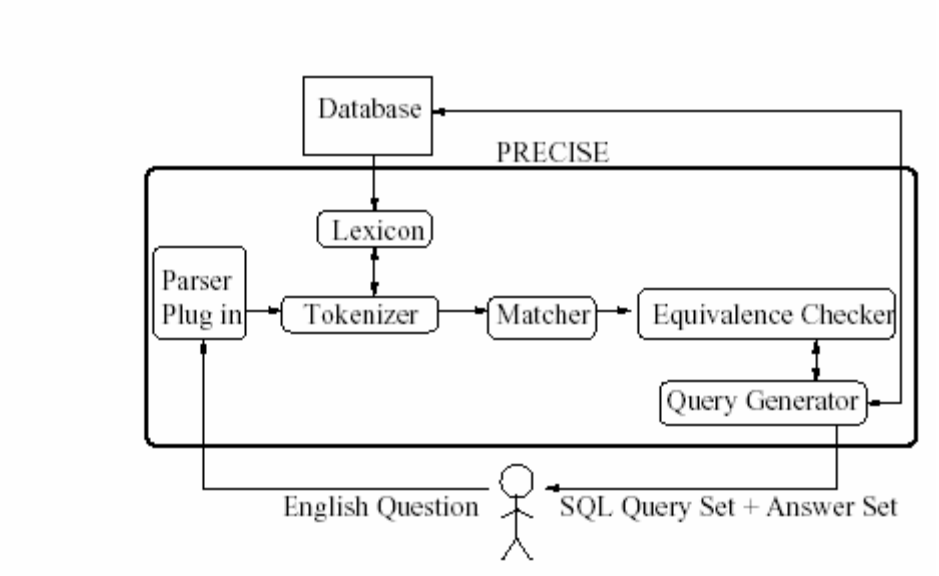

Figure 5-2 : « Architecture du système PRECISE »

Le lexique (Lexicon) extrait tous les noms des éléments de la base de données (ce sont les racines) et ensuite leurs définis des synonymes. Ces éléments seront stockés dans une table de hachage indexée par racine.

L'étiqueteur (Tokenizer) l'étiqueteur reçoit en entrée une question en langue naturelle et produit l'ensemble des étiquettes correspondant à chaque mot.

Le projecteur (Matcher) le projecteur constitue la clé d'innovation dans PRECISE. Le problème de recherche d'un élément de BDD correspondant à une étiquette ambiguë, est réduit à un problème de correspondance de graphe.

[1]Le lexique contient des synonymes (ex. 'system' et 'platform' sont des synonymes).

L'analyseur (parser plug in) l'analyseur relie les étiqueteurs en un arbre syntaxique. Ces liaisons seront utilisées par le projecteur (Matcher) pour sélectionner les projections valides (seulement les interprétations sémantiques qui satisfont les contraintes des liaisons syntaxiques représentent des projections valides).

Le générateur de requêtes (Query Generator)

Le générateur de question utilise les éléments de la base de données sélectionnés par le Matcher pour formuler une requête SQL bien formée. Le système prend en charge les requêtes mono table ainsi que les requêtes multi tables.

5.3 Les systèmes à base d'Acteurs

Les systèmes que nous allons décrire dans ce paragraphe sont tous basés sur le modèle de calcul d'Acteur et le traitement parallèle, bien que les réelles implémentations parallèles ne sont pas encore achevées. L'objectif de ce paragraphe est de savoir dans quel niveau de traitement et comment est exploité le paradigme d'acteur en traitement de la langue.

PEP (Parallel Expert Parser) (1994)

Développeur	Mark Devos, Geert Adriaens
Organisation	University of Leuven, Dept. Computer Science; Siemens Nixdorf Information Systems
Statut	Expérimental
Références	[Devos et Adriaens, 1994], [Lohuizen, 1998], [Small,1980]

Objectif L'analyse de la langue naturelle.

Domaine Jusqu'à présent, le système n'est pas encore testé sur un domaine réel.

Architecture L'analyseur expert parallèle (PEP) est un modèle d'analyse de la langue naturelle basé sur une interaction parallèle des composants de connaissances distribués. Ces composants sont des entités actives (experts) qui construisent une analyse conceptuelle d'un fragment de texte par échange de messages contenant des informations de nature lexicale, fonctionnelle (syntaxique) et sémantique/pragmatique.

PEP est implémenté en *Flat Concurrent Prolog (FCP)* en utilisant l'environnement de programmation *Logix*.

ParseTalk (1996--)

Développeur	Udo Hahn
Organisation	Albert-Ludwigs-Universität Freiburg, Linguistisch Informatik/Computerlinguistik Germany.
Statut	Expérimental.
Référence	[Hahn, 1996], [Lohuizen, 1998], [Hahn,1994]

Objectif Compréhension de la langue naturelle des textes sans restriction sur un domaine spécifique dans le but d'un apprentissage automatique.

Domaine Deux domaines d'essai ont été exploités: revues de produits informatiques et les rapports de résultats médicaux.

Technique ParsTalk représente un traitement orienté-objet distribué de la langue naturelle basé sur le modèle acteur. Chaque niveau d'analyse –conceptuel, opérationnel, et connaissances linguistiques- sont décrits en terme d'acteur.

L'analyseur a été développé en C sous Unix et implémenté sur une station SUN

Architecture L'analyseur est prévu pour le traitement parallèle. Il est basé sur le modèle de calcul d'acteur.

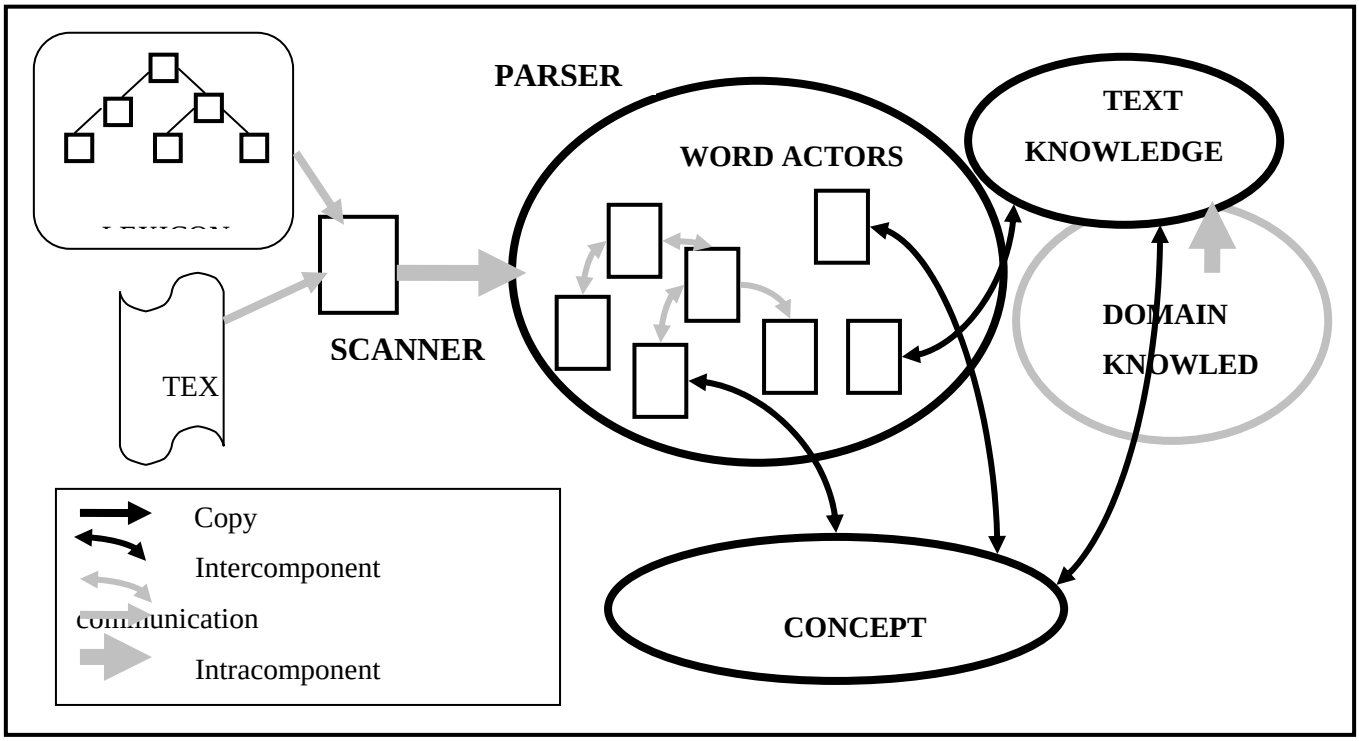

Figure 5-3 : « Architecture de ParseTalk »

Cette figure fournit une vue d'ensemble du noyau d'architecture du traitement linguistique. On distingue trois composants principaux : *l'analyseur – Base de connaissance – Etudiant de concepts.*

Quand un nouveau texte est introduit dans le système, une copie de la base du domaine est tirée, plus tard désignée sous le nom *base de connaissance du texte.* La base de connaissance de textes est incrémenté pendant que le processus de compréhension de texte procède. La base de connaissance de texte est interrogée sans interruption et mise à jour par les deux autres modules : l'analyseur et l'étudiant de concepts. L'analyseur se compose d'un ensemble de *mots acteurs*, chacun représente une forme simple de mot à partir du texte entré. Les mots acteurs sont crées par un module *de balayage* (scanner module) qui lit le texte en entrée mot par mot, localise chaque mot obtenu par le module *grammaire lexicale* (lexicon grammar), et les instances correspondantes à la description des classes objet, incorporant, ainsi, l'information fournie par le texte (exemple : priorité linéaire des unités lexicales). Après initialisation, chaque mot acteur débute la communication avec d'autres mots acteurs (*communication intra-composants).*

L'étudiant de concept (concept learner) est responsable d'identifier et d'assimiler de nouveaux concepts dans la base de connaissance de texte, c'est un processus qui exige une interaction intensive avec l'analyseur et la base de connaissance de texte.

L'apprentissage L'analyseur est capable de développer la connaissance dans un domaine spécifique.

L'analyseur de Yonezawa

Développeur	Akinori Yonezawa
Organisation	Dept. of Information Science, Faculty of Science, University of Tokyo.
Référence	[Yonezawa et Ohsawa, 1994]

Objectif Un analyseur orienté-objet parallèle pour une grammaire à contexte libre. Il est implémenté en utilisant le langage ABCL/1. Sa complexité étant de O(n*h), tel que ***n*** est la longueur de la phrase à analyser et ***h*** la hauteur de l'arbre syntaxique.

Technique Le modèle utilise l'analyse ascendante (bottom-up). Supposons qu'on a une règle de grammaire hors contexte telle que

$$VP \rightarrow V \ NP \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

Dans l'analyse ascendante, l'interprétation de cette règle est comme suit :

- *Dans une chaîne entrée, si sa première partie gauche peut être réduite à une catégorie (symbole terminal ou non terminal) V, et si sa deuxième partie droite peut être réduite à une catégorie (symbole terminal ou non terminal) NP, alors toute la chaîne peut être réduite à une catégorie VP ;*

Dans cette approche :

- Plusieurs objets (acteurs) sont autorisés à travailler concurremment, chacun exécute une tâche simple,
- Pour chaque symbole terminal ou non-terminal de la grammaire, un objet est associé ;
- Un objet reçoit des messages, manipule et enregistre des messages dans sa mémoire locale, et peut aussi envoyer des messages *asynchrones* pour d'autres objets (correspondant aux symboles terminaux ou non-terminaux), et
- Les données échangées entre les objets sont des arbres partiels d'analyse.

On suppose, dans la règle (1) ci-dessus, que l'objet qui agit sur le symbole V reçoit un arbre d'analyse partiel t1, et que l'objet qui agit sur le symbole NP reçoit un arbre d'analyse partiel t2. Si le symbole terminal qui est la limite droite de t1 est adjacent à celui de la limite gauche de t2, alors t1 et t2 peuvent être entrés ensemble et former un arbre plus large qui correspond au symbole VP dans la règle (1).

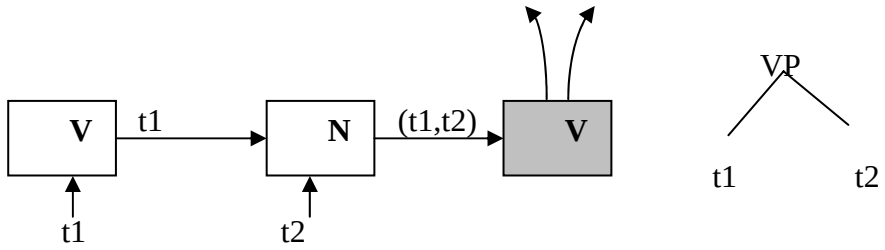

Figure 5-4 : « Les objets représentant une règle »

Comment construire l'arbre d'analyse (lexical) ?

Il est évident que l'objet NP reçoive t2 après que l'objet V reçoive t1 (ceci est dû à la nature du gauche à droite du traitement considéré). Dans le but de construire l'arbre lexical partiel, l'objet V doit envoyer t1 à l'objet NP. Après réception, l'objet contrôle si les limites de t1 et t2 sont adjacentes. Si t2 n'est pas encore reçu, NP enregistre t1 dans sa mémoire locale et reporte le contrôle de l'adjacence jusqu'à l'arrivée de t2. Si les deux limites ne sont pas adjacentes, l'objet NP stocke t1 dans sa mémoire locale jusqu'à l'arrivée d'une nouvelle

référence. Si le test d'adjacence est réussi, l'objet NP concatène t1 et t2 et les envoie à l'objet agissant sur le symbole VP.

L'objet VP construit avec t1 et t2 un arbre lexical partiel avec VP comme racine. Cet arbre lexical nouvellement crée sera alors distribuée par l'objet VP pour tous les objets ayant le symbole VP dans le côté droit de leur règle.

Les types d'objets Yonezawa a conçu trois types d'objets :

Type1 : correspondant au symbole gauche de la règle ;

Type2 : correspondant au symbole le plus à gauche des symboles du côté droit de la règle ;

Type3 : correspondant aux autres symboles du côté droit de la règle ; (si la règle contient plus de deux symboles du côté droit, chaque symbole sauf le plus à gauche est de type3).

Par exemple, dans la règle (1), VP est de type1, V est de type2 et NP de type3.

5.4. Conclusion

L'étude des systèmes précédents nous a permis d'obtenir quelques directives pour un système d'interface de BDD en langue naturelle que nous allons résumer dans las points suivants:

- En plus des problèmes auxquels est confronté tout système d'ILNBDD (ex. les conjonctions, les quantifications les anaphores…(voir chapitre 2)), il doit permettre une couverture totale d'autres formes linguistiques (les formes passives, impératives, possessives… **(systemX)**) ;
- La gestion de dialogue ne doit pas être négligée (il faut pouvoir interagir avec l'utilisateur dans le cas où le système ne trouvera pas une interprétation valide d'une question **(SystemX, SCHISMA)**) ;
- L'automatisation du processus de génération d'un modèle conceptuel de BDD est l'une de principales et grandes tâches que nécessite la personnalisation d'une interface en langue naturelle pour une nouvelle BDD.
- L'utilisation d'une représentation sémantique de la phrase (voir **LOLITA)** permettra de faire des inférences, c'est-à-dire d'impliquer des connaissances qui ne sont pas explicitement mentionnées dans la question ;
- Le système PRECISE utilise une méthodologie basée sur le *Pattern-matching* (voir chapitre 2), il prend en charge les requêtes mono table ainsi que les requêtes multi tables.
- Le modèle acteur a été exploité en plusieurs niveaux du processus de traitement de la langue. Le système PEP l'a utilisé pour la description des composants linguistiques distribués se communicant entre eux.
- Dans ParsTalk l'utilisation du modèle acteur était plus poussée. Les connaissances ainsi les processus linguistique sont tous exprimés en terme d'acteur. (d'autres recherches sont en cours pour le traitement des formes elliptiques).
- Yonezawa a exploité le modèle acteur pour la génération d'un arbre syntaxique de la phrase.

En terme de ce chapitre, ces points devront nous permettre de positionner les tendances d'un système d'interface de BDD en langue naturelle.

Chapitre 6. La Conception

6.1. Introduction

Les problèmes auxquels doit faire face un système d'interrogation de BDD en LN sont nombreux et jusqu'à l'heure actuelle aucune méthode générale pour les résoudre n'a pu être proposée. Et lorsque nous devons réaliser un système entier, en considérant la sémantique, l'interrogation et la génération de réponse, la conception d'un tel système devient très compliquée. Cette complexité est due à la diversité des sources de connaissances que manipule le système, la nécessité d'une communication continue et d'un échange d'informations entre ses différents composants.

De telles exigences impliquent le choix d'une approche permettant d'une part la distribution des connaissances et d'autre part d'assurer une communication efficace entre les sources de connaissance. Dans une telle approche les niveaux de traitement seront vus comme des processus actifs communiquant entre eux, et chaque niveau sera considéré comme une composition d'un ensemble d'objets dynamiques. Le modèle acteur, se présente comme une solution idéale pouvant satisfaire de telles exigences. Par sa définition, un acteur est considéré comme un expert vivant en société et communiquant avec d'autres experts pour résoudre des problèmes. Chacun des experts peut lui-même être une société d'experts plus élémentaires. Le modèle est donc distribué (ou réparti). Les connaissances et les comportements sont diffusés parmi les acteurs, qui effectuent des tâches en parallèle, de manière indépendante.

Dans ce chapitre nous allons proposer une solution à base d'acteurs pour l'interrogation d'une BDD en langue naturelle.

6.2. Contexte de l’étude

Les avancés dans le domaine des machines parallèles et des réseaux à très haut débit laissent croire que les environnements répartis et parallèles formeront les nouveaux standards des architectures à venir. Le développement de logiciels doit donc s'adapter à ce type d'architectures pour bénéficier de leurs avantages.

Beaucoup de travaux et de recherches sont consacrés, ces dernières années, au domaine de bases de données parallèles et plus particulièrement aux SGBD objets parallèles.

Les modèles objet et acteur sont assez proches l'un de l'autre ce qui permet de les combiner et bénéficier de leurs avantages réunis (puissance de modélisation + parallélisme implicite). L'objectif est d'étudier cette intégration entre *objets* et *acteurs* dans le domaine des bases de données.

L'idée est de concevoir un concept offrant la même puissance de modélisation que le modèle objet tout en étant fortement orienté vers les systèmes ouverts et parallèles. La solution proposée est la conception des *"Acteurs de Base de Données (DB-ACT)"*. Ceux sont des objets autonomes et actifs représentant une base de données (schémas, extensions, contraintes,….). Ils collaborent à la résolution des requêtes et des programmes d'application en parallèle [Hidouci et Zegour, 2001].

6.2.1. Présentation du SGBD Act21

Le SGBD Act21 [Hidouci et Zegour, 2001] est basé sur un modèle de données combinant le modèle objet *'par classe'* et celui *'par acteur'*.

Dans ce modèle un acteur de base de données DB-act est un objet actif défini par un comportement (behavior), ce dernier est formé par un état (des variables locales : Les accointances) et un script (un ensemble de méthodes). Ce modèle est inspiré du modèle ABCL (Yonezawa). L'identité de chaque acteur est unique OID (Object IDentifier), de plus un nom symbolique peut être attribué à chaque acteur.

La communication entre acteurs est asynchrone ou synchrone, les messages sont traités suivant l’ordre d’arrivée, un par un. Dans le cas asynchrone le résultat sera envoyé à la continuation si elle existe. L’acteur peut avoir un délégué, a qu’il va envoyer les messages non compris.

On distingue trois types de DB-act :

- *Les Acteurs de typage (T-Act)*: C'est des objets actifs sensés représenter des types avec leurs extensions. Ils ont pour rôle de maintenir et de gérer les données stockées et de répondre aux requêtes des autres acteurs du système;
- *Les acteurs collection (C-Act):* C'est des objets actifs jouant le rôle de conteneur. Ils permettent de stocker une collection de données.
- *Les acteurs de requêtes (R-Act):* C'est des acteurs de programmation qui se chargent en général de résoudre des requêtes en collaboration. Ils n'ont pas de rôle prédéfini comme les deux premiers, leur script doit entièrement être spécifié par l'utilisateur.

Chaque acteur est caractérisé par: un *OID*, un *type* (T-Act, C-Act ou R-Act), un *comportement* (optionnel dans le cas d'un C-Act) et un *nom* (obligatoire dans le cas d'un T-Act).

Les T-Act et les C-Act sont dotés de méthodes prédéfinies qui définissent le comportement par défaut de ce type d'acteurs (gestion d'instances et de schémas dans le cas de T-Act et méthodes génériques de manipulation de collections dans le cas de C-Act). Le comportement est décrit à l'aide d'un langage de programmation (PACT) et permet de spécifier les champs d'un acteur (partie DATA) et les méthodes applicables (partie Script) en plus celles prédéfinies [Hidouci et Zegour, 2001].

Exemple de définition de comportement

```
Def_behavior TypeEmp {
    Data: Int nss;
          Float salaire;
          String departement;
          OID chef=NULL;
    Script:
          NomDuChef( OID e): Continuation A {
              String s:
              if (chef(e)!=NULL) {
                        s = get_field('nom', chef(e) )
                        send String(s) to A;
              }
            }
            EmpDudepartement( string dep): Continuation C {
               Select ('departement = #dep'): C      //#dep est le contenu du paramètre
            }
};

Def_Behavior TypePers {
    Data: String nom, prenom, adresse;
          Int age;
          OID pere = NULL, mere = NULL;
}
```

Figure 6-1 : « Comportements TypeEmp et TypePers »

Dans cet exemple le comportement d'un employé (TypeEmp) a été décrit par les champs: nss, salaire, département et chef. Ce dernier étant de type OID initialisé à NULL. Le script défini deux méthodes l'une retournant le nom du chef d'un employé donné et l'autre produit comme résultat l'ensemble des employés travaillant dans un département donné. Un deuxième comportement défini le type Personne (TypePers) comme étant l'agrégation de six champs (nom, prénom, adresse, âge, père et mère).

Les champs d'un acteur, définis dans la partie *data* du comportement, ne sont accessibles que par des méthodes, réalisant ainsi le principe d'encapsulation analogue aux langages objets par classe.

6.2.2. Les primitives du modèle

Le modèle offre des primitives qui peuvent être exécuté par les acteurs. Ces primitives gèrent les acteurs (création, destruction,…), les groupes et la communication entre les acteurs.

- ***Manipulation des acteurs***

Les primitives de manipulation des acteurs concernent la création, la destruction, l'identification et l'accès aux informations générales des acteurs. Ces primitives sont:

- *new_act(comportement, type, nom, délégué)* ; Créer un nouvel acteur avec le nom *nom* et de type *type* et un comportement *comportement* et le délégué *délégué.*
- *delete(oid_act) :* Supprimer un acteur identifier par l'OID *oid_act.*
- *new_oid () ;* Générer un nouvel OID *;*
- *get_oid(nom) ;* Récupérer l'OID de l'acteur par son nom *nom.*
- *get_name(oid_act)* ; Récupérer le nom de l'acteur par son OID *oid_act* .
- *is_active() ;* Tester si l'acteur existe.

- ***Les groupes***

Un groupe est un ensemble d'acteurs actifs, ces acteurs peuvent comprendre les mêmes messages. Pour la gestion de groupe (création, destruction, gestion des membres et les informations sur le groupe) on offre les primitives suivantes :

- *new_group (nom_groupe)*; Création d'un nouveau groupe.
- *del_group (nom_groupe**);*** Suppression d'un groupe.
- *add_to_group (oid_act , nom_groupe);* Insertion d'un acteur dans un groupe.
- *del_from_group (oid_act ,nom_groupe);* Suppression d'un acteur d'un groupe.
- *get_oid_grp (nom_grp **)*** ;Récupération de l'OID d'un groupe.
- *get_name_grp (oid_grp)* ;Récupération du nom d'un groupe.

- ***La communication***

La communication entre acteurs est réalisée par envoie de messages. Il existe trois types de communication : synchrone, asynchrone et la diffusion à un groupe d'acteurs :

- *send (message) : cont to act ;* Envoyer le message *message* à l'acteur *act* (envoi asynchrone*)* et le résultat est redirigé vers la continuation *conti.*
- *retour = send (message) to act ;* Envoyer le message *message à l'acteur act (envoi synchrone)* et l'acteur qui a émit le message est bloqué jusqu'au retour du résultat dans *Retour.*
- *broadcast (message) to oid_grp ;* Diffuser le message *message* vers tous les acteurs membres de groupe *oid_grp* .

Si le message est de la forme *'var= send message to act'* il devient alors synchrone, c'est-à-dire que l'exécution est suspendue jusqu'à ce que le résultat soit retourné vers la variable *'var'* [Hidouci et Zegour, 2001].

6.3. Contribution de notre étude

Dans le paragraphe précèdent nous avons introduit le SGBD d'acteur ACT21 [Hidouci et Zegour, 2001] en présentant son principe et ses primitives, nous donnerons ci-après son architecture :

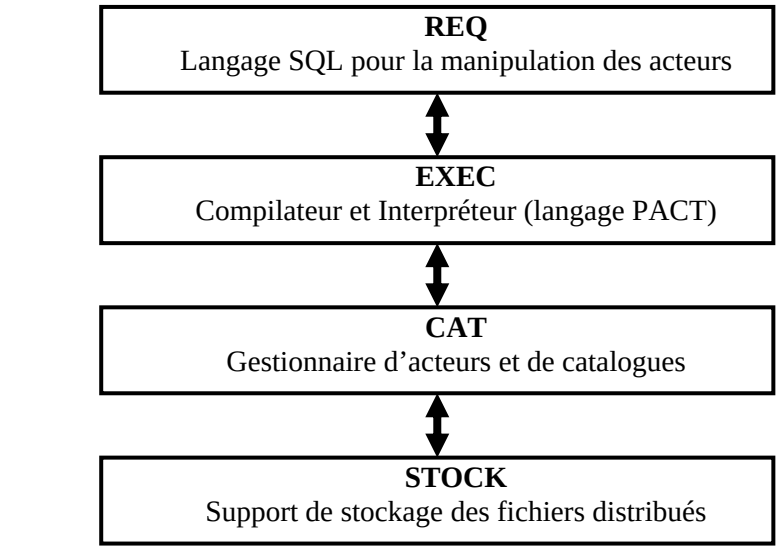

Figure 6-2 : « L'architecture de ACT21 »

Notre but n'est pas de décrire les fonctionnalités de tous ces modules, mais de présenter celui ayant un impact sur le choix de notre sujet : ***le module REQ.***

Le module REQ : ce module offre aux utilisateurs un langage de requêtes déclaratif (type SQL) et prend en charge son exécution en générant les différents acteurs nécessaires à la résolution de la requête. Son rôle est de traduire les requêtes en un programme écrit en PACT et appelle l'interpréteur du module EXEC pour contrôler leur exécution.

Notre travail consiste à proposer une architecture d'un système autorisant la manipulation de la BDD en langue naturelle.

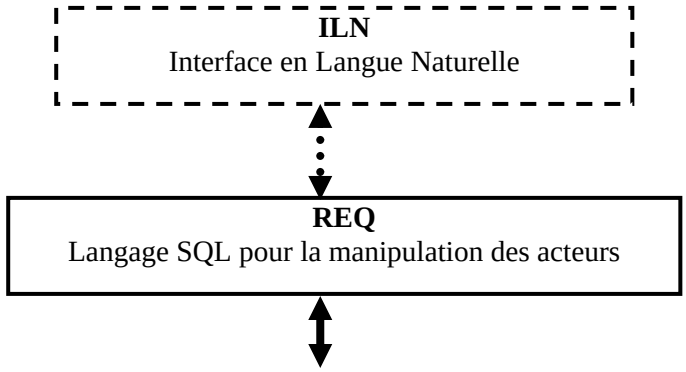

Figure 6-3: « L'intégration d'un module d'interface en langue naturelle au SGBD ACT21 »

6.4. Caractéristiques d'une méthodologie à base d'acteurs

Avant d'aborder en détails l'architecture du système à proposer, nous donnerons les objectifs d'une méthodologie de conception des systèmes distribués basés sur le paradigme d'acteur.

D'après [Hahn, 1994], celle-ci est caractérisée par la détermination:

- *des **types** d'acteurs ;*
- *de type de **message** pour chaque type d'acteur,*
- *et du **comportement** de chaque type d'acteur à la réception d'un message de la part de ses accointances* [Hahn,1994].

6.5. Fonctionnalités du système

Nous définissons deux niveaux opérationnels du système

- ***Le système en construction***

Ce niveau de traitement concerne l'étape de configuration, d'installation et de maintenance du système.

- ***Le système en opération***

Dans lequel la question sera traitée et évaluée par rapport à la base de données.

Le schéma suivant illustre ces deux niveaux de traitement.

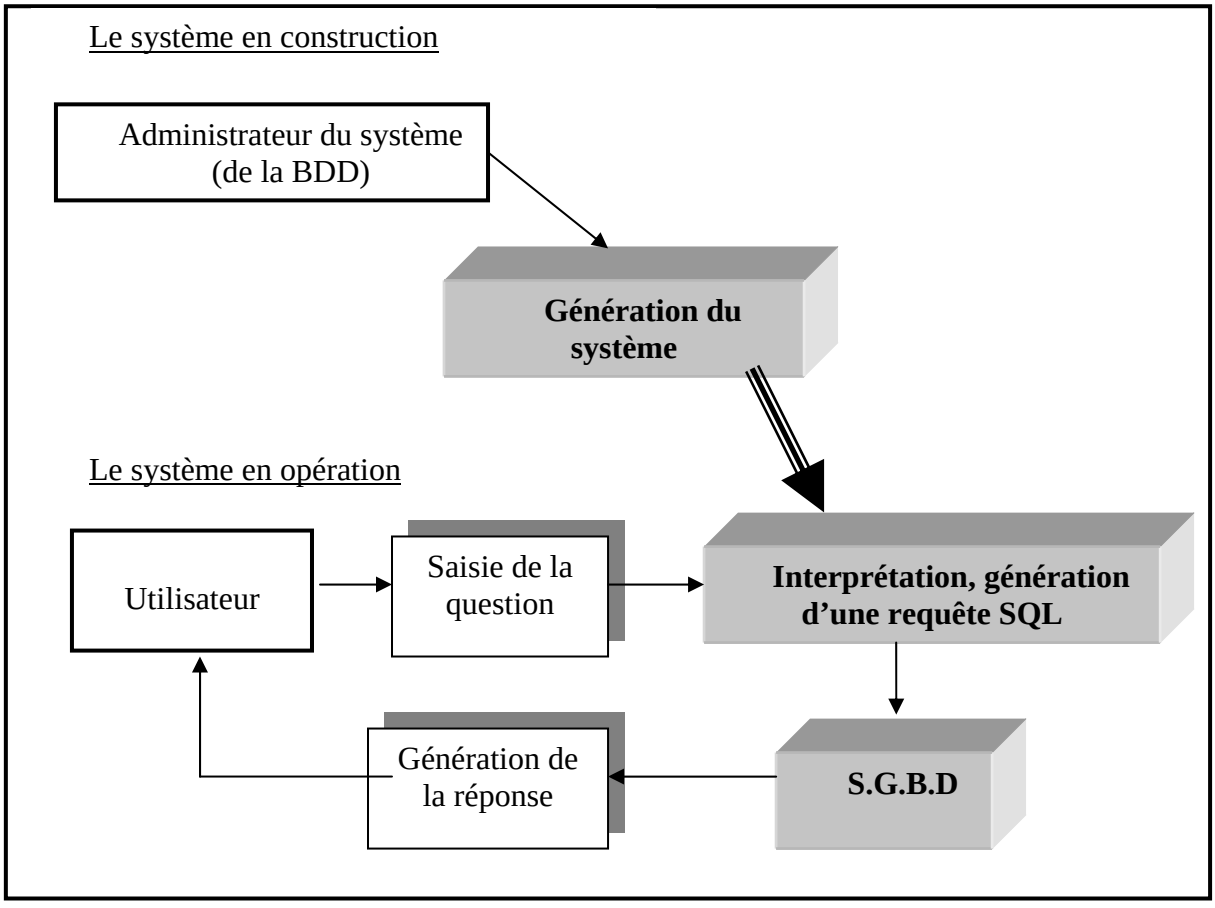

Figure 6-4: « Niveaux du système proposé »

Selon ces niveaux, nous définissons les objectifs suivants :

Au niveau de construction du système

Le système doit être:

- *Indépendant de toute connaissance particulière :* c'est-à-dire que son installation ne doit pas nécessiter l'intervention d'un expert en linguistique, ou en intelligence artificielle ;
- *Portable:* possibilité de reconfigurer le système avec d'autres BDD ;
- *Evolutionnel (par rapport à la BDD):* possibilité d'ajouter des tables, des attributs dans la BDD;
- *Evolutionnel (par rapport au domaine de connaissance -lexique-);* possibilité d'augmentation de lexique;
- *Maintenable:* offrir la possibilité de consulter et d'analyser les requêtes échouées afin d'augmenter le domaine de connaissance;

À ce niveau l'administrateur :

- Installe le dictionnaire grammatical ;
- Installe le dictionnaire des synonymes ;
- Construit le dictionnaire de données (relatif à la base de données) ;

6.5.1. Installation du dictionnaire grammaticale

Le lexique contient le vocabulaire et permet d'introduire les catégories auxquelles un mot appartient. On distingue :

- ***Les catégories lexicales :*** ce sont les mots simples, les mots composés, les expressions,…
- ***Les catégories syntaxiques :*** ce sont les fonctions que peuvent prendre les catégories lexicales : verbe, nom, article, adjectif, . . .

Quelle représentation linguistique ?

Nous utiliserons la représentation en ***structures de traits*** [Bouillon, 1998].

Définition

Une structure de trait [Bouillon, 1998] décrit ou représente un objet par l'énumération de ses caractéristiques significatives. Chaque caractéristique est étiquetée (le trait) et la valeur correspondante à l'objet décrit lui est associée. La ST se présente comme un ensemble de paires attribut-valeur.

Exemple

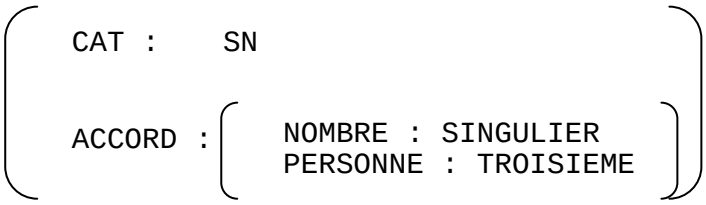

Figure 6-5 : « Exemple d'une structure de traits »

CAT et ACCORD sont les deux attributs de la ST dont les valeurs sont successivement SN et la ST

```
NOMBRE : SINGULIER
PERSONNE : TROISIEME
```

Arguments de choix

Nous justifions le choix de la représentation des informations linguistiques en ST par les deux points suivants [Bouillon, 1998] :

(i). La ST permet de regrouper et de mettre sur le même plan, dans une structure de données globale des informations de différentes natures qu'elles soient morphologiques, syntaxiques, sémantiques ;

(ii). Elle permet d'affiner les contraintes au niveau du lexique et de simplifier ainsi les règles de grammaire ;

Le premier point nous permettra de réduire les ambiguïtés lexicales. Le second, diminuera considérablement la complexité de l'analyse syntaxique.

La couverture lexicale

Le dictionnaire doit reconnaître :

- Les noms simples ;
- Les pronoms relatifs ;
- Les verbes ;
- Les prépositions (à, de, avec) ;
- Les adjectifs ;
- Les déterminants.

6.5.2. Construction du dictionnaire des synonymes

Ce dictionnaire permet d'établir le lien entre le sens de la question et le sens de la base de données, il contribue à la génération des requêtes SQL.

6.5.3. Construction du dictionnaire de données

Deux types d'informations sont nécessaires pour l'interrogation d'une BDD [Bouchou et Maurel, 1999] :

- Les informations sur les objets que manipule la BDD ;
- La nature de lien entre ces objets (les dépendances fonctionnelles) ;

Ces informations doivent être préalablement définies par l'administrateur (l'installateur) qui donne pour chaque table et attribut son nom interne ainsi qu'un nom de la langue courante pour la désigner.

Cette phase déclenche la construction automatique du dictionnaire sémantique de la BDD à partir du dictionnaire des synonymes.

6.6. Modélisation du comportement général du système en opération

Nous allons maintenant modéliser notre système afin de mieux expliciter ses composants et les fonctions que ceux-ci doivent assurer.

Pour se faire nous préférons se baser sur un outil de modélisation normalisé. Nous choisirons pour cela la modélisation UML (Unified Modeling Language) ([Naiburg et Maksimchuk, 2002], [Booch et al., 2003]).

Nous utiliserons le diagramme d'activité pour modéliser le comportement global du système.

Nous remarquons d'après ce diagramme *(voir figure 6-6)* que le système se décompose en trois modules, qui sont :

Le traitement linguistique de la question ;

La génération d'une requête SQL ;

La génération de la réponse en LN.

Le diagramme d'activité

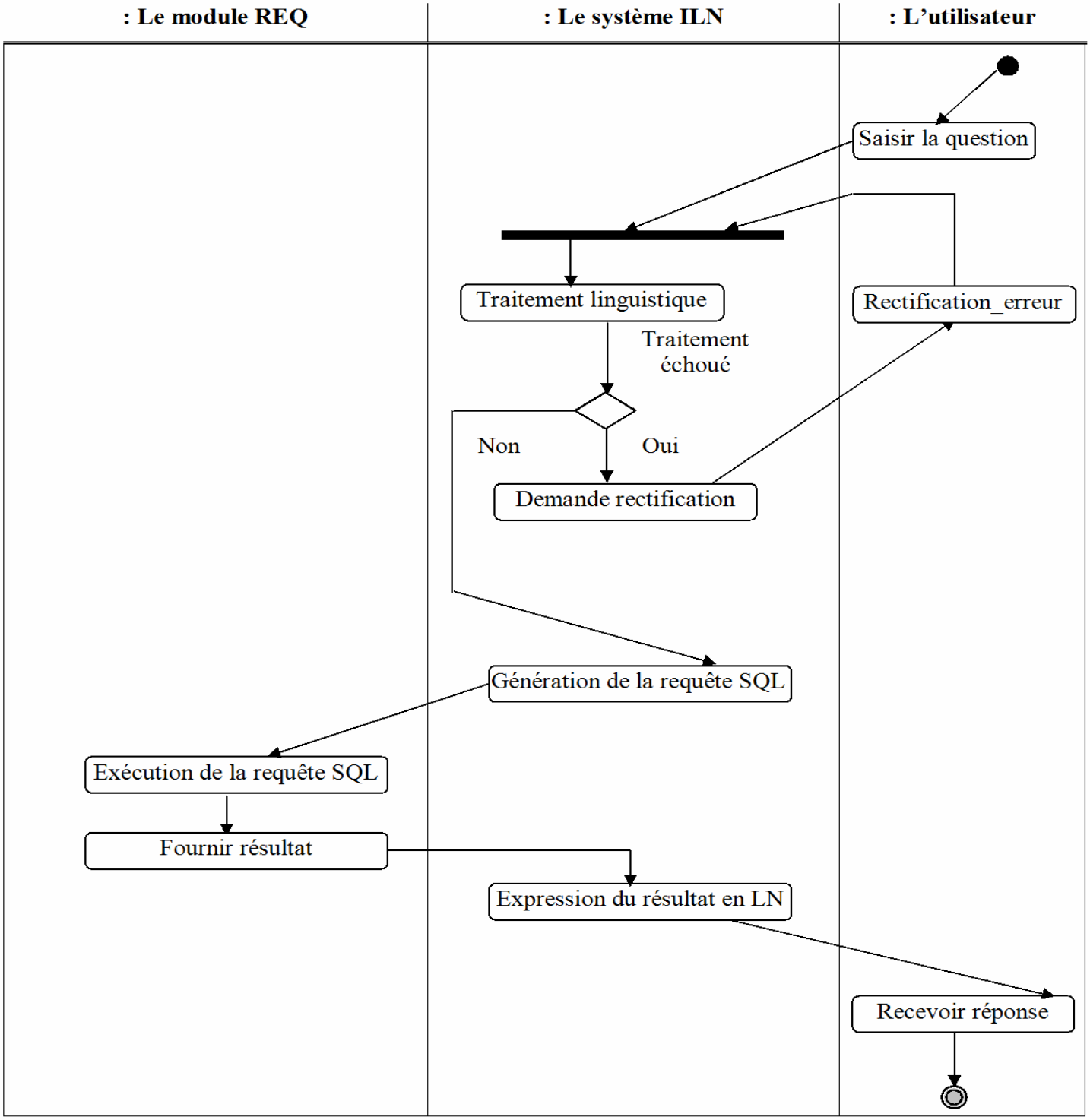

Figure 6-6 : «Diagramme d'activité pour traitement requête »

6.7. La définition des acteurs

Notre système est composé de trois groupes d'acteurs, à savoir les acteurs ***linguistiques***, les ***résolveurs*** et ***les gestionnaires de dialogues.***

Pour la description des types d'acteurs nous utiliserons le diagramme *généralisation d'acteurs* (de l'UML).

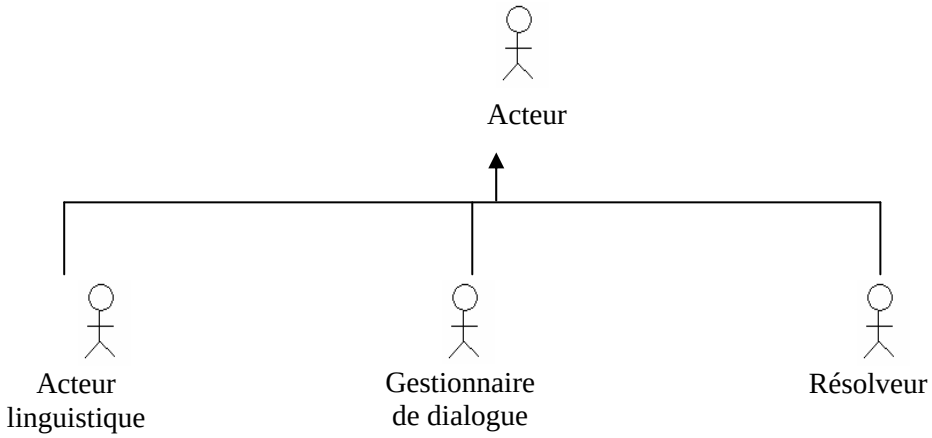

Figure 6-7 : « Généralisation d'acteurs »

Les acteurs linguistique : ces acteurs contribuent à la définition et la validation de la structure syntaxique de la question en entrée.

Les gestionnaires de dialogue : leur but est de communiquer à l'utilisateur la réponse à sa question.

Les résolveurs : leur rôle est de traiter certaines formes particulières de la phrase (les ellipses, les anaphores, les conjonctions, les négations et les quantifications.

6.8. Modélisation du comportement de chaque module du système

La description générale du système a été décrite dans le paragraphe 6.6 du présent chapitre, nous allons, dans ce paragraphe, exposer le comportement général de ses modules.

6.8.1. Le Traitement linguistique

Dans le domaine de l'interrogation de bases de données en langue naturelle, l'expression "langue naturelle" signifie en effet que l'utilisateur d'une base de données peut formuler sa requête en français ou en anglais, par exemple, sans devoir utiliser de syntaxe ou de symboles particuliers. De tels systèmes impliquent une analyse complète de la phrase pour permettre au système de comprendre la requête de l'utilisateur et d'y accéder. Nous appelons cette analyse *traitement linguistique*. Le diagramme d'activité ci-dessous illustre le comportement de ce module.

Le diagramme d'activité

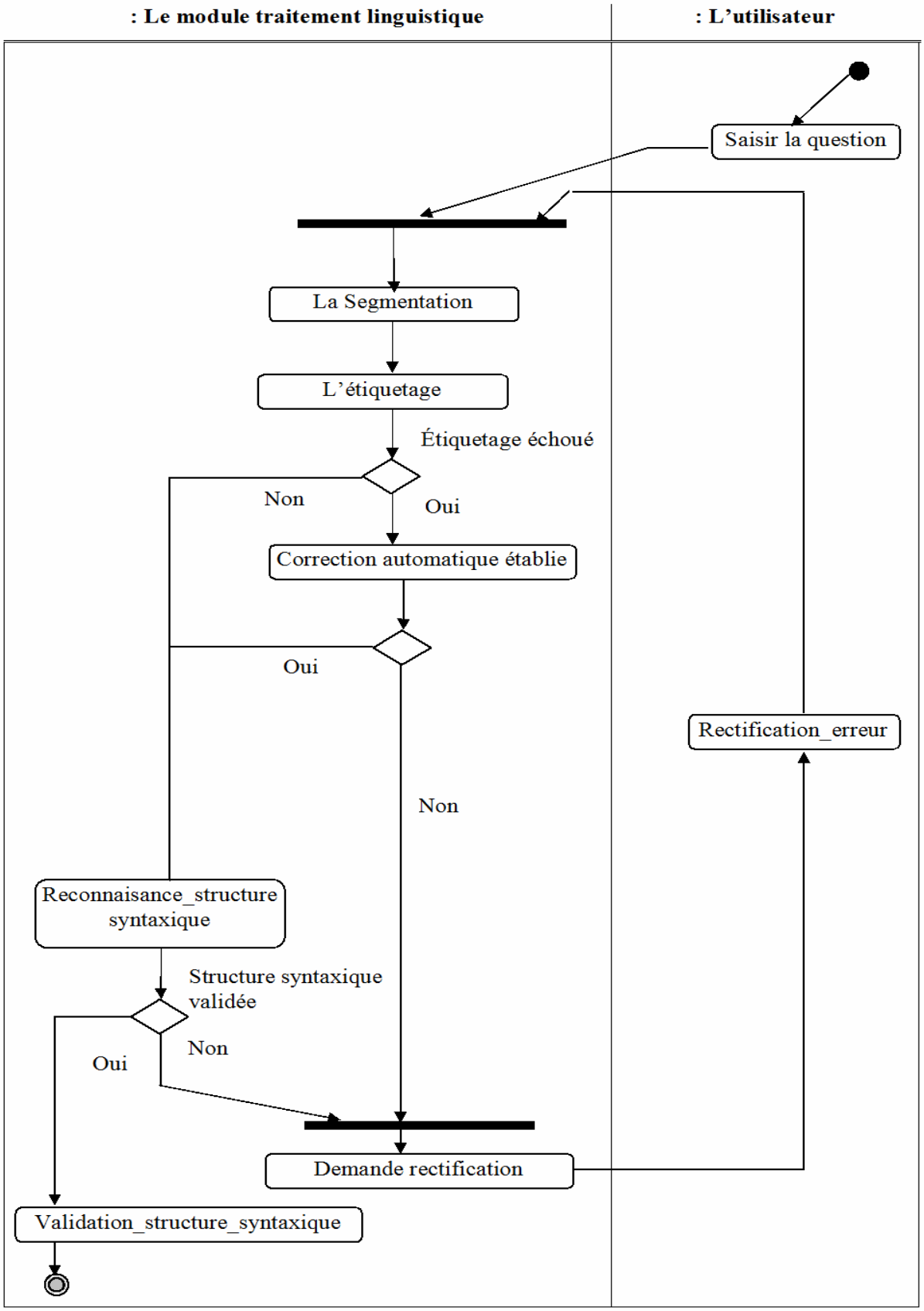

Figure 6-8 : «Diagramme d'activité pour traitement linguistique de la requête »

Description narrative

La segmentation : la segmentation consiste au découpage de texte en mots.

L'étiquetage : c'est la localisation des mots dans le lexique (le dictionnaire grammaticale) et la création des *mot_acteurs*.

L'identification des erreurs

Les erreurs sont une conséquence quasi inévitable de la production du langage. Les erreurs peuvent avoir deux conséquences dans le dialogue entre l'homme et la machine : le système interprète la requête de manière erronée et fournit donc une réponse incorrecte ; ou alors, il ne reconnaît pas la requête et est incapable de fournir une réponse.

L'objectif primordial d'un traitement des erreurs dans un système d'interrogation de base de données est de pouvoir interpréter la phrase malgré les erreurs, et non de corriger celles-ci. Deux stratégies sont possibles. La première consiste à analyser un texte au moyen de règles strictes qui rejettent toutes les phrases ne correspondant pas aux règles. La seconde est celle d'une analyse dite tolérante ([Fouquere, 1986], [Fouqueré,1988]). Lorsqu'une requête erronée est formulée, un analyseur contrôlant fortement le texte (et donc, ne pouvant reconnaître que des phrases correctes) rejettera la plupart des erreurs orthographiques ou syntaxiques et échouera dans l'interprétation de la phrase. Par contre, une analyse tolérante qui assouplit les contraintes d'acceptabilité, telle que la propose ([Fouquere, 1986], [Fouqueré,1988]) peut s'effectuer malgré la présence d'erreurs dans le texte.

Pour l'interface en langue naturelle des systèmes d'interrogation de bases de données, le but à atteindre est que le système comprenne toute requête, qu'elle soit bien formulée ou non. On ne s'attardera donc pas sur les erreurs, mais on cherchera à comprendre le texte au-delà de celles-ci.

La reconnaissance de la structure syntaxique

Le but est de détecter si la question appartient ou non au langage et respecte donc les règles de grammaire de la langue.

L'architecture générale en terme d'acteurs du module traitement linguistique

Le module 'traitement linguistique' est composé de ssix acteurs, à savoir l'acteur ***'Segmenteur'***, l'acteur ***'Lexical'***, l'acteur ***'Syntaxe'***, l'acteur '**acteur_phrase',** l'acteur '**Ellipse'** et l'acteur **'Anaphore'** *(voir figure 6-9).*

L'acteur 'Segmenteur': cet acteur a pour objectif de décomposer la question en mots.

L'acteur 'Lexical' : cet acteur vérifie l'appartenance au langage de chaque mot, obtenu à partir de l'acteur 'Segmenteur'. Il a pour objet de déterminer les caractéristiques morpho-syntaxiques de chaque mot.

L'acteur 'Syntaxe': cet acteur détermine si une question (ou une succession de mots) appartient ou non au langage et respecte donc les règles de grammaire de la langue.

L'acteur 'acteur_phrase' : il reçoit de l'acteur **'Syntaxe'** l'état de validité de la phrase et détermine sa forme (anaphorique, elliptique, conjonctive, négative, quantificative).

L'acteur '**Ellipse' :** cet acteur a pour but de reconstruire les différents types d'ellipses.

L'acteur 'Anaphore' : le but de cet acteur est de retrouver l'antécédent d'une forme anaphorique. Il établit aussi le lien entre la forme anaphorique et son antécédent.

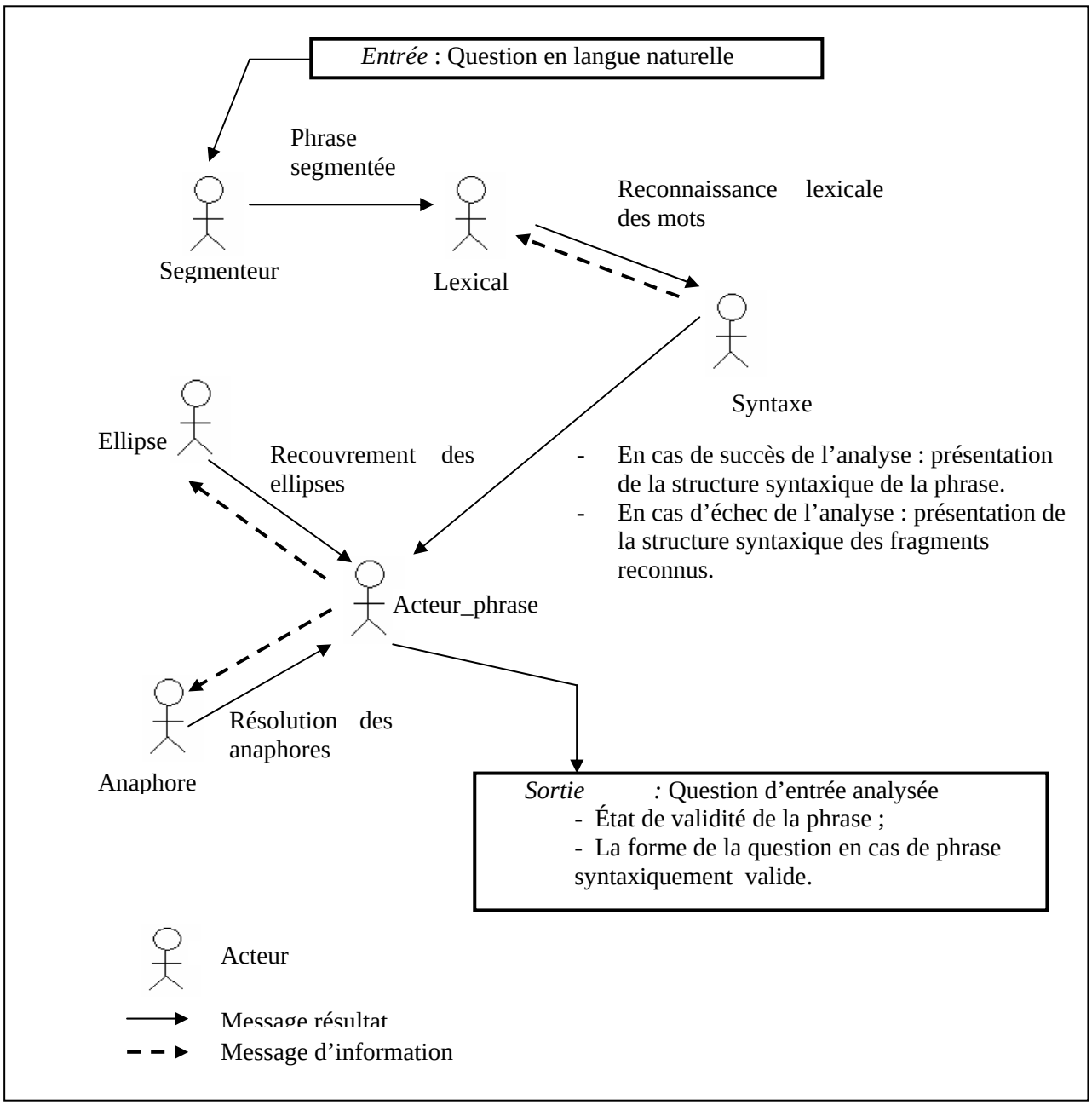

***Figure 6-9 : «Architecture générale en terme d'acteurs du module** 'traitement linguistique' »*

Les différents acteurs du module *'traitement linguistique'* communiquent entre eux par un échange dynamique de messages. Le premier acteur qui se lance, suite à la réception d'une question, est l'acteur **de segmentation** ***(Segmenteur).*** Suite à l'identification du premier mot de cette question, un message est envoyé à l'acteur ***Lexical***. Ce dernier vérifie l'appartenance du mot dans le langage (le dictionnaire) et détermine les caractéristiques morphologiques du mot reçu. Signalons qu'un mot donné peut générer plus qu'une seule liste de caractéristiques morphologiques possibles.

L'acteur ***Syntaxe*** sera lancé dès la réception des caractéristiques morphologiques du premier mot. A ce moment, cet acteur commence par générer puis sélectionner la liste des règles de la grammaire validant ce premier mot trouvé. Cette liste de règles sera raffinée au fur et à mesure de l'évolution de l'analyse. L'acteur ***Syntaxe*** peut orienter l'acteur ***Lexical*** dans le choix des caractéristiques morphologiques des mots suivants. Ceci se fera par envoie de messages d'information présentant les caractéristiques morphologiques des mots attendus. L'analyse s'achève par la reconnaissance d'une structure syntaxique valide, ou bien se limite à la présentation des fragments syntaxiques reconnus. Cette structure syntaxique sera envoyée à l'acteur ***Acteur_phrase*** qui détecte la présence de certaines formes telles que les ellipses, les anaphores, les négations, etc., et selon le cas communique avec l'acteur ***ellipse*** ou l'acteur ***anaphore*** dont le rôle est de recouvrir les structures elliptiques ou anaphoriques.

6.8.2. La génération de requête SQL

Le but de ce module est de générer une requête formelle (type SQL) à partir d'une question issue du module traitement linguistique. À ce niveau,

Description de notre approche

C'est un fait établi en IHM que tout utilisateur se forge un modèle mental de l'application informatique qu'il utilise. Ainsi l'utilisateur de la base a son propre modèle du système d'information qu'elle représente lorsqu'il l'interroge (plus ou moins clair, complet, correct vis-à-vis de l'implantation effective de la base) [Bouchou et Maurel, 1999]. Cette hypothèse nous a donné l'idée que le BDD et la question peuvent partager une même représentation conceptuelle.

Dans le deuxième chapitre nous avons présenté les différentes approches permettant le passage d'une question en langue naturelle vers une requête formelle.

Nous optons pour une représentation conceptuelle de la question projetée sur celle de la BDD.

Cela se justifie par les points suivants :

(i). Le passage par un langage intermédiaire nous semble excessivement cher et très délicat : à chaque installation du système sur une nouvelle BDD, un grand effort doit être donné à l'alimentation manuelle du module sémantique utilisé pour la production de la requête intermédiaire.

(ii). Une représentation conceptuelle peut supporter des inférences afin d'obtenir des relations qui ne sont pas explicitement mentionnées dans la question.

(iii). Le passage par une représentation conceptuelle augmente la portabilité du système et facilite l'interface pour d'autres langues naturelle (du fait que la représentation conceptuelle est indépendante de la langue).

Ce module se déroule en cinq étapes :

La première étape. Epuration de la question ;

But : Eliminer les mots inutiles.

Exemple : je veux la liste des employés : Ici nous savons que le verbe *'vouloir'* signifie que son objet *'employés'* doit être sélectionné, le verbe lui-même ainsi que son sujet *'je'* seront éliminés.

La deuxième étape : Analyse sémantique ou la reconnaissance des mots pertinents à la BDD.

Ce module est chargé d'identifier les mots de la phrase et les correspondre à leur référence (attribut, tables, valeurs) de la BDD. A partir du dictionnaire des synonymes, il associe à chaque mot pertinent de la phrase sa référence sémantique.

La troisième étape : Construction du schéma conceptuel correspondant à la question.

La quatrième étape : Projection du schéma conceptuel correspondant à la question sur celui de la BDD.

La cinquième étape : Génération de la requête SQL à partir d'une représentation conceptuelle.

6.8.3. La Génération de réponse

Dans un système d'interrogation de BDD en langue naturelle, la génération de réponse est aussi importante que la formulation et l'interprétation de la question. Certains ILN pour des BDD relationnelles retournent simplement les tuples retrouvés par la requête de BDD, cette approche n'est pas toujours convenable. Si le système est incapable d'interpréter la question, aucune réponse n'est retournée. Dans ce cas, les raisons d'échec doivent être expliquées à l'utilisateur (ex. mot inconnu, syntaxe complexe, information introuvable dans la BDD, etc.). Le générateur intervient donc au niveau de l'interface personne-machine et communique avec le système (le module REQ) et l'utilisateur.

6.9 Conclusion

Dans ce paragraphe nous avons présenté un système à base d'acteurs pour le traitement d'une requête en langue naturelle. Il s'agit d'une interface entre l'utilisateur et le module REQ du projet Act21 en cours de développement à l'institut d'informatique d'Alger.

En phase opérationnelle du système, l'utilisateur écrit sa question en langage naturel, puis le système lui fournit une réponse issue du module REQ. Avant d'être opérationnel, le système est "installé" sur la base de données cible. Le lien entre le sens de la question et le sens des données stockées est établi au cours de cette installation, sous la forme d'un dictionnaire électronique de mots clés. La phase d'interprétation de la question vers SQL est fournie par un ensemble d'acteurs qui contribuent à la reconnaissance de certaines formes à savoir les négations, les quantifications, les anaphores.

Nous avons décomposé le système en trois modules de base :

- Le traitement linguistique de la question,
- La génération d'une requête SQL ;
- La génération de la réponse en langue naturelle.

Chapitre7 Conclusion générale

7.1 Conclusion de l'étude

Grâce aux progrès constants des matériels et des logiciels, il est possible de développer des systèmes informatiques de plus en plus complexes. Les raisons majeures de la complexité d'un système sont le nombre et la variété de ses éléments, les relations et interactions complexes entre ses éléments, et enfin les différents besoins, souvent conflictuels, de la part des utilisateurs**.** Réduire la complexité de ces systèmes a conduit au développement des modèles objets satisfaisant les exigences des nouvelles applications en terme de modélisation.

En plus de cette modélisation, les nouvelles applications nécessitent un temps de calcul très élevé et un espace de stockage énorme, d'où l'avènement du modèle d'acteur.

Les « *acteurs* » permettent une programmation originale en introduisant une nouvelle structure de contrôle : la transmission de message. Chaque objet, i.e. chaque élément du langage, est décrit sous deux aspects : un aspect statique, ensemble d'attributs pouvant prendre une ou plusieurs valeurs, et un aspect dynamique, ensemble de méthodes, comportements qui sont déclenchés à la réception d'un message.

Le but de notre étude est d'exploiter cette puissance qu'offre le modèle acteur pour la mise en place d'une interface permettant la manipulation des BDD en langue naturelle. Cet objectif ne peut être atteint sans une étude théorique préalable dont le but est de définir l'architecture globale du système. C'est ce qui fait l'objet de notre travail. Nous avons commencé par la collecte d'informations concernant les interfaces de BDD en langue naturelle (voir chapitre 2). Depuis les années 60, de nombreux systèmes d'interrogation de BDD en langue naturelle ont été réalisés et ont démontré des caractéristiques impressionnantes et constituent jusqu'aujourd'hui une référence pour toutes les recherches dans ce domaine (tel que LUNAR, BASBALL, TEAM). Actuellement, et avec l'avènement du Web, les interfaces en langues naturelles pour les BDD tiennent une place de premier ordre dans les recherches en traitement de la langue. Les problèmes auxquels doit faire face un système d'interrogation de BDD en LN, rendent complexe l'implémentation d'un tel système.

Cette complexité est due à la diversité des sources de connaissances que manipule le système et la nécessité d'une communication continue et d'un échange d'informations entres ses différents composants.

De telles exigences impliquent le choix d'une approche permettant d'une part la distribution des connaissances et d'autre part d'assurer une communication efficace entre les sources de connaissance. Dans une telle approche, les niveaux de traitement seront vus comme des processus actifs communiquant entre eux, et chaque niveau sera considéré comme une composition d'un ensemble d'objets dynamiques. Le modèle acteur (voir chapitre 3), se présente comme une solution idéale pouvant satisfaire de telles exigences. Par sa définition, un acteur peut être considéré comme un expert vivant en société et communiquant avec d'autres experts pour résoudre des problèmes. Chacun des experts peut lui-même être une société d'experts plus élémentaires. Le modèle est donc distribué (ou réparti). Les connaissances et les comportements sont diffusés parmi les acteurs, qui effectuent des tâches en parallèle, de manière indépendante.

L'exploitation du modèle acteur pour le traitement de la langue (voir chapitre 4) a constitué le sujet de plusieurs projets de recherche dont l'analyseur de Yonezawa, PEP et ParseTalk sont les plus connus.

7.2 Contribution de la présente recherche

Notre travail s'inscrit dan le projet Act21 qui est un SGBD parallèle à base d'acteurs (DB-Act) en cours de développement à l'Institut National d'Informatique d'Alger (I.N.I). Il est formé de quatre modules : STOCK, CAT, EXEC, REQ. Ce dernier offre aux utilisateurs un langage de requêtes déclaratif (type SQL) et prend en charge son exécution en générant les différents acteurs nécessaires à la résolution de la requête. Son rôle est de traduire les requêtes en un programme écrit en PACT (c'est un langage de programmation de type impératif augmenté d'un mécanisme d'envoi de messages inter-acteurs), et appelle l'interpréteur du module EXEC pour contrôler leur exécution.

Notre travail consiste à proposer une interface en langue naturelle entre l'utilisateur et le module REQ (voir chapitre 6). Le système d'interrogation reçoit en entrée une question en langue naturelle puis la traduit en une requête SQL et la transmet au module EXEC.

Avant d'être opérationnel, le système est "installé" sur la base de données cible. Le lien entre le sens de la question et le sens des données stockées est établi au cours de cette installation, sous la forme d'un dictionnaire électronique de mots clés.

Nous avons décomposé le système en trois modules de base :

- Le traitement linguistique de la question ;
- La génération d'une requête SQL ;
- La génération de la réponse en langue naturelle.

Nous avons défini trois groupes d'acteurs :

- *Les acteurs linguistiques* dont le rôle est la définition et la validation de la structure de la phrase. Ils interviennent principalement dans le module traitement linguistique.
- *Les gestionnaires de dialogue* leur but est de communiquer à l'utilisateur la réponse à sa question. Ils interviennent au niveau du deuxième et troisième module.
- *Les résolveurs* qui contribuent au traitement de certaines formes particulières, se sont par exemple les acteurs ellipse, anaphore.

7.3 Opportunités pour d'autres recherches

Comme nous l'avons mentionné, l'objectif est de définir l'architecture d'un système à base d'acteurs pour l'interrogation de BDD en langue naturelle. De ce fait, notre travail ne peut être productif, rentable et évalué, que s'il est suivi par d'autres projets dont le but est la mise en place des acteurs proposés dan la phase de conception.

- Concernant le module traitement linguistiques, les acteurs à mettre en place sont :
  - **L'acteur 'Segmenteur':** dont le rôle est de décomposer la question en mots.
  - **L'acteur 'Lexical' :** vérifie l'appartenance au langage de chaque mot, obtenu à partir de l'acteur 'Segmenteur'. Il a pour objet de déterminer les caractéristiques morpho-syntaxiques de chaque mot.
  - **L'acteur 'Syntaxe':** cet acteur détermine si une question (ou une succession de mots) appartient ou non au langage et respecte donc les règles de grammaire de la langue.
  - **L'acteur 'acteur_phrase' :** il reçoit de l'acteur **'Syntaxe'** l'état de validité de la phrase et détermine sa forme (anaphorique, elliptique, conjonctive, négative, quantificative).
  - **L'acteur** '**Ellipse' :** cet acteur a pour but de reconstruire les différents types d'ellipses.
  - **L'acteur 'Anaphore' :** le but de cet acteur est de retrouver l'antécédent d'une forme anaphorique. Il établit aussi le lien entre la forme anaphorique et son antécédent.

- Etude approfondie du module génération d'une requête SQL ;
- Etude approfondie du module génération de la réponse en langue naturelle.

Bibliographie

[Abeillé et Blache, 1999] ABEILLÉ, A ; BLACHE,P. *Grammaires et analyseurs syntaxiques.* Version v1.0e, 14 juin 1999.

[Adriaens et Hahn, 1994] ADRIAENS,G; HAHN,U. *Parallel natural language processing*. Ablex Publised Corporation. 1994. PP 90

[Agha et Wooyoung,1999] AGHA,G ; WOOYOUNG,K. *Actors : A unifying model for parallel and distributed computing*. Elsevier, Journal of Systems architecture, 1999

[Aloulou,2003] ALOULOU,C ; *Analyse syntaxique de l'Arabe: Le système MASPAR* **.** RÉCITAL 2003, Batz-sur-Mer, 11-14 juin 2003

[Amrouche,2004] AMROUCHE, H. *TRANS_ACT : Un gestionnaire de transactions pour le SGBD parallèle à base d'acteurs ACT21*. Thèse de magister, INI-Oued Smar, Juillet 2004

[Androutsopoulos et al., 1993] ANDROUTSOPOULOS, I; RITCHIE, G.D; TANISCH, P. MASQUE, *An Efficient and Portable Natural Language Query Interface for Relational Databases*. Proceedings of the Sixth International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems', Edinburgh, June 1993.

[Androutsopoulos et al., 1995] ANDROUTSOPOULOS, I; RITCHIE, G.D; TANISCH, P. *Natural Language Interfaces to Databases An Introduction.* Natural Language Engineering, vol 1, 1995, pages 29-81.

[Booch et al., 2003] Booch, G ; , Rumbaugh, J ; Jacobson, I. : "Le guide de l'utilisateur UML", Eyrolles 2003.

[Bouchou et Maurel, 1999] BOUCHOU, B; MAUREL, D. *Interrogation de BDD en LN: une nouvelle approche.* Conférences TALN 1999, Cargés, 12-17 juillet 1999.

[Bouillon, 1998] BOUILLON,P. *Traitement automatique des langues naturelles*. Ed. Duculot, 1998

[Bouzeghoub et al., 2000] BOUZEGHOUB, M ; GARDARIN, G; VALDURIEZ, P. *Les objets*, Ed. Eyrolles, 2000.

[Bozenn et Frederic, 1983] BOZENN, H; FREDERIC, B. *Introducing ASK, a simple knowledgeable system.* In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, 1983, pages 17–24.

[Burros et DeRoeck, 1992] BURROS, A.F; DEROECK, A. *Resolving Anaphora in a Portable Natural Language Front End to Databases*. Department of Computer Science University of Essex. 1992.

[Charniak et al., 1983] CHARNIAK, E; GE, N, HALE, J. *A Statistical Approach to Anaphora Resolution*. Dept. of Computer Science, Brown University, 1998.

[Desarte et Thayse, 2001] DESARTE, P ; THAYSE,A. *Logique pour le traitement de la langue, application à la langue française*. Ed. Hermès Science, 2001

[Devos et Adriaens, 1994] Devos, M; Adriaens, G. *The Parallel Expert Parser*, PP 350-373

[Dominé,1988] DOMINÉ, C.H. *Techniques de l'intelligence artificielle, un guide structuré*, 1988, Ed. Dunod informatique.

[Eiselt et Granger, 1994] EISELT, K.P; GRANGER, R.H. *Process independence and concurrency in a model of sentence understanding*. (Parallel Natural Language processing) PP285.

[Fass,1998] *Le SystemX:*
Natural Language Laboratory at Simon Fraser University
http://fas.sfu.ca/cs/research/groups/NLL/3.html
date de dernière modification 9 Octobre 1998 (date de consultation 4 Décembre 2005)

[Fouquere, 1986] FOUQUERE, C ; *Analyse tolérante de textes en langage naturel*, séminaire GRECO-Communication Parlée, pp.67-74, Toulouse, 1986.

[Fouqueré,1988] FOUQUERÉ, C. *Un modèle pour la correction de phrases : une grammaire à configuration minimale.* In 3ème Colloque International `Cognition et Connaissance', ARC, pages 127-142, Toulouse, France, March 1988.

[Gabrias, 1997] GABRIAS, H. *Encyclopédique du génie logiciel.* Ed. Masson, 1997

[Gilles,2000] GILLES,B. *Langage naturel, Intelligence artificielle*. 2000.

[Grosz, 1983] GROSZ, J. *TEAM: a transportable natural language interface system.* In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, 1983, pages 39–45.

[Grune et Jacobs; 1994] GRUNE, D; JACOBS,C. *Parsing techniques a Practical Guide*", Ellis Horwood Limited, 1994

[Hahn, 1996] HAHN, U. *"concurrent, object-oriented natural language parsing: the ParseTalk model",*
http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol2/pl5/
http://www.isys.dia.fi.upm.es/doctorado0405/UdoHan.pdf

[Hahn,1994] HAHN, U. *An Actor model of distributed natural language parsing*. In Parallel natural language processing, 1994, PP 307-350.

[Hidouci et Zegour, 2001] HIDOUCI, W.K ; ZEGOUR, D.E. *Integration des acteurs de programmation dans les bases de données*.

[Hobbs, 1986] HOBBS, J.R. *Resolving Pronoun References*. In B.J. Grosz, K. Sparck Jones, and B.L.
Webber, editors, Readings in Natural Language Processing, Morgan Kaufmann Publishers, California. 1986,
pages 339–352.

[Kaplan et Davidson, 1983] KAPLAN, S.J; DAVIDSON, J. *Natural Language Access to Data Bases Interpreting Update Requests*. American Journal of Computational Linguistics, April-June 1983? Volume 9, Number2.

[Kaplan et Davidson] KAPLAN, S.J. DAVIDSON, J. *Interpreting natural language database updates.*

[Knowles et Mitrovic, 1999] KNOWLES, S; MITROVIC, T. *A natural language database interface for SQL tutors.* 5th November, 1999.

[Labidi et Lejouad,1993] LABIDI,S ; LEJOUAD,W. *De l'intelligence artificielle distribuée aux systèmes multi-agents.* INRIA rapport de recherche N°2004, Aout 1993.

[Lieberman,1990] LIEBERMAN,H. *Habilitation à diriger des recherches, Mémoire de synthèse.* Octobre 1990

[Lin et Goebel, 1994] LIN, D; GOEBEL, R. *Context-Free Grammar Parsing by Message Passing,*

[Lohuizen, 1998] LOHUIZEN, M.P, 1998. *IMPACT parallel natural language interfaces architectural aspects of natural language processing systems*.

[Mahoudeaux, 2004] MAHOUDEAUX, P.M. *Introduction au Traitement Automatique du Langage Naturel*. Automne 2004

[Manning et Schiitze, 2000] MANNING,D.C; SCHIITZE,H.. *Foundations of Statistical Natural Language Processing.* The MIT Press Cambridge, Second printing with corrections, 2000

[Matthews,1999] MATTHEWS, J. *Conceptual Representation and Scripting.* 1999 **http://www.generation5.org/content/1999/concept.asp**

[Mazlack et Feinauer, 1982] J. MAZLACK, J; A. FEINAUER, R: *Surface analysis of queries directed toward a database*. North Holland Publishing Company 1982.

[Minhwa et Dan, 1994] MINHWA, C; DAN,M. *Applying parallel processing to natural-language processing*. IEEE EXPERT, 1994.

[Moore, 1982] MOORE, R.C. *Natural language access to databases-theoretical/technical issues.* In *Proceedings of the 20th Annual Meeting of the Association f o r Computational Linguistics*, pages 44-45. Association for Computational inguistics, June, 1982.

[Morgan, 1994] Morgan,R. *The LOLITA Natural Language Processing System.* Laboratory for Natural Language Engineering, Department of Computer Science, University of Durham, England http://homepages.inf.ed.ac.uk/wadler/realworld/natlangproc.html

[Naiburg, Maksimchuk, 2002] NAIBURG, E.J ; MAKSIMCHUK, A.R . *Bases de données avec UML*, CampusPress 2002.

[Nijholt,1994] NIJHOLT, *A. Parallel approaches to Context-Free language parsing*, in Parallel natural language processing,
PP 135-167.

[Pappa et al., 2002] PAPPA,A ; BERNARD,G ; OUEKERADI, H. *Détection automatique de frontières des phrases - un système adaptatif multi-langues*. 2002, Laboratoire IA Holistique-Groupe C.S.A.R Université Paris 8.

[Pierrel,1987] PIERREL,J.M ; *Dialogue oral homme-machine.* Ed. Hermès, 1987

[Popescu et al., 2003] POPESCU, A.M; ARMANASU,A; ETZIONI, O; KON, D; YATES, A.: *Modern natural language interfaces to Databases: Composing Statistical Parsing with Semantic Tractability*. 2003

[Popescu et Etzioni, 2004] POPESCU, A.M; ETZIONI,O. *Towards a theory of natural language interfaces to Databases*.

[Riesbeck, 1986] Riesbeck, C.K. *From Conceptual Analyzer to Direct Memory Access Parsing : An Overview*. In Advances in cognitive science 1r edition. Sharkey, N.E Ellis Horwood. 1986.
www.cogsci.northwestern.edu/courses/cg207/Readings/Riesbeck_From_CA_to_DMAP.pdf

[Slonneger et Kurtz; 1995] SLONNEGER,K; KURTZ,B.L *Formal Syntax and Semantics of Programming Languages, a laboratory based approach*. 1995.Addison-Wesley Publishing Company.

[Small,1980] SMALL,S.L. *Word expert parsing* a Theory of Distributed Word-Based Natural Language Understanding
Department of Computer Science, University of Maryland
College Park, Maryland 20742. 1980
http://acl.ldc.upenn.edu/P/P79/P79-1003.pdf

[Sowa, 1976] SOWA, J.F. *Conceptual Graphs for a Data Base Interface*.

[Templeton et Burger, 1983] TEMPLETON, M; BURGER, J. *Problems in Natural Language Interface to DBMS with Examples from EUFID*. In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, 1983, pages 3–16.

[Thibeault,2004] THIBEAULT, M. *La catégorisation grammaticale automatique: adaptation du catégoriseur de Brill au français et modification de l'approche*, 2004-11

[Thompson,1994] THOMPSON,S.H. *Parallel parsers for context-free-grammars, Two actual implementations compared*" in Parallel natural language processing, 1994 PP 168-187.

[Trigano, 2005] TRIGANO, P. *SIC : Systèmes Interactifs de Connaissances. Interrogation en Langage Libre de Bases de Données et de Connaissances (FQUEL).* Université de technologie Compiègne, Paris. 2005
http://www.hds.utc.fr/~ptrigano/fquel.html

[Veronis, 2001] VERONIS, J. I*nformatique et linguistique1*. 2001

[Warren et Fernando, 1982] WARREN, H.D; FERNANDO, C. N. *An Efficient Easily Adaptable System for Interpreting Natural Language Queries*, American Journal of Computational Linguistics, 1982, Volume 8, Number 3-4, July-December.

[Winograd,1983] WINOGRAD,T. *Language as a cognitive process.* V.1 – SYNTAX- , 1983, Addison-Wesley Student Edition,

[Yonezawa et Ohsawa, 1994] YONEZAWA, A.; OHSAWA I., *Object-Oriented Parallel Parsing for Context-Free Grammars*, in Parallel Natural Language Processing, PP 188-236

[Zwigenbaum et al., 2003] ZWEIGENBAUM, P ; POIBEAU, T ; NAZARENKO, A. *Traitement automatique des langues pour les systèmes de question/réponse*. Document de travail élaboré dans le cadre de l'action spécifique RIP-WEB, 2003

ANNEXE Le Traitement Automatique des Langues Naturelles "TALN"

Définition

Le **T**raitement **A**utomatique des **L**angues **N**aturelles ou **TALN**, est la discipline qui étudie les techniques permettant à une machine d'analyser, de comprendre et/ou de produire des messages en langage humain, qu'il s'agisse de textes, d'énoncés en situation de dialogue ou encore d'autres types de message**.**

Applications du Traitement Automatique des Langues

On distingue deux types principaux d'application du TALN : les applications relatives à la réalisation d'**interfaces** et celles faisant appel à des notions de **compréhension**.

Interfaces (dialogue) en langue naturelle

Avec les systèmes interfaces, on cherche à communiquer avec la machine pour entrer les informations dans un programme existant. Il s'agit alors d'ajouter une couche de programme supplémentaire à un système existant, pour, par exemple, transformer un texte en une série de commandes et l'exécuter.

C'est typiquement le cas pour la communication avec les bases de données, les systèmes experts ou les robots. On s'en sert pour les opérations de mise à jour ou d'interrogation.
La notion de dialogue est équivalente, si ce n'est qu'elle implique la mise en forme élaborée de la réponse du système à une requête. On la trouve dans des applications telles que :

- Gestion des questions-réponses sur une base de données.
- Formation à distance, tutorage.
- Contrôle et réglage de machines industrielles. [Mahoudeaux, 2004]

Compréhension des textes

Ces applications sont plus difficiles à mettre en œuvre. Elles supposent une véritable compréhension du texte, par exemple :

- La recherche des documents pertinents correspondant à un sujet donné.
- L'extraction d'informations textuelles et le résumé automatique de texte.
- La traduction automatique.
- L'indexation automatique de grandes bases textuelles.
- La construction automatique de bases de données et bases de connaissance (fouille de données ou data mining). [Mahoudeaux, 2004]

Domaines de recherches concernés par l'étude de la langue

Il n'est pas envisageable de développer un projet en TALN si l'on est totalement ignorant des nombreuses notions impliquées. Nous donnons ci-dessous une liste de celles-ci, avec une brève description de leur implication dans l'étude du langage.

- **Linguistique** : élaboration des règles qui fournissent une description structurelle ou fonctionnelle du langage de façon à identifier ses caractéristiques. Les travaux sur la linguistique permettent de mettre en lumière le rôle joué par les catégories linguistiques telles sujet ou groupe sujet, verbe ou groupe verbal, complément ou groupe complément, dans la structure de la phrase.
- **Logique** : formalisme mathématique qui peut être appliqué à la construction de modèles de représentation des catégories et de leur regroupement en structures. C'est évidement un domaine privilégié pour développer des systèmes informatiques de résolution de problèmes à base de règles. C'est le mode de représentation le plus approprié pour la modélisation des fonctions d'analyse et de synthèse syntaxique.
- **Psycholinguistique** : étude des représentations mentales qui permettent d'engendrer et de comprendre des énoncés verbaux. L'étude de la façon dont un être humain comprend un texte permet de proposer des modèles que l'on peut implémenter sur machines.
- **Neurosciences** : modèles connectionnistes des fonctions d'intelligence. Elles s'intéressent à la biologie humaine qui explique comment les cellules sont connectées dans le cerveau et leur fonction dans la compréhension du langage naturel. Leur implication en informatique et notamment en intelligence artificielle prend une place de plus en plus importante.
- **Intelligence artificielle** : simulation de comportements intelligents par ordinateur. Son objet est de développer des programmes qui simulent les comportements humains à l'aide d'ordinateurs. [Mahoudeaux, 2004]

Niveaux de traitement

Nous introduisons dans ce paragraphe les différents niveaux de traitement nécessaires pour parvenir à une compréhension complète d'un énoncé en langage naturel. Ces niveaux correspondent aux modules qu'il faudrait développer et faire coopérer dans le cadre d'une application complète de traitement de la langue.

Première étape : L'analyse morpho-lexicale

L'objectif est de retrouver le mot étudié tel qu'il apparaît dans le texte (on parle de la forme) et de récupérer toute l'information utile à son sujet. Par exemple s'il s'agit de la forme

« chevaux », cette étape permettra de lui associer les propriétés suivantes : nom commun masculin pluriel, dérivé du lemme « cheval », sens possible : animal équidé.

- L'analyse morpho-lexicale se décompose en trois étapes :

Etape 1 : La segmentation : Dont le but est de découper le texte en phrases puis en mots distincts. On parle de la détection automatique des phrases. Le problème posé dans cette étape est celui de l'ambiguïté de certains signes de ponctuation. Par exemple un point peut être utilisé pour déclarer la fin d'une phrase mais aussi pour exprimer une abréviation ou un acronyme, ou même un nombre décimal par exemple : 3.14 (écriture anglo-saxonne) [Pappa et al., 2002].

Etape 2 : La lemmatisation : Qui s'attache à déterminer la forme canonique des mots isolés dans l'étape précédente.

Etape 3 : L'étiquetage : Dont l'objectif est d'identifier la bonne catégorie morpho-syntaxique (verbe, nom,…) des mots selon le contexte. Cette dernière étape est très importante, car elle conditionne l'interprétation du texte. Elle est également très délicate. En effet, il est parfois difficile d'attribuer une catégorie à un mot.

Dans [Mahoudeaux, 2004] l'auteur a résumé les difficultés de cette étape dans les points suivants ; (il est à noter que ces difficultés différent selon la langue):

– Les informations linguistiques sur les mots ne sont pas déductibles de leur forme (en français, les mots terminés par -s sont souvent au pluriel, comme *raisons*, mais certains peuvent être au singulier comme *stimulus* et *prends* ; pour d'autres, la notion de pluriel n'est pas pertinente; et certains mots qui ne sont pas terminés par -s sont pourtant au pluriel, comme *réseaux*) ;

– Certaines formes peuvent recevoir plusieurs étiquettes comme *ferme* qui est à la fois un nom, un adjectif et un verbe ;

– Certains systèmes d'écriture n'ont pas de séparateur, ce qui pose le problème de la séparation des mots. Ce problème se pose également en français et en anglais, mais pour les mots composés.

En effet, ceux-ci ne sont pas toujours graphiquement marqués par des séparateurs, comme *fibre optique*;

– Certains systèmes d'écriture ne sont pas normalisés, c'est à dire qu'il existe plusieurs façons correctes d'orthographier un texte donné, en raisons de lettres facultatives, de séparateurs facultatifs, ou parce qu'il permette de passer d'un alphabet à un autre (japonais).

Deuxième étape : L'analyse syntaxique

La syntaxe est définie comme l'ensemble des règles qui président à l'ordre des mots et à la construction des phrases [Pierrel,1987]**.** Le rôle de cette étape est de lever les ambiguïtés éventuelles de l'étape précédente en établissant un schéma des relations entre les mots (grammaires de la phrase). Il s'agit de lever les doutes grâce au contexte syntaxique de la phrase.

Exemple : la forme « avions » peut, prise isolément, désigner :

1. Soit un nom commun masculin pluriel dans le champ des moyens de locomotion ;
2. Soit la forme conjuguée du verbe avoir, à la première personne du pluriel, au présent de l'indicatif.

Mais la présence de la forme « des » juste avant « avions » permet de lever immédiatement l'ambiguïté**.** Cette étape aura deux fonctions définies par [Bouillon, 1998]

Une fonction normative: fixer les règles selon lesquelles une phrase est déclarée grammaticale, par opposition aux phrases agrammaticales qui transgresses cette nature;

Une fonction représentative: qui consiste à associer aux phrases des représentations syntaxiques qui explicitent leur structure.

Cependant, la syntaxe seule n'est pas toujours suffisante. Certaines phrases ne répondant pas à une structure syntaxique classique sont porteuses de sens, par exemple une annonce telle que : *Vds lect. CD, tel 024 80 44 23* [Pappa et al., 2002]. A l'inverse, certaines phrases peuvent avoir une bonne syntaxe, mais n'ont pas de sens :

– une pomme mange Rachid ;

– la vertu est triangulaire, les géants sont petits

L'analyse syntaxique permet de lever certaines ambiguïtés, telles que dans les phrases suivantes : *– le médecin ferme le porte doucement*

– le médecin ferme la fenêtre doucement

mais pas toutes : *le médecin ferme la porte doucement*

(on ne sait pas si le médecin est ferme et porte un objet du genre féminin ou si le médecin accomplit l'action de fermer la porte).

De même en utilisant les accords genre nombre, on pourra distinguer clairement le sujet dans les phrases suivantes : *le cavalier de la reine qui est menacée ;*

le cavalier de la reine qui est menacé

mais pas dans celle-ci. . .*le cavalier du roi qui est menacé.* [Mahoudeaux, 2004]

Traditionnellement, le résultat de l'analyse syntaxique est représenté sous la forme **d'un arbre syntaxique**, ce qui permet d'identifier simultanément les frontières de constituants, ainsi que les relations de dominance qu'ils entretiennent [Veronis, 2001]. On obtiendrait par exemple, pour la phrase

«*Le chien mange le chat* », l'arbre syntaxique

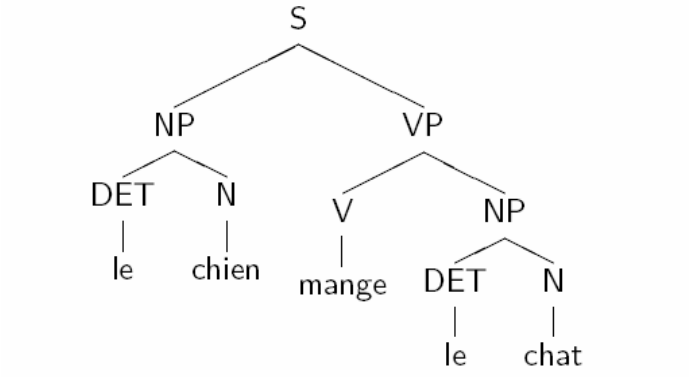

Figure 1: « Arbre syntaxique de la phrase 'le chien mange le chat' »

Troisième étape : L'analyse sémantique

S'intéresse au sens des phrases considérées individuellement. Elle consiste à construire une représentation conceptuelle du sens du texte, idéalement même indépendante de la langue.

Cette étape permet, également, de rejeter une phrase de structure syntaxique correcte, mais qui n'a pas de sens. Par exemple, *le courrier lit mon père*.

Elle s'appuie sur la notion de catégories de sens. On les définit au niveau lexical. Par exemple le mot *orage* peut être associé aux mots pluie ou grêle dans des expressions telle que : *un orage de grêle, un orage de pluie*, mais pas au mot *soleil* dans un *orage de soleil*. De même on définit les catégories de groupe de sens. Par exemple le groupe *récipient {casserole, seau, baquet}* permet de définir le fait qu'on peut verser de l'eau dans récipient.

Un tel groupe permet de reconnaître certaines connotations

– *Rachid aime le chocolat : gourmandise*

– *Rachid aime l'argent : avarice*

Quatrième étape : L'analyse pragmatique/contextuelle

La pragmatique est définie comme l'étude des aspects du langage qui font référence aux relations entre locuteur et interlocuteur, d'une part, entre interlocuteurs et situation concrète, d'autre part. Elle recouvre l'ensemble des relations entre langage et contexte d'énonciation.

Dans le cadre de la compréhension du langage naturel, la pragmatique permet de traiter et d'interpréter:

- les déictiques ou ensemble des mots dont la référence fait partie de la situation de communication, tels le locuteur (je, me, moi,…) ou les lieux (ici, là,…..);
- les références anaphoriques, le plus souvent pronominales, qui permettent de reprendre un terme ou une phrase antérieur (*il l'*avait souvent entendu dire *cela*);
- les ellipses : *Rachid aime le chocolat et Zohra la vanille*. [Pierrel,1987]

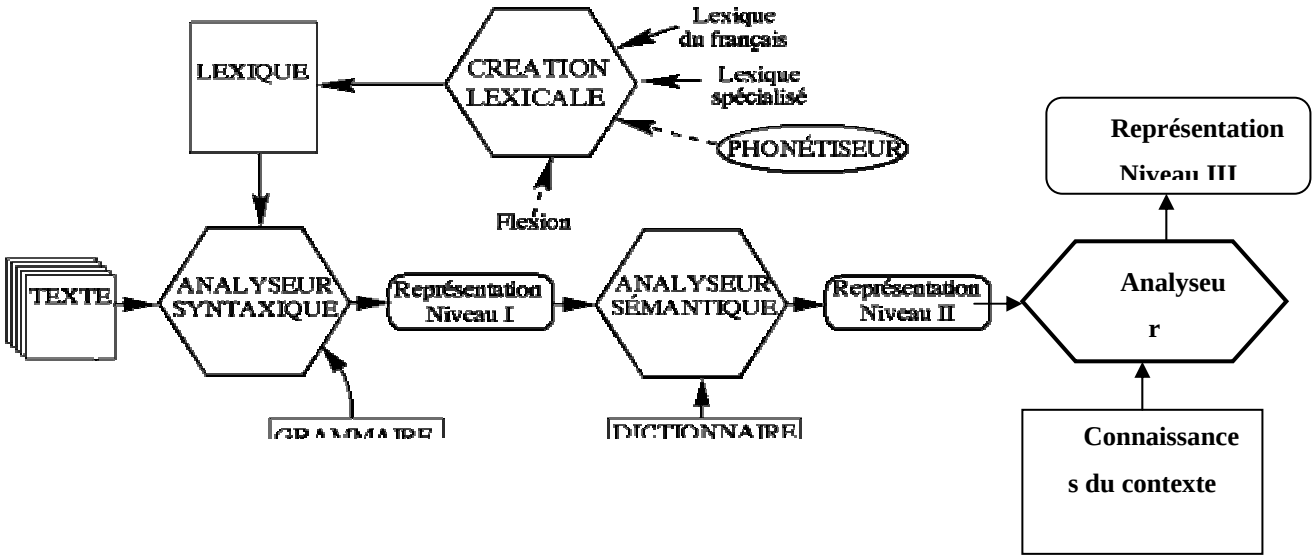

Figure 2 : Illustration d'une architecture simplifiée de système de TAL

Exemple : progression d'une analyse

Soit la phrase tirée de [Mahoudeaux, 2004]: *le secrétaire vole des livres.* Si l'on considère les différents sens que l'on peut donner à chacun des mots qui la constituent, c'est à dire, si on combine tous les homonymes on obtient 108 interprétations possibles.

Mot	**Sens**	**Nbr Homonymes**	**Nbr Combinaisons**
Le	Article, pronom	2	2
Secrétaire	Etre humain, meuble, oiseau (serpentaire)	3	6
Voler	Dérober, se déplacer dans les airs	2	12
Des	Article indéfini, article défini contracté, article partitif	3	36
Livres	Ouvrages, monnaie, unité de poids.	3	108

1. Analyse morphologique

– le : article ou pronom masculin singulier

– secrétaire : nom masculin singulier, nom féminin singulier

– vole : voler (transitif) ou voler (intransitif) 1ère ou 3ème personne indicatif présent

– des : article défini contracté, article indéfini

– livres : nom féminin pluriel, nom masculin pluriel

2. Analyse syntaxique

– le : article

– secrétaire : nom masculin

– vole : verbe transitif, 3ème personne indicatif présent

3. Analyse sémantique

voler : sujet animé et objet (COD) concret ⇒ secrétaire_meuble éliminé

4. Il reste 4 sens possibles

– l'oiseau dérobe des ouvrages

– l'oiseau dérobe de l'argent

– l'homme dérobe des ouvrages

– l'homme dérobe de l'argent

5. Analyse pragmatique : on utilisera le contexte si possible pour lever les dernières ambiguïtés

L'ambiguïté des langues naturelles

Une des grandes difficultés du traitement des langues naturelles est l'ambiguïté : chaque mot, chaque structure, chaque phrase peut souvent être analysé et interprété de plusieurs façons. Inversement, des mots ou des expressions différents peuvent désigner des objets identiques**.**

Il existe différents types d'ambiguïtés :

L'ambiguïté lexicale (ou catégorielle)

Se présente quand plusieurs catégories *lexicales* peuvent être associées à un mot [Bouillon, 1998]**.** Le mot *son* est ambigu lexicalement. Il peut désigner une céréale ou encore une impression auditive. (On parle aussi de synonyme)

Le mot *ferme*, quant à lui, est ambigu grammaticalement. Il peut appartenir à quatre catégories grammaticales différentes : nom, verbe, adjectif et adverbe. Le sens de ce mot sera très différent selon sa catégorie : nom = « bâtiment », verbe = « clore », adjectif = « dur » et adverbe = « fort ».

L'ambiguïté syntaxique (ou structurelles)

Une phrase est syntaxiquement ou structurellement ambiguë quand elle permet plusieurs analyses syntaxiques.

Exemples

(a) La belle ferme le voile: qui comporte trois analyses syntaxiques possibles (article-nom-verbe-article-nom; article-adjectif-nom-pronom-verbe, ou, article-nom-adjectif-pronom-verbe)

(b) Il a envoyé les dossiers nécessaires au procureur: l'ambiguïté est au niveau du syntagme 'au procureur' qui peut être complément de l'adjectif ou du verbe.

L'ambiguïté sémantique

L'ambiguïté sémantique apparaît quand un mot peut avoir plusieurs sens possibles, on parle alors de *polysémie* ;

Dans on distingue la polysémie de **:**

- **l'homographie** : deux mots s'écrivant de la même manière; exemple : *ferme* (bâtiment) *ferme* (adjectif)
- **l'homophonie** : deux mots qui se prononcent de la même manière; exemple : *la voie* et la *voix*

L'ambiguïté pragmatique

Apparaît lorsqu'une phrase correspond à plusieurs situations de communication; La phrase « *peux-tu me passer le sel »* peut signifier à la fois « passe-moi le sel » et « es-tu en mesure de le faire ».

Comment résoudre les ambiguïtés?

La levée de l'ambiguïté grammaticale (catégorielle)

La levée des ambiguïtés grammaticales est généralement basée sur le contexte, c'est-à-dire de la proximité de certains mots (un mot précédé de *M.* est un nom propre) ou de certains types de mots (ex. : le mot *le* est un pronom quand il est suivi d'un verbe et un déterminant quand il est suivi d'un nom), et de la présence d'une suite de lettres dans le mot (ex. : un mot se terminant par *-emment* est un adverbe). Toutefois, la seule présence d'un morphème n'est souvent pas suffisante pour déterminer la catégorie d'un mot. Un même morphème peut être associé à des mots de différentes catégories. C'est le cas de *-ique* qui peut être associé soit à un nom (ex. : la *mécanique*), soit à un adjectif (ex.: la pelle *mécanique*). De plus, ce ne sont

pas tous les types de mots qui disposent d'une morphologie leur étant propre. [Thibeault,2004]

Dans ce qui suit les grandes approches suivies pour la catégorisation grammaticale.

- ***Les différents modèles de catégorisation grammaticale automatique***

Les différents modèles utilisent tous les mêmes informations pour catégoriser les mots d'un texte : le contexte et la morphologie. Ce qui diffère, c'est la façon de représenter ces éléments.

Il existe deux grands types de catégoriseurs. Les premiers sont ceux qui appliquent des règles qui leur ont été fournies par des experts humains. Dans ce type de catégoriseurs, il y a très peu d'automatisation; c'est le concepteur qui dicte toutes les règles de catégorisation et qui fournit au besoin une liste de morphèmes. La conception d'un tel catégoriseur est fastidieuse, longue et coûteuse. De plus, les catégoriseurs ainsi conçus ne sont pas portables, c'est-à-dire qu'ils ne sont efficaces que pour une langue donnée et pour un domaine donné (ex. : le droit, la médecine, etc.). Le deuxième type apprend de façon automatique comment catégoriser. Parmi les catégoriseurs de ce type, il existe deux grandes familles : les catégoriseurs supervisés qui apprennent à partir de corpus précatégorisés, et les catégoriseurs non supervisés qui apprennent à partir de corpus bruts. Puis chaque grande famille peut être divisée en trois branches chacune : catégoriseur à base de règles, statistique et neuronal. La figure ci-dessous est évidemment simplifiée, dans les faits, certains catégoriseurs grammaticaux combinent des éléments de plusieurs approches.

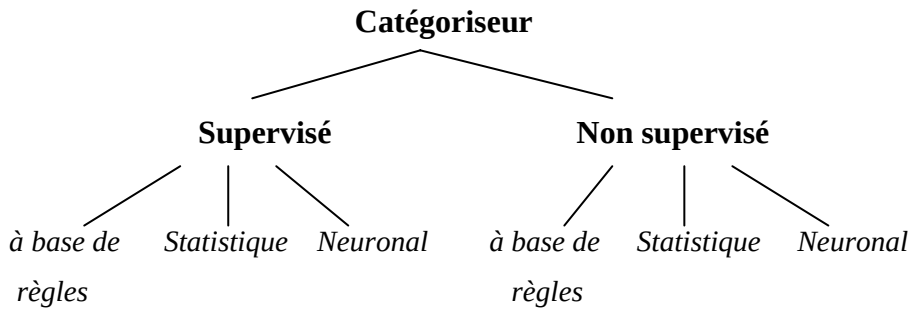

Figure 3 : Les différents types de catégoriseurs

Les premiers modèles créés étaient supervisés et à base de règles. Par la suite, des modèles statistiques ont été créés, dont le modèle de Markov qui est le plus répandu. Les logiciels créés sur la base de ces modèles étaient entraînés sur de très gros corpus précatégorisés et atteignaient 96 % d'efficacité pour l'anglais. Puis furent créés des modèles non supervisés pour éviter la catégorisation de gros corpus d'entraînement. Puis les modèles neuronaux ont

été développés tel que Lippman vers 1989. Ils atteignaient un niveau d'efficacité similaire pour l'anglais à celui obtenu avec les autres techniques. [Thibeault,2004]

La représentation des connaissances

La représentation des connaissances est un sous domaine de l'intelligence artificielle qui se rapporte à la conception, l'implémentation et la compréhension d'une représentation d'informations que les programmes peuvent utiliser dans le but de :

- Dériver (impliquer) d'autres informations ;
- Converser avec les êtres humains ;
- Résoudre des problèmes dans des domaines exigeant de l'expertise humaine.

En traitement de la langue on s'intéresse à la représentation des mots et leurs traits syntaxiques et sémantiques **(on parle du lexique)**, ainsi que les principes et les contraintes qui régissent la combinaison des mots, et qui permettent de distinguer les phrases correctes des phrases incorrectes **(on parle de la syntaxe)**.

Dans ce qui suit les différentes méthodes qui ont été développées en TALN pour la représentation des données linguistiques. On distingue :

Le calcul des prédicats et schémas de représentation logiques

Un des buts fondamentaux de l'Intelligence Artificielle est le traitement des langues naturelles par des moyens informatiques. Dans cette perspective, les diverses langues logiques jouent un rôle du premier ordre: elles peuvent non seulement être considérées comme un langage cible (dans le processus de traduction), mais également celui de langage intermédiaire entre la langue naturelle et le langage de programmation. [Desarte et Thayse, 2001]

Plusieurs formalismes ont été proposés. L'un des plus simples consiste en une traduction des énoncés dans **la logique des prédicats**, dite aussi **logique du 1er ordre** ou **logique classique.**

Ainsi, les énoncés

Socrate est grec

Les grecs sont des menteurs.

Pourront se traduire par les formules

grec(Socrate)

$\forall x$ *grec(x) menteur(x)*

A partir d'une telle représentation, l'ordinateur peut déduire de nouvelles propositions telles que: *menteur(Socrate)*

et donc répondre correctement à la question *"Est-ce que Socrate est un menteur?"*. Ce type de logique est évidemment très limité, car il ne porte que sur les **valeurs de vérité des** énoncés. On s'est aperçu très tôt que des phrases comme: *"La bataille aura lieu demain*" ne peuvent être qualifiées ni de vraies ni de fausses au moment où elles sont proférées. En fait de nombreux phénomènes échappent à la logique classique, tels que le rapport au temps, à l'action, les modalités (nécessaire, possible, contingente), les croyances, commandements et interrogations, etc. [Veronis, 2001]

Citons que *Montague (1974)* était le premier à proposer une logique typée pour représenter la sémantique des langues naturelles.

Les systèmes à base de grammaire

Une grammaire définit les principes et les contraintes qui régissent la combinaison des mots, et qui permettent de distinguer les phrases correctes des phrases incorrectes [Bouillon, 1998]**.** Les théories développées dans ce domaine sont nombreuses, on cite :

a) La théorie de grammaire formelle

La théorie de grammaire formelle qui remonte à la fin des années cinquante est due à *Chomsky* et a été particulièrement utile pour le développement des compilateurs. Elle utilise comme concepts de base :

– des symboles non terminaux, qui correspondent aux catégories syntaxiques : S, V, COD, Adj., Art., Adv., GN, GV, . . .

– des symboles terminaux, qui sont les mots du lexique ;

– des règles de réécriture, qui sont des combinaisons de symboles terminaux et non-terminaux.

Par exemple : **<phrase>::=<groupe nominal> <groupe verbal>**

<article>::= le / la / un / une

– un symbole distingué (initial), qui peut être par exemple : **<phrase>**

Voici un exemple simple de grammaire formelle :

```
<phrase>::=<groupe nominal> <groupe verbal>
<groupe nominal>::=<article><adjectif> <substantif>
<groupe verbal>::= <verbe> <attribut>
<verbe> ::=est
<article>::= le / un
<adjectif>::=petit/grand/beau/vilain
<substantif>::=chat/chien/garçon/tigre
<attribut>::= affamé/blessé/mort
```

Figure 5 : exemple d'une grammaire formelle

Elle permet de reconnaître des phrases telles que, *le petit chien est blessé*, *un vilain tigre est mort*, *le beau garçon est affamé*

La programmation de ces grammaires à l'aide de langages appropriés est assez simple, voici ci-dessous un exemple de programme écrit en Prolog de la grammaire formelle présentée dans la figure 4 : [Mahoudeaux, 2004]

```
phrase(x, y) -> groupe-nominal (x) groupe-verbal (y);
groupe-nominal(x,y,z,nil)   ->   article(x)   adjectif(y)
substantif(z);
groupe-verbal(x,y,nil) -> verbe(x) attribut(y);
article("le") ->;
article("un") ->;
substantif("chat") ->;
substantif("chien") ->;
substantif("garçon") ->;
substantif("tigre") ->;
adjectif("petit") ->;
adjectif("grand") ->;
adjectif("beau") ->;
adjectif("vilain") ->;
verbe("est") ->;
attribut("blessé") ->;
attribut("affamé") ->;
attribut("mort") ->;
```

Figure 6: exemple d'un programme en Prolog décrivant une grammaire formelle

b) Grammaires transformationnelles (dits aussi Génératives)

Ce sont des grammaires formelles (syntaxiques) auxquelles on a ajouté des règles de transformation. Introduites dans les années cinquante par *Chomsky*. Elles ont recours à des opérations de transformation.

Les Transformations sont des opérations s'appliquant sur des *structures de base* (dites *profondes*) pour dériver des structures *finales,* dites *de surface*. Elles permettent de réduire le nombre de règles de réécriture.

1 Exemples de transformation

- (a1). Vous avez invité qui (*Structure profonde)*
- (b1). Qui avez-vous invité ? (*Structure de surface)*

Deux transformations : (i) déplacement de *qui* en début de phrase (ii) inversion *sujet-verbe*

(a1). *Le garçon lit un livre* peut générer

(b1). une question : *qui lit un livre*

(b2). ou une forme passive : *un livre est lu par le garçon*

Pour plus de détails sur ces grammaires, nous conseillons les ouvrages ([Winograd,1983] **,** [Slonneger et Kurtz; 1995]**,** [Grune et Jacobs; 1994] **).**

c) Les réseaux de transition

L'idée générale qui a donné naissance à la notion de réseaux de transitions est de remplacer les règles des grammaires formelles par des formes de phrases acceptables. Ainsi, si l'on veut décrire les formes de phrases telles que *le petit garçon mange* et *la jeune fille chante une belle chanson*, on peut utiliser les deux séquences

– DET ADJ NOM VERBE

– DET ADJ NOM VERBE DET ADJ NOM

Si l'on veut rendre compte du fait qu'elles commencent de la même façon, il faudra utiliser une notation (par exemple { }) indiquant que la fin de la seconde est optionnelle. On peut aussi définir un symbole de répétition (*) pour les adjectifs dans les phrases : *la fille, la jeune fille, la belle jeune fille, . . .* ainsi qu'un symbole d'alternative (∨) pour indiquer qu'on peut remplacer tout cela par un pronom. On aura alors la forme unique :

(PRON∨ (DET ADJ* NOM)) VERBE {(PRON∨(DET ADJ* NOM))}

Les réseaux de transition offrent une notation plus simple pour représenter le même type de phénomènes.

On utilise un graphe formé d'un ensemble d'*états* reliés par des (*arcs orientés*) qui représentent des *transitions* entre les états. Les arcs sont étiquetés par des mots ou des catégories lexicales et les noms des états servent à la description du réseau. On distingue les *états initiaux*, marqués par une flèche, et les *états finals* (dans un double rond). Les séquences d'étiquettes rencontrées en allant, de toutes les façons possibles, d'un état initial à un état final, représentent l'ensemble des phrases acceptées par le réseau. [Winograd,1983]

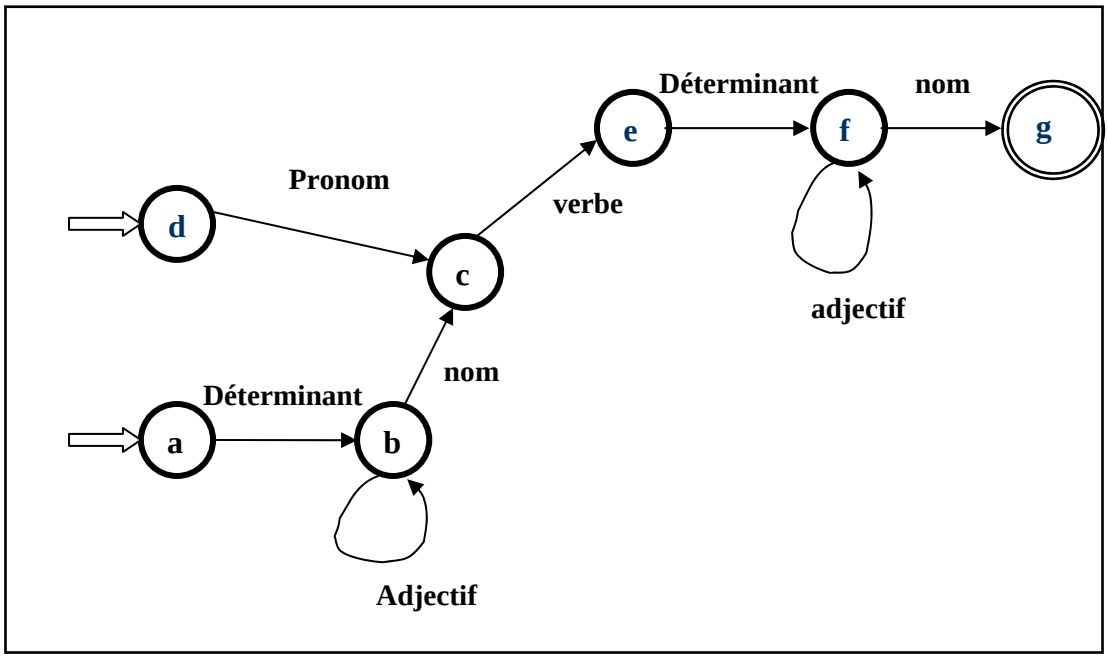

Figure 7: Un simple réseau de transition

d) Structure de trait

Une structure de trait décrit ou représente un objet par l'énumération de ses caractéristiques significatives. Chaque caractéristique est étiquetée (le trait) et la valeur correspondante à l'objet décrit lui est associée. La ST se présente comme un ensemble de paires attribut-valeur.

Exemple

```
[ CAT :    SN
  ACCORD : [ NOMBRE : SINGULIER
             PERSONNE : TROISIEME ] ]
```

Figure 8 : Exemple d'une structure de traits.

`CAT` et `ACCORD` sont les deux attributs de la ST dont les valeurs sont successivement `SN` et la ST

```
[ NOMBRE : SINGULIER
  PERSONNE : TROISIEME ]
```

(iii). La ST permet de regrouper et de mettre sur le même plan, dans une structure de données globale des informations de différentes natures qu'elles soient morphologiques, syntaxiques, sémantiques ;

(iv). Elle permet d'affiner les contraintes au niveau du lexique et de simplifier ainsi les règles de grammaire ; [Pierrel,1987]

Les modèles issus de la psychologie

Le problème auquel se heurtent les chercheurs en traitement informatique de la langue est la représentation du sens. Une des grandes questions qui se posent est la suivante : le sens d'une phrase est-il contenu entièrement dans la phrase, ou bien nécessite-t-il des notions extra-langagières ?

Puisque la compréhension de la langue nécessite des connaissances, que ces connaissances doivent être représentées sous une forme manipulable, une des disciplines choisies fut donc la psychologie.

Les travaux en psychologie sur la mémoire et la représentation des connaissances émergèrent dans les années 60, avec les travaux de Collins et Quillian sur les *réseaux sémantiques*, ceux de Minsky sur les frames et ceux de Sowa sur les *graphes conceptuels*. L'idée sous-jacente à ces travaux était de développer un modèle du stockage en mémoire de connaissances diverses et variées, connaissances qui pouvaient très bien correspondre au sens d'une phrase.

a) Les réseaux sémantiques

Le *réseau sémantique* a été proposé par Quillian en 1968, c'est un graphe composé de nœuds reliés par des liens étiquetés. A chaque nœud correspond un concept de l'univers à modéliser. Le but des réseaux sémantiques est de fournir une représentation souple des connaissances**.**

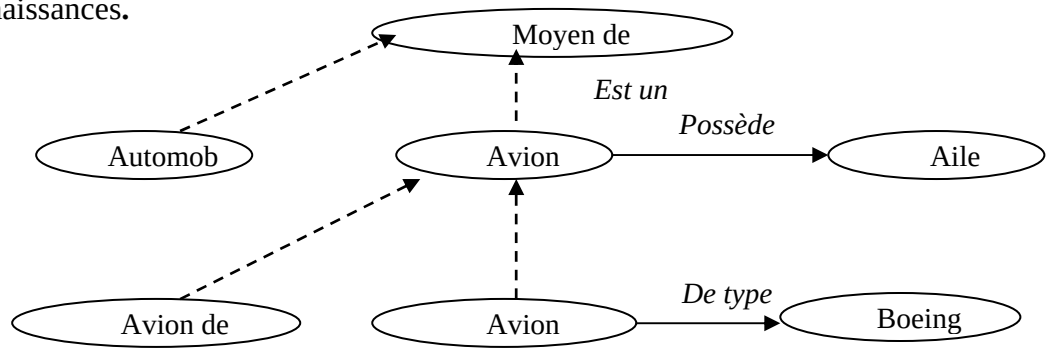

Figure 9 : Exemple d'un réseau sémantique

On utilise de façon fondamentale les propriétés des relations ; par exemple :

La transitivité : « est un » est une relation transitive ;

L'héritage via d'autres relations (le Boeing 747 a des ailes)

Relations *réciproques* (le Boeing 747 est un avion de passager).

b) Les graphes conceptuels

Les réseaux sémantiques ont comme avantage de pouvoir réaliser des inférences, c-à-d, déduire de nouvelles connaissances à partir de celles présentées par le graphe. Cependant, elles ne permettent de représenter que les relations **binaires** [Veronis, 2001]**.**

Ce désavantage a conduit Sowa (1984) à proposer le formalisme **de graphes conceptuels** qui est considéré comme extension pour les réseaux sémantique ([Veronis, 2001]**,** [Sowa, 1976])**.** Sowa prétend que toute forme de représentation pourrait être écrite sous forme de Graphe Conceptuel (GC). Le sens d'un concept se réduit à sa position relative par rapport aux autres concepts, il ne prend donc un sens que par rapport à un réseau sémantique modélisant les connaissances générales du système.

Dans le graphe conceptuel, *la boîte* contient le concept, *le cercle,* la relation et des *traits orientés* entre les divers éléments dirigent le sens de la lecture. L'avantage du graphe conceptuel est de représenter des données indépendamment de la langue.

Exemple

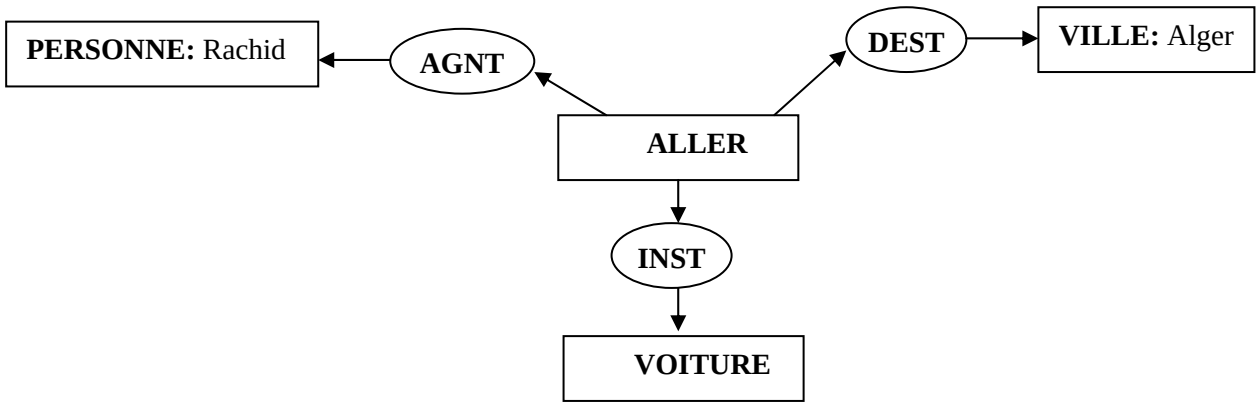

Figure 10: Graphe conceptuel correspondant à la phrase "Rachid se rend à Alger par voiture"

c) Les Schémas (Frame)

Développé en 1975 par *Minsky*, et repris par d'autres: *Fillmore*, *Simmons, Schank, Riesbek.* L'idée des *frames* est d'offrir un support permettant de regrouper l'ensemble des informations sur un objet. Elle a pris corps dans le cadre des travaux sur la compréhension du monde courant (vision et compréhension de scène, langage naturel et compréhension de textes) [Dominé,1988]**.**

Le frame regroupe un ensemble d'informations concernant un concept particulier. Les concepts sont généralement organisés en treillis suivant la relation *sorte-de*. A chaque nœud

est associé un schéma qui décrit les propriétés du concept dont héritent les concepts descendants [Bouillon, 1998].

Exemple

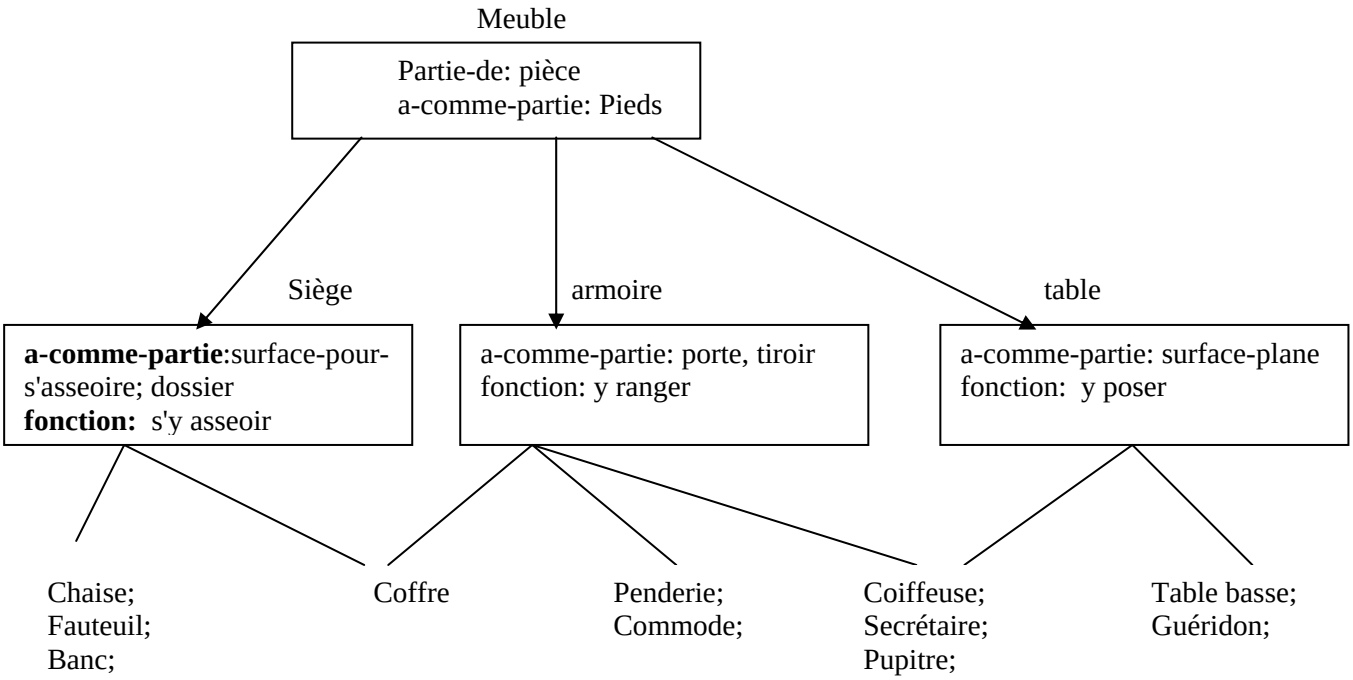

Figure 11 : Exemple de Frame

d) Les scripts (ou scénario),

Il existe un type d'analyse beaucoup plus pragmatique : l'utilisation de scripts (ou *scénario*), qui reposent sur une analyse du contexte. Cette approche s'appuie sur des modèles de représentation des connaissances issus des travaux en intelligence artificielle.

– on suppose que le locuteur connaît un grand nombre de choses sur le sujet donné

– la donnée du sujet implique l'allocation de valeurs par défaut à des variables, et on précise les éventuelles anomalies dans le déroulement de l'action.

Script:	COMPUTER-ON.
Track:	Computer Room.
Props:	Computer. On-button. Keyboard.
Roles:	User.
Entry Conditions:	User needs computer. Computer is off.
Results:	User can use computer. Computer is on.

Scene 1: Locating On Button.

If PC, look on front (or back) of computer for on-switch.

Wait.

Scene 2: Booting up.

Computer starts to boot up.

If computer starts ok, go to (Operating System).

Else:

Look at error.

Fix problem. (Another script).

Turn off computer.

Go to (Locating On Button).

Scene 3: Operating System.

Computer has started, run program needed.

User computer.

Figure 12: Exemple d'un script [Matthews]

L'un des systèmes les plus connus utilisant le principe de scripts étant SAM de Cullingford (Script Applier Mechanism) ([Matthews,1999] , [Riesbeck, 1986])

Les algorithmes d'analyse syntaxique

L'objectif des algorithmes d'analyse syntaxique est de découvrir la forme syntaxique de la phrase. De nombreuses techniques d'analyse ont été proposées et les algorithmes sont de complexité polynomiale[1]. L'algorithme CYK constitue une référence fondatrice dans ce domaine. Cependant l'algorithme le plus productif est celui de Earley. Il a connu de plus de 20 ans d'existence de nombreuses adaptations. On doit également citer Tomita qui a proposé un certains nombre d'algorithmes, ainsi que des heuristiques particulièrement adaptées au TAL, comme la technique de coins gauche. [Abeillé et Blache, 1999]

On peut aussi distinguer les approches probabilistes**,** (voir [Manning et Schiitze, 2000]**)**, elles s'appuient généralement sur des grammaires apprises automatiquement sur corpus (préalablement annoté ou non) : l'étiqueteur morpho-syntaxique ne dispose que d'un dictionnaire indiquant pour chaque forme sa (ou ses) catégorie(s) possible(s), ordonnées selon leur probabilité, et d'une centaine de bigrammes (ou trigrammes) pour les séquences de catégories possibles. L'analyseur maximise la probabilité d'une dérivation, pour une phrase, comme produit des probabilités des règles qu'elle met en jeu. Les méthodes probabilistes ont pour avantages connus l'autoapprentissage des règles sur corpus [Abeillé et Blache, 1999].

[1] La complexité (en temps) polynomiale (au pire de cas) est en fonction de la longueur de la phrase à analyser, mais la taille de grammaire peut en pratique jouer un rôle important, de même que le temps d'accès au lexique.