RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

---

UNIVERSITE ECHAHID HAMMA LAKHDAR - EL OUED
FACULTÉ DES SCIENCES EXACTES
Département D'Informatique

Mémoire de Fin D'étude

Présenté pour l'obtention du Diplôme de

MASTER ACADEMIQUE

Domaine : **Mathématique et Informatique**
Filière : **Informatique**
Spécialité : **Systèmes Distribués et Intelligence Artificielle**

Présenté par :

- **HADDAD Aimen**
- **AHTHERIB Abdelouahab**

Thème

Conception et réalisation d'un contrôleur d'inférence pour les bases de données déductives multi-niveaux.

M.	……	MCA	Président
M.	……	MAA	Rapporteur
M.	MEFTAH Mohammed Charaf Eddine	MAA	Encadreur

Année Universitaire : 2016-2017

REMERCIEMENTS

Après touts, nous tenons à remercier le Dieu Tout-Puissant qui nous a bénis, qui nous a aidés à terminer ce travail.

En présentant ce travail, nous tenons à remercier Mr. MEFTAH Mohammed Charaf Eddine, l'encadreur pour sa serviabilité, sa disponibilité et ses remarques constructives qui nous ont utiles tout au long de notre projet.

Nous remercions tous ceux qui nous ont aidés à atteindre notre objectif, en particulier Mr. ZOUBEIDI Marouane , Docteur en Informatique à l'université de EL oued , Mr. LEJDEL Brahim et Mr. ZAIZ Faouzi pour leur soutien continu.

Nous tenons à remercier les messieurs membres de jury pour l'honneur qu'ils nous ont fait en acceptant de juger notre travail.

Je dédie :

- Notre source du savoir : prophète Mohammed
- A mon père qui n'a jamais hésiter de me soutenir
- A la prunelle de mes yeux Ma mère
- A toute la famille, mes frères et mes sœurs
- A mes amis et mes collègues de promotion
- A tout ce qui l'ont participé de prés ou loin tout au long de mon cursus scolaire
- Tous ceux qui me connaissent.

HADDAD AIMEN

Je dédie ce travail à :

- Ma très chère mère.
- Mon cher père.
- Mes frères.
- Mes sœurs.
- Tous mes amis (es).
- Et tous ceux qui me connaissent.

Abd El-oahab AHTIRIB

ملخص

تتكون قاعدة البيانات الاستنتاجية من مجموعة من الأسس (العلاقات) تسمى المشتقة أو المتعمدة، التي يعرف امتدادها بقواعد استنتاجية. هذا النوع من قاعدة البيانات يسمح للحد من المعلومات عبر مستويات مختلفة. ومن أجل الحفاظ على سرية المعلومات، حاولنا من خلال عملنا حل مشاكل حماية هذه البيانات. وقد تم ذلك من خلال تطوير نظام خبير مع وحدة تحكم الاستدلال على أساس سلسلة متخلفة.

من أجل تحقيق أهدافنا أجرينا دراسة حالة حيث استخدمنا برولوغ. وقد قدم لنا هذا التنفيذ نتائج مرضية للغاية.

الكلمات الرئيسية: عودة تسلسل، قاعدة بيانات متعددة المستويات، حاسوب، وحماية البيانات.

Résumé

Une base de données déductive (BDD) est constituée d'un ensemble de prédicats (relations), dits dérivés ou intentionnels, dont l'extension est définie par des règles déductives. Ce type de base de données permet la déduction des informations à travers les différents niveaux. Afin de maintenir la confidentialité des informations secrètes, nous avons à travers notre travail essayé de résoudre les problèmes de protection de ces données. Ceci a été effectué par la réalisation d'un système expert avec un contrôleur d'inférence basé sur le chainage arrière.

Afin de réaliser nos objectifs nous effectué un cas d'étude où nous avons utilisé Prolog. Cette implémentation nous a donné des résultats très satisfaisants.

MOTS-CLES : Chainage arrière, base de données multi niveaux, Prolog, protection de données.

Summary

A déductive database (BDD) consists of a set of predicates (relations), called derivative or intentional, whose extension is defined by deductive rules. This type of database allows the deduction of information across different levels. In order to keep the confidentiality of the information secret, we have through our work tried to solve the problems of protection of this data. This was done by the réalisation of an expert system with an inference controller based on backward chaining.

In order to achieve our objectives we carried out a case study where we used Prolog. This implementation has given us very satisfactory results.

KEYWORDS : Reverse chaining, multi-level database, Prolog, data protection.

Sommaire

Table des matières

Table des figures

Introduction générale

Un système expert est un outil capable de reproduire les mécanismes cognitifs d'un expert, dans un domaine particulier. Il s'agit de l'une des voies tentant d'aboutir à l'intelligence artificielle. Le cœur du système expert est le moteur d'inférence. Des informations peuvent être inféré ou déduite à partir de faits stockés dans une base de données, ceci est une base de données déductive. Une BD déductif est un système qui comporte des possibilités pour définir des règles grâce à la déduction. Dans un système de gestion de bases de données déductif, un langage déclaratif (tel que Prolog et Datalog) est utilisé pour spécifier des règles. Un mécanisme d'inférence (ou mécanisme de déduction) peut alors déduire de nouveaux faits à partir de la base de données en interprétant les règles. La conception d'un contrôleur d'inférence pour les bases de données déductives multi-niveaux est un sujet qui a été largement étudié est encore un domaine de recherche important aujourd'hui.

Dans cette direction, notre projet consiste la protection des données confidentielles dans des entreprises ou des structures privées. Nous avons spécialement visé le contrôleur d'inférence pour les bases de données déductives où nous avons effectué un programme à l'aide d'un groupe des outils tels que java et prolog pour protéger toutes les informations sur la pour les bases de données.

Les problèmes dans ce projet est le contrôle les faits et les règles de chaque niveau de la base de données, nous constatons qu'il ya des faits ou des règles mal classés entre les différents niveaux de la base de données,

Afin de répondre à nos problèmes et les résoudre nous avons procédé comme suit. Nous avons avant tous effectué une étude théorique afin de bien nous situé dans le domaine. Par la suite nous avons effectué une conception dans la quelle nous avons proposé notre architecture et la solution proposée. Enfin dans une dernière partie nous avons choisis le langage déclaratif Prolog pour l'implémentation du cas d'étude.

Chapitre 01 :

Notions sur les bases de données déductives

1. Introduction

Depuis que la notion de base de données déductive est bien comprise, sous la pression des applications potentielles, les concepteurs de systèmes s'efforcent de proposer des algorithmes et méthodes efficaces pour réaliser des SGBD déductifs traitant de grands volumes de faits (les bases de données classiques) et de règles.

L'objectif est de fournir un outil puissant pour aider à résoudre les problèmes, que nous nous concentrons sur les questions des étudiants qui nécessitent de grandes quantités de données et les règles de la solution.

Ce chapitre fournit des définitions générales pour la base de données déductive et montrer comment leur représentation dans le SGDB et le problème de la protection des informations contenues dans les multi- niveaux de base de données déductives, et nous allons clarifier davantage le contenu de ce chapitre.

2. Définition de base de données déductive

Une base de données déductive (BDD) est constituée d'un ensemble de prédicats (relations), dits dérivés ou intentionnels, dont l'extension est définie par des règles déductives

Base de données intentionnelle : Prédicat grand parent à 2 arguments (ou relation grand parent à 2 attributs) :

Grand parent(X, Y) ← parent(X, Z) AND parent (Z, Y)

Dans un système de gestion de bases de données déductif, un langage déclaratif est utilisé pour spécifier des règles. Un mécanisme d'inférence (ou mécanisme de déduction) peut alors déduire de nouveaux faits à partir de la base de données en interprétant les règles.

3. Problématique des SGBD déductifs

Un SGBD déductif est tout d'abord un SGBD. En ce sens, il doit posséder un langage de description de données permettant de définir les structures des prédicats de la base B1, B2, ... Bn, par exemple sous forme de relations, et les contraintes d'intégrité associées. Il offre aussi un langage de requête permettant de poser des questions et d'effectuer des mises à jour. Ces deux langages peuvent être intégrés et posséder une syntaxe propre, ou plusieurs, offertes aux usagers. Parmi ces langages, il est permis de penser que SQL restera une des interfaces offertes par un SGBD déductif, surtout devant la poussée de sa normalisation.

4. Langage de règles

L'interface nouvelle offerte par un SGBD déductif est avant tout le **langage de règles**.

- ✓ **Langage de règles (Rule Langage)**

Langage utilisé pour définir les relations déduites composant la base intentionnelle permettant d'écrire des programmes de règles du style <condition> = > <action>.
Le langage de règle est donc utilisé afin de spécifier les parties conditions et actions des règles de déduction. Plus précisément, à partir des prédicats B1, B2, ... Bn définis dans la base implantée (extensionnelle), le langage de règles permet de spécifier comment construire des prédicats dérivés R1, R2, ... interrogeables par les utilisateurs. Un langage de règles peut donc être perçu comme une extension des langages de définition de vues et de *triggers* des SGBD relationnels classiques.
L'extension est de taille car le langage de définition et de manipulation de connaissances va intégrer les fonctionnalités suivantes :
1. la possibilité d'effectuer les opérations classiques du calcul relationnel (union, restriction, projection, jointure, différence).

2. le support des ensembles incluant les fonctions d'agrégats traditionnelles des langages relationnels classiques ainsi que les attributs multi values.

3. la récursivité, qui permet de définir une relation déduite en fonction d'elle-même.

4. la négation, qui permet de référencer des faits non existants dans la base.

5. les fonctions arithmétiques et plus généralement celles définies par les utilisateurs.

6. les mises à jour des faits au travers des règles.

7. la modularité avec la gestion de niveaux d'abstraction successifs et de méta-règles.

En bref, toutes les facilités qui existent dans les langages de développement de bases de données vont chercher à figurer dans les langages de règles. L'objectif visé est d'ailleurs de remplacer ces langages. ***[2]***

4.1 Couplage ou intégration

La réalisation d'un SGBD déductive nécessite donc l'intégration d'un moteur de règles au sein d'un SGBD. Celui-ci doit être capable de réaliser l'inférence nécessaire lors de l'interrogation, voire la mise à jour, des prédicats dérivés. Une fonctionnalité analogue à celle d'un SGBD déductif peut être obtenue en couplant un moteur de règles à un SGBD. On distingue le couplage faible où les deux composants restent visibles à l'utilisateur du couplage fort où seul le langage de règles est visible. La figure1 illustre les techniques de couplage et d'intégration. Un SGBD déductif essaie donc de réaliser l'intégration forte, en offrant un langage de définition et de manipulation de connaissances intégré.

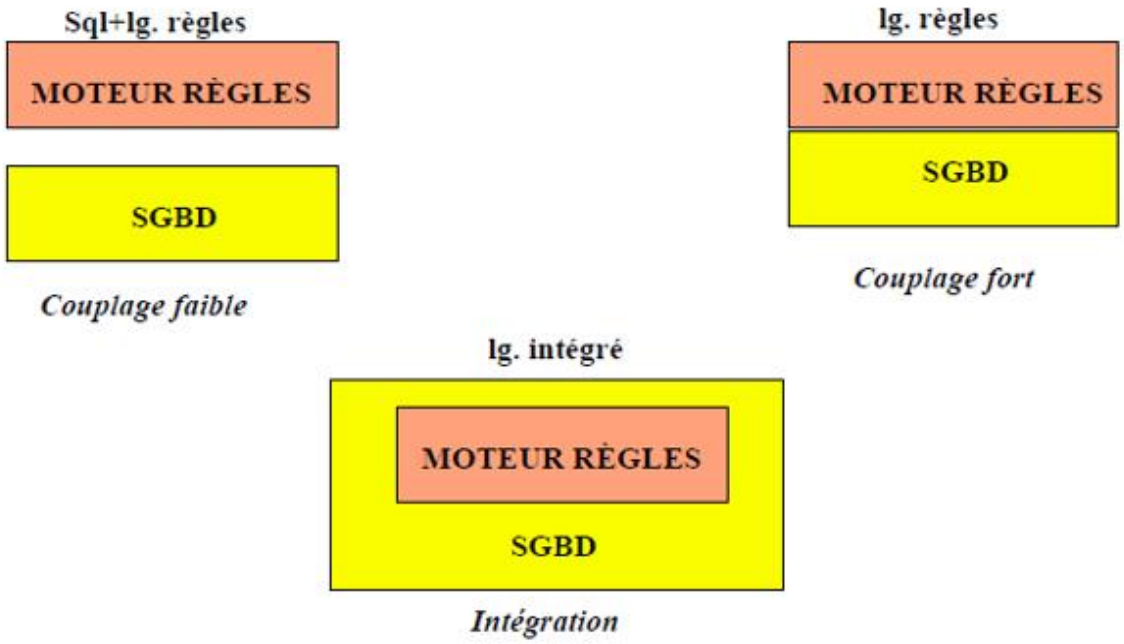


Figure 1: couplage versus integration

Réalisant l'inférence à partir des faits et des règles, sans générer de faits inutiles ni redondants, mais aussi sans oublier de réponses. Le problème de l'efficacité du mécanisme d'inférence en présence

d'un volume important de faits et de règles, notamment récursives, est sans doute l'un des plus difficiles routes. ***[2]***

4.2 Prédicats extensionnels et intentionnels

Dans le contexte logique, une base de données est perçue comme un ensemble de prédicats. Les extensions des **prédicats extensionnels** sont matérialisées dans la base de données. Les prédicats extensionnels correspondent aux relations du modèle relationnel.
Une base de données est manipulée par des programmes logiques constitués d'une **:**

- ✓ **Prédicat extensionnel (Extensionnel prédicat)**

Prédicat dont les instances sont stockées dans la base de données sous forme de tuples.
Suite de clauses de Horn qui définissent des **prédicats intentionnels**. Un prédicat intentionnel est donc défini par un programme de règles logiques ; il correspond à une vue du modèle relationnel.

- ✓ **Prédicat intentionnel (Intensionnel prédicat)**

Prédicat calculé par un programme constitué de règles logiques dont les instances ne sont pas stockées dans la base de données.

Une base de données logique est constituée d'un ensemble de prédicats extensionnels constituant la base de données extensionnelle et d'un ensemble de prédicats intentionnels constituant la base de données intentionnelle. Les règles permettant de calculer les instances des prédicats intentionnels sont donc partie intégrante de la base de données logique. La figure 2 illustre les notions de bases de données extensionnelle et intentionnelle, la seconde étant dérivée de la première par des règles stockées dans la méta-base du SGBD.

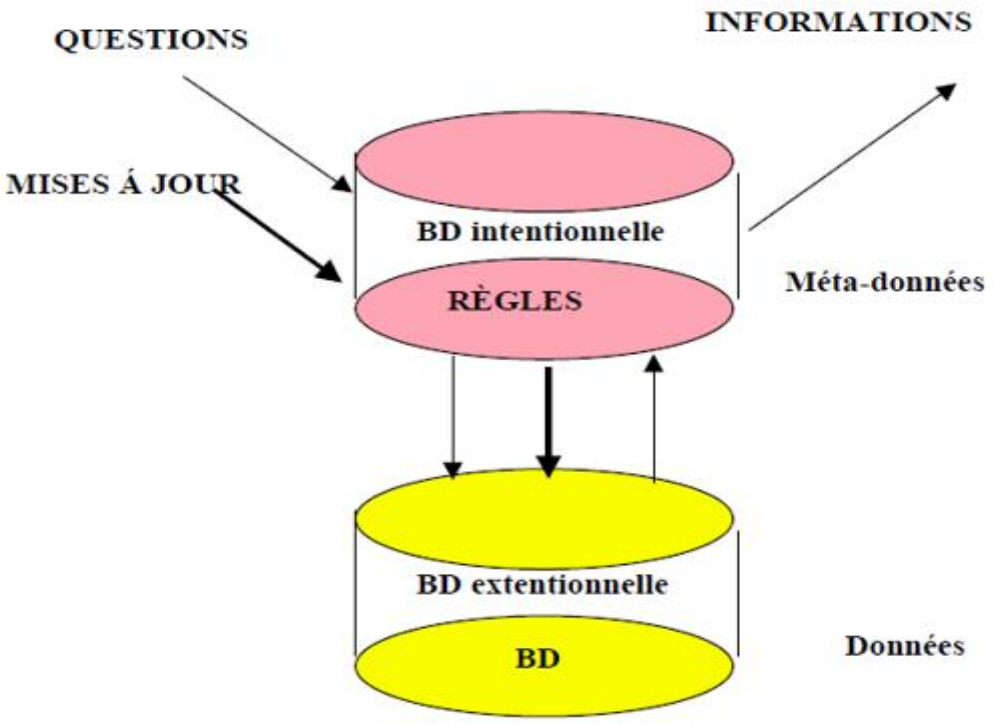


Figure 2: Base de données extensionnelle et intentionnelle

4.3 Architecture type d'un SGBD déductif intégré

Le SGBD est dit **déductif** car il permet de déduire des informations à partir de données stockées par utilisation d'un mécanisme d'inférence logique. Les informations sont les tuples des prédicats intentionnels ; elles peuvent être déduites lors de l'interrogation des prédicats intentionnels ou lors des mises à jour des prédicats extensionnels. La mise à jour des prédicats intentionnelles est difficile il faut :

- ✓ **SGBD déductive (Déductif DBMS)**

SGBD permettant de dériver les tuples de prédicats intentionnels par utilisation de règles.
Théoriquement répercuter sur les prédicats extensionnels, ce qui nécessite une extension des mécanismes de mise à jour En résumé, un SGBD déductif va donc comporter un noyau de SGBD permettant de stocker faits et règles dans la base, et d'exécuter des opérateurs de base comme un SGBD classique. Au-delà, il va intégrer des mécanismes d'inférence pour calculer efficacement les faits déduits. Un langage intégré de définition et manipulation de connaissances permettra la définition des tables, des règles, l'interrogation et la mise à jour des informations (voir figure 3).

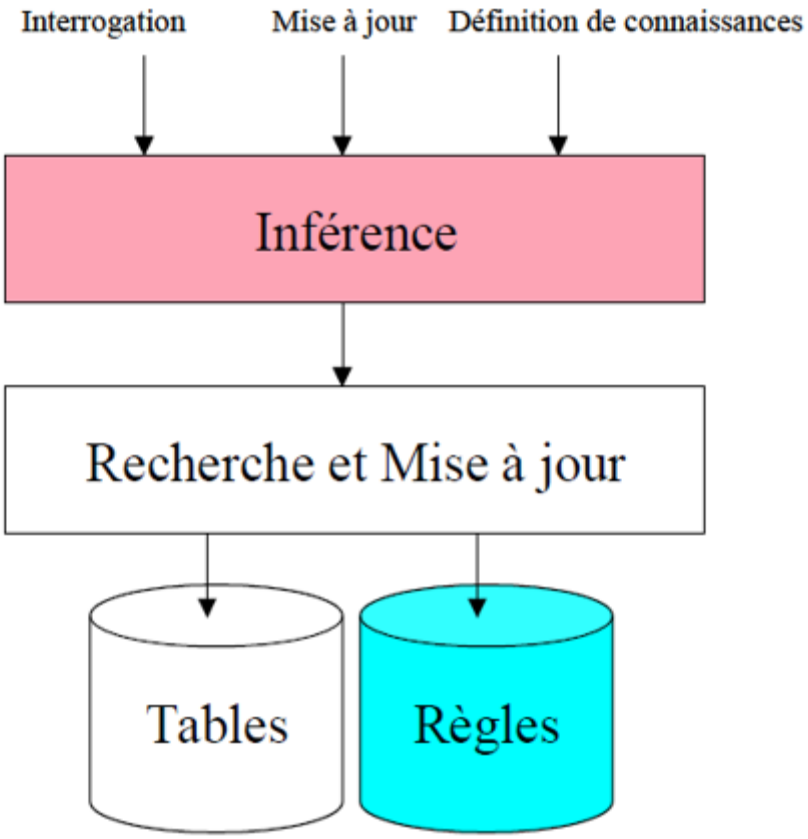


Figure 3:Architecteurs d'un SGBD déductives

5. Architectures des bases de données multi-niveaux

Dans une base de données à plusieurs multi-niveaux, nous constatons qu'il comprend plusieurs types d'approches, y compris :

- ***approche filtre***
- ***approche noyau de sécurité***
- ***approche sujet de confiance***
- ***approche réplication***

❖ Approche filtre

✓ Principes

- On ajoute un filtre au SGBD, entre le client et le SGBD.
- Chaque client est associé à un niveau de sécurité unique (mono-niveau).
- Le serveur gère toutes les données de la base (multi-niveaux).
- Comment fonctionne le filtre ?

- authentifie le client pour connaître son niveau de sécurité.
- Intercepte toutes les requêtes émises par les clients et toutes les réponses fournies par le serveur.

Fonctionnalités du filtre

- modifie la requête émise par le client pour que l'évaluation de la requête ne fournisse pas de données ayant un niveau de classification supérieur au niveau du client.
- filtre le résultat pour éliminer toutes les données qui n'auraient pas un niveau de sécurité inférieur ou égal à celui du client.

✓ Avantages

- simplicité de mise en œuvre (pas de modification des fonctionnalités).
- le filtre est de confiance.
- Inconvénients
- vulnérabilité aux attaques (le SGBD n'est pas de confiance).
- contournement du filtre.

❖ Approche noyau de sécurité

✓ Principes

- Développer un SGBD en s'appuyant sur un OS multi-niveaux.
- L'authentification et les contrôles d'accès sont réalisés par l'OS.
- Un OS multi-niveaux ne gère que des fichiers mono-niveau (un fichier ne contient que des données ayant le même niveau de classification).
- Partitionnement des données du SGBD sur plusieurs fichiers.

✓ Avantages

- Le SGBD n'a pas besoin d'être de confiance (on se repose sur l'OS).
- En fait, on fait l'hypothèse que certaines fonctions du SGBD sont de confiance.

✓ Inconvénients

- Performances à cause du partitionnement lié à la gestion mono-niveau des fichiers.
- Avoir un OS multi-niveaux.

❖ Approche sujet de confiance

✓ Principe

- Variante de l'approche noyau de sécurité.

- Les contrôles (accès et authentification) ne sont pas effectués par l'OS mais par le SGBD.
- Le SGBD gère son propre ensemble de niveaux de sécurité.
- Il effectue les contrôles d'accès conformément à la relation d'ordre sur cet ensemble.

✓ Avantages

- Gain de performance.

✓ Inconvénients

- Adaptation du SGBD à l'OS.

❖ Approche réplication

✓ Principe

- Un serveur séparé pour gérer toutes les données de niveau inférieur à un niveau donné.
- Autant de serveurs que de niveaux.
- Les données sont répliquées.
- Une donnée de niveau n est répliquée dans tous les serveurs de niveau supérieur à n.
- Les serveurs SGBD peuvent ne pas être de confiance.
- C'est le serveur frontal qui est de confiance (authentification et contrôle d'accès).

✓ Avantages

- Un seul serveur intervient pour évaluer la requête du client.
- Le serveur ne contient que les données que le client le droit d'observer.
- Pas de possibilité de détourner les données/requêtes.

✓ Inconvénients

- Réplication des données: cohérence, dégradation des performances.

6. contrôle d'accès à la base et sécurité de données

- Un SGBD organise les données et donne aux utilisateurs des moyens pour extraire de l'information.
- Si l'accès est incontrôlé, on n'y mettrait pas de données sensibles.
- Problème de confidentialité : prévention de la divulgation non autorisée de données/d'information.
- Au sens large, la sécurité informatique inclut également.
- Les problèmes d'intégrité : prévention de modification non autorisée des données.

- Disponibilité : prévention de refus d'accès autorisé à des données des modifications anarchiques.

6.1-Principes fondamentaux de la sécurité

• **Identification/Authentification :** qui êtes-vous, pouvez-vous le prouver ?

• **Autorisation :** pouvez-vous faire cette opération sur cet objet ?

• **Intégrité :** les données sont protégées contre toute modification accidentelle ou malveillante.

• **Confidentialité :** les données restent privées et elles ne peuvent pas être vues par des utilisateurs non autorisés.

• **Audit :** un audit et une journalisation sont essentiels pour résoudre les problèmes de sécurité a posteriori.

• **Non-répudiation :** le système peut prouver qu'un utilisateur a fait une opération.

• **Disponibilité :** capacité pour des systèmes de rester disponibles pour les utilisateurs légitimes (ex.: pas de refus de service).

6.2 Définitions menaces, vulnérabilités et attaques

Une menace : est un événement potentiel, malveillant ou pas, qui pourrait nuire à une ressource toute opération préjudiciable à vos ressources est une menace.

Une vulnérabilité : est une faiblesse qui rend possible une menace peut être due à une mauvaise conception à des erreurs de configuration ou à des techniques de codage inappropriées et non fiables.

Une attaque : est une action qui exploite une vulnérabilité ou exécute une menace par ex., envoyer des données d'entrée malveillantes à une application ou saturer un réseau en vue d'entraîner un refus de service. ***[3]***

6.3 Les risque

- Attaques sur le SGBD lui-même.
- Failles connues classiques (buffer overflows, bugs d'authentification…).
- Failles dans les applications associées : serveurs Web d'administration, démons SNMP (Simple Network Management Protocol), programmes setuid root installés par le SGBD.
- Mauvaises configurations.
- Modes d'authentification dégradés (.rhosts…).
- Mots de passe par défaut.
- Fichiers de la BD non sécurisés (lecture par tous).

- Interception de mots de passe.
- Par écoute du réseau.
- Par lecture de fichiers de configuration sur disque.

6.4 Contrôle d'accès et SGBD

Deux niveaux d'accès à un SGBD :

- Manipulation des données sur les relations de la base : protection sur les entrées de la base.
- Opérations composées comme des vues : restriction de ce que l'utilisateur peut faire sur la base.

Une politique de sécurité doit viser deux propriétés :

- Complétude : tous les attributs sont protégés, même si la protection indique accès libre.
- Cohérence : pas de conflit entre les différentes règles de sécurité.

À la création d'un objet, son propriétaire a tous les droits, y compris celui d'accorder ou de révoquer des privilèges à d'autres utilisateurs.

Un privilège est composé des éléments suivants :

- Utilisateur qui accorde le privilège.
- Objet.
- Action permise.
- Transmission possible du privilège.

6.5 Sécurité discrétionnaire

Deux mécanismes :

- Les vues : permet de cacher des informations sensibles à des utilisateurs non-autorisés.
- Sous-système d'autorisation : permet aux utilisateurs ayant des privilèges d'attribuer sélectivement et dynamiquement ces privilèges à d'autres utilisateurs (et de les révoquer).

Contrôle d'accès discrétionnaire :

- Par des vues et le contrôle d'accès sur ces vues.
- L'accès aux données seulement par des vues rappelle le modèle formel de Clark-Wilson.
- Comment donner accès à un utilisateur l'accès à une vue sans avoir à lui donner l'accès à toutes les relations sous-jacentes ?
- Par l'introduction d'un mode de référence.

- Ce mode est protégé par un privilège (comme les requêtes SQL).
- L'utilisateur n'a donc besoin que du privilège de voir la vue et de celui du mode référence sur les relations sous-jacentes à cette vue.

6.6 Sécurité obligatoire

Sécurité multi-niveaux (cas particulier) :

Dans une politique de sécurité multi-niveaux les utilisateurs reçoivent un niveau d'habilitation les informations reçoivent un niveau de classification niveaux dans un ensemble partiellement ordonné très secret > Secret > Confidentiel > Public.

6.7 SGBD et sécurité multi-niveaux

Comment obtenir un niveau élevé de protection sur les éléments d'un SGBD ?

- Contrôle d'accès obligatoire (étiquettes de sécurité).
- Les principaux vendeurs de SGBD (Oracle, Informix, Sybase) ont tous une version MAC de leur système, évalué au niveau B1 du Livre Orange. Modèle simplifié de Bell-La Padula.
- Soit S, un ensemble d'utilisateurs du système du SGBD.
- Soit O, un ensemble d'objets (ex: tables, relations).
- Une relation <= d'ordre partiel sur les étiquettes L, aussi appelées classes d'accès.
- Soit fs : S $\longrightarrow$ L et fo : O $\longrightarrow$ L , assignant respectivement des classes d'accès aux utilisateurs et aux objets.

✓ **Règle de simple sécurité :**

- Un sujet s peut lire un objet o seulement si sa classe d'accès domine celle de l'objet, c'est-à-dire fo(o) <= fs(s).
- Règle étoile :
- Un sujet s peut modifier un objet o seulement si sa classe d'accès est
- Dominée par celle de l'objet, c'est-à-dire fs(s) <= fo(o).
- Ces politiques s'appliquent aux manipulations sur le SGBD et s'adressent aux flots d'information directs
- L'information peut aussi s'échapper par des canaux cachés

- refuser l'accès à une requête donne de l'information à un utilisateur s'il ne savait pas déjà que la requête allait échouer. ***[3]***

6. Conclusion

A la fin de ce chapitre nous constatons que les SGBD déductifs permettent d’associer de façon souple les fonctionnalités de gestion de données des SGBD et celles des systèmes experts. De nombreuses applications utilisant de tels systèmes sont opérationnelles.

Chapitre 02 :
Les systèmes experts

1. Introduction

La notion de systèmes experts est une notion assez ancienne qui est apparu dans les années 70 avec l'apparition du système expert célèbre MYCIN dont le but était d'aider les médecins à effectuer le diagnostic est le soin des maladies infectieuses du sang. La version de base contenait 200 règles ensuite 300 règles concernant les méningites ont été ajoutées.

Aujourd'hui, les systèmes experts constituent une technologie bien définie faisant partie des systèmes à base de connaissances. Les systèmes experts ont comme finalité la modélisation de la connaissance et de raisonnement d'un expert (ou d'un ensemble d'experts) dans un domaine donné fixe.

Pour cela, trois acteurs principaux doivent contribuer à l'élaboration d'un système expert à savoir : l'utilisateur final, l'expert du domaine et l'ingénieur de connaissances. L'interaction entre ces trois acteurs amènera à l'élaboration d'une première version de systèmes experts contenant une base de connaissances, une base de faits et un moteur d'inférence effectuant une forme définie de raisonnement.

Dans ce chapitre, nous allons parler des concepts de base des systèmes experts entre autre : la définition, l'historique, constitution, rôle, la structure, le fonctionnement, la classification, caractéristiques, le développement technologique, les avantages et les inconvénients des systèmes experts.

2. Définition d'un système expert

D'une manière générale, un système expert est un outil capable de reproduire les mécanismes cognitifs d'un expert, dans un domaine particulier. Il s'agit de l'une des voies tentant d'aboutir à l' intelligence artificielle.

En outre, Un système expert est un système d'aide à la décision basé sur un moteur d'inférence et sur une base de connaissances. Il est la transcription logicielle de la réflexion d'un expert dans un domaine donnée.

Plus précisément, un système expert est un logiciel capable de répondre à des questions, en effectuant un raisonnement à partir de faits et de règles connus. Il peut servir notamment comme outil d' aide à la décision. ***[4]***

3. Historique

Le premier système expert fut Dendral en 1965, créé par les informaticiens Edward Feigenbaum, Bruce Buchanan, le médecin Joshua Lederberget le chimiste Carl Djerassi. Il permettait d'identifier les constituants chimiques d'un matériau à partir de spectrométrie de masse et de résonance magnétique nucléaire, mais ses règles étaient mélangées au moteur. Il fut par la suite modifié pour en extraire le moteur de système expert nommé Meta-Dendral.

Le plus connu, peut-être, fut Mycin en 1972-73, système expert de diagnostic de maladies du sang et de prescription de médicaments, avec un vrai moteur et une vraie base de règles. Cependant ses règles étaient affectées de coefficients de vraisemblance qui donnaient à chacune d'entre elles un poids particulier face aux autres. Le moteur produisait un chaînage avant simple tout en calculant les probabilités de chaque déduction, ce qui le rendait incapable d'expliquer la logique de son fonctionnement et de détecter les contradictions. Quant aux experts, ils étaient obligés de trouver des coefficients de vraisemblance pour chacune des conclusions de leurs règles, une démarche compliquée et antinaturelle qui déniait leur capacité de raisonnement.

Les systèmes experts ont eu leur heure de gloire dans les années 1980, où on a trop rapidement pensé qu'ils pourraient se développer massivement. En pratique, le développement de ce genre d'application est très lourd car, lorsque l'on dépasse la centaine de règles, il devient difficile de comprendre comment le système expert « raisonne » (manipule faits et règles en temps réel), et donc d'en assurer la mise au point finale puis la maintenance.

Le projet Sachem (Système d'Aide à la Conduite des Hauts fourneaux En Marche, chez Arcelor), opérationnel dans les années 1990, est l'un des derniers projets « système expert » à avoir vu le jour. Il est conçu pour piloter des hauts-fourneaux en analysant les données fournies en temps réel par un millier de capteurs. Le projet a couté entre 1991 et 1998 environ 30 millions d'euros, et le système économise environ 1,7 euros par tonne de métal. ***[5]***

4. Constitution

Un système expert se compose de deux parties :

- **une base de connaissance** qui est composé de la base de faits (un ensemble de connaissance fournies par un expert humain) représentées par des règles et une base de règles (est l'ensemble de connaissance du spécialiste qui lui permet de résoudre des problèmes spécifiques).
- **un moteur d'inférence** (mécanisme d'inférence) est un mécanisme qui permet au système expert de raisonner et tirer des conclusions.

5. Rôles

Les rôles des systèmes experts dépendent du contexte dans lequel ils sont appliqués dont notamment:

a. Pour la résolution de problèmes difficiles, ils permettent :

- D'agir efficacement et rapidement devant un problème pose.
- De tirer des conclusions à partir de relations complexes.
- D'expliquer le raisonnement.

b. Pour le niveau d'intervention, ils permettent :

- De remplacer un expert (pour automatiser une tâche routinière ou par besoin d'une expertise dans un environnement hostile).
- D'assister un expert (pour gérer la complexité ou pour améliorer la productivité).

c. Pour les autres tâches, ils permettent :

- De capter et préserver le savoir-faire.
- Rassembler et organiser une connaissance disséminée entre plusieurs experts.
- D'aider à la diffusion des connaissances.

6. Quelques définitions des mots clés

a) Système Expert : Application capable d'effectuer dans un domaine des raisonnements logiques comparables à ceux que feraient des experts humains de ce domaine. C'est avant tout un système d'aide à la décision.

b) Moteur d'inférence : Partie d'un système expert qui effectue la sélection et l'application des règles en vue de la résolution d'un problème donné.

c) Cognitique : La cognitique représente la partie appliquée des Sciences de la cognition et de la connaissance. Elle s'intéresse aux mécanismes de la pensée, au traitement de l'information par l'humain et aux systèmes anthropotechniques et propose des moyens de modélisation de ces connaissances, en permettant leur imitation, leur substitution ou leur aide par des moyens artificiels informatiques ou automatiques.

d) Cogniticien : Spécialiste des sciences de la pensée (sciences cognitives). Il travaille dans le domaine de l'intelligence artificielle.

e) Metadata/Méta-données : Les méta-données, ou données sur les données, renseignent sur la nature, les caractéristiques et la disponibilité des données. Elles rendent les données compréhensibles et partageables pour les utilisateurs dans le temps.

f) Metarules/Méta-règles : Règles contrôlant la sélection des règles à appliquer.

g) Expert : Professionnel qui connaît un domaine et qui est plus ou moins capable de transmettre ce qu'il sait : par exemple, ce n'est pas le cas d'un enfant par rapport à sa langue maternelle.

h) Connaissances incertaines : Données dont il est très difficile de déterminer les valeurs exactes. Elles demandent alors des quantificateurs pour signifier leurs valeurs et des probabilités pour les estimer. ***[6]***

7. Concepts généraux des systèmes experts

La connaissance d'un système expert peut être représentée de plusieurs manières (elle peut être encapsulée dans les règles et les objets). Une méthode commune de représenter la connaissance est sous forme des règles du type SI ... ALORS, par exemple : SI la lumière est rouge ALORS arrêtez-vous. Un système expert classique embarque une connaissance non écrite qui doit être obtenue d'un spécialiste à travers d'intenses interviews par un ingénieur de connaissance (cogniticien) pendant une longue période. Le processus de la construction d'u système expert s'appelle ingénierie de connaissance et consiste en l'acquisition de la connaissance auprès d'un spécialiste humain ou d'une autre source, et sa codification dans le système expert.

L'ingénieur de connaissance établit d'abord un dialogue avec un spécialiste pour obtenir sa connaissance. Ensuite, l'ingénieur de connaissance codifie explicitement la connaissance dans la base de connaissance sous forme des règles. Le spécialiste évalue alors le système expert et présente des critiques à l'ingénieur de connaissance. Ce processus se répète jusqu'à ce que l'expert juge satisfaisant le comportement du système expert.

Les systèmes experts sont souvent conçus de manière différente à celle des programmes conventionnels parce que les problèmes n'ont généralement pas une solution algorithmique et dépendent des inférences pour obtenir une solution raisonnable, en considérant la solution obtenue comme la meilleure à laquelle on puisse s'attendre s'il n'y a pas d'algorithme conduisant la solution optimale. Puisque le système expert dépend de l'inférence, il doit être capable d'expliquer son raisonnement pour qu'il soit vérifié. La facilité

D'explication fait partie intégrale des systèmes experts sophistiqués. Certains systèmes experts permettent même que le système apprenne des règles à travers l'exemple en utilisant l'induction des règles à partir des tables des données. ***[7]***

8. Caractéristiques d'un système expert

Un système expert peut être conçu pour qu'il ait les caractéristiques générales suivantes :

- ✓ **Haut rendement :** Le système doit avoir la capacité de répondre à un niveau de compétence égal ou supérieur à celui d'un spécialiste du domaine. Cela signifie que la qualité de conseil donné par un système doit être très haute.
- ✓ **Temps de réponse adéquat :** Le système doit agir en un temps raisonnable, comparable ou meilleur au temps exigé par un spécialiste, pour prendre une décision.
- ✓ **Fiabilité :** le système expert doit être fiable et ne doit pas connaître des "failles" sinon il ne sera pas utilisé.
- ✓ **Compréhensible :** le système doit être capable d'expliquer les étapes de son raisonnement pendant qu'elles s'exécutent, au lieu d'être seulement une boîte noire qui produit une réponse miraculeuse.
- ✓ **Flexibilité :** Vu la grande quantité de connaissance qu'un système expert peut avoir, il est important d'avoir un mécanisme efficient pour ajouter, modifier, et éliminer la connaissance. Une raison de la popularité des systèmes experts basés sur les règles est la capacité de stockage efficiente et modulaire des règles.

9. Avantages et inconvénients d'un système expert

1. Avantages d'un système expert

Les systèmes experts ont plusieurs caractéristiques attrayantes :

- ✓ **Grande disponibilité :** L'expérience est disponible pour tout matériel de traitement adéquat. Dans un sens plus réel, un système expert est la production massive de l'expérience.
- ✓ **Coût réduit :** Le coût de mettre l'expérience à la disposition de l'utilisateur est réduit énormément.
- ✓ **Danger réduit :** Les systèmes experts peuvent être utilisés dans des environnements qui pourraient être dangereux pour un être humain.
- ✓ **Permanence :** L'expérience est permanente. Contrairement aux spécialistes humains qui peuvent se retirer, renoncer ou mourir, la connaissance d'un système expert durera indéfiniment.
- ✓ **Expérience multiple :** La connaissance des plusieurs spécialistes peut être disponible pour travailler simultanément et continuellement sur un problème, à toute heure de la nuit ou du

jour. Le niveau d'expérience combinée de beaucoup de systèmes experts peut dépasser celui d'un seul spécialiste humain.

- ✓ **Explication :** Le système expert peut expliquer clairement et en détail le raisonnement qui conduit à une conclusion, cela augmente la confiance que la décision prise était correcte. Un être humain peut être fatigué, peut renoncer ou ne pas être capable de le faire toujours.
- ✓ **Réponse rapide :** Une réponse rapide ou en temps réel peut être nécessaire pour certaines applications. En fonction du logiciel ou matériel utilisé, un système expert peut répondre plus rapidement et être pus disposé qu'un spécialiste humain, de telle sorte qu'un système expert en temps réel constitue un bon choix.
- ✓ **Réponses solides, complètes et sans émotions, en tout moment :** Ceci peut être très important en temps réel et en des situations d'émergence, quand un spécialiste humain ne fonctionnera pas avec toute sa capacité à cause de la pression et de la fatigue.
- ✓ **Enseignement intelligent :** Un système expert peut agir comme un enseignant intelligent en laissant que l'étudiant exécute des programmes d'exemple et en expliquant le raisonnement du système.
- ✓ **Base de données intelligente :** Les systèmes experts peuvent être utilisés pour avoir accès à une base de données de manière intelligente.

2. Inconvénients d'un système expert

Les principaux inconvénients des systèmes experts sont :

- Dans les systèmes experts, on fait inférence à des connaissances même si elles sont dépassées. ***[8]***

10. Développement de la technologie des systèmes experts

L'IA a beaucoup de branches en rapport avec la parole, la robotique, la compréhension et l'apprentissage du langage naturel, les systèmes experts, etc. les racines des systèmes experts embarquent beaucoup de disciplines ; en particulier, une des racines principales des systèmes experts est le domaine de traitement de l'information humaine, appelé la science cognitive. La cognition est l'étude de la manière dont les hommes traitent l'information.

L'étude de la cognition est très importante si l'on prétend réussir des ordinateurs qui émulent les spécialistes humaines. Souvent, les spécialistes ne peuvent pas expliquer comment ils résolvent des

problèmes (simplement, les solutions leur arrivent). Si le spécialiste ne peut pas expliquer comment se résoudre un problème, il n'est pas possible de codifier la connaissance dans un système expert basé sur la connaissance explicite. Dans ce cas, l'unique possibilité est les programmes qui apprennent par eux-mêmes à émuler le spécialiste. Ce sont des programmes basés sur l'induction et sur les systèmes neuronaux artificiels.

11. Fonctionnement d'un système expert

Dans un premier temps, le système expert reçoit de l'expert toute les connaissances relatives au problème à gérer (règles, procédures, méthodes, stratégies,...). Celles-ci sont stockées dans sa base des faits. Ensuite, l'expert décrète au système expert les règles générales à suivre pour trouver de lui-même la solution à un problème posé (déductions, conclusion,...). Le cycle de fonctionnement d'un système se décompose en quartes phrases :

1. Interaction utilisateur-moteur : l'utilisateur demande de l'aide au moteur à partir d'une interface où il introduit les données à traiter.

2. Mémorisation des données : le moteur stocke les données dans la base de faits pour traitement.

3. Raisonnement : le moteur applique une stratégie de résolution de problème définie par les règles stockées dans la base de connaissance.

4. Communication de la solution : le moteur communique à l'utilisateur la meilleure solution au problème posé par l'intermédiaire de l'interface utilisateur et attend des nouvelles instructions.

11.1 Classification des systèmes experts

En ce qui concerne la classification, nous distinguons :

- Les systèmes expert à base de règles et
- Les systèmes experts par modèles. ***[9]***

11.2 Technologie des systèmes experts

Les experts humains sont capables d'effectuer un niveau élevé de raisonnement à cause de leur grande expérience et connaissance sur leurs domaines d'expertise. Un système expert utilise la connaissance correspondante à un domaine spécifique afin de fournir une performance comparable à l'expert Humain.

En général, les concepteurs de systèmes experts effectuent l'acquisition de connaissance grâce à un ou plusieurs interviews avec l'expert ou les experts du domaine. Les humains qui enrichissent le système avec leurs connaissances ne fournissent pas seulement leur connaissance théorique ou académique mais aussi des heuristiques qu'ils ont acquises grâce à l'utilisation de leurs connaissances.

Contrairement à la modélisation cognitive, les systèmes experts n'ont pas comme finalité de s'inspirer des théories du fonctionnement du cerveau humain mais ce sont des programmes qui utilisent des stratégies heuristiques pour la résolution des problèmes spécifiques.

Le raisonnement effectué par un système expert doit être objet à l'inspection, et ceci en fournissant d'information sur l'état de la résolution du problème et des explications sur les choix et les décisions du système.

D'un autre côté, la solution fournie par le système doit être évaluée par un expert humain et ceci dans le but de modifier l'information contenue dans la base de connaissances. ***[10]***

11.3 Structure d'un système expert

La structure d'un système expert est représentée par ci-dessous :

Moteur d'inférences : Un moteur d'inférence permet aux systèmes experts de conduire des raisonnements logiques et de dériver des conclusions à partir d'une base de faits et d'une base de connaissances.

Module d'acquisition : La méthode d'acquisition est une méthode qui s'applique à l'occasion de l'acquisition d'une société (dite filiale) par une autre société (société mère).

Moteur d'apprentissage : d'apprentissage moteur pour qualifier un ensemble de phénomènes cognitifs et moteurs qui s'acquièrent au long de la vie.

Base de faits : est l'une des entrées d'un moteur d'inférence. C'est un ensemble de connaissances appelées "faits" et considérés comme vrais.

Base de règles : Une base de connaissance regroupe des connaissances spécifiques à un domaine spécialisé donné, sous une forme exploitable par un ordinateur elle peut contenir des règles dans ce cas, parle de base de règles.

Métarègles : La métarègle correspond à la règle de niveau supérieur qui permet à un système à base de connaissances de déterminer quand il doit utiliser telle règle ordinaire plutôt que telle autre.

Base de connaissances : Une base de connaissance regroupe des connaissances spécifiques à un domaine spécialisé donné, sous une forme exploitable par un ordinateur. Elle peut contenir des règles (dans ce cas, on parle de base de règles), des faits ou d'autres représentations.

Module d'explication : est un le module Service continu/discontinu pour d'autres explications.

Expert : c'est une expérience reconnue lui permet d'apporter une réponse.

Cogniticien : le cogniticien est spécialisé dans l'intelligence humaine et artificielle.

Usagers : Un usager est une personne qui utilise un service public.

Selon ci-dessus, nous avons les organes ci-après :

- ✓ **La base de connaissances** contient les connaissances concernant la résolution du problème c'est-à-dire les faits, règles et méta-règles.
- ✓ Les faits : mémoire à court terme, dépendant du problème tandis que les règles : mémoire à long terme indépendant du problème, dépendant du domaine et enfin les méta-règles : mémoire à long terme indépendant du problème, voire du domaine. Une règle manipule des faits tandis qu'une méta-règle manipule des règles.
- ✓ **Le module d'acquisition** (appelé aussi interface expert) qui sert à emmagasiner l'expertise et les règles à suivre dans la base de connaissance.
- ✓ **Le module d'explication** permet au système expert d'expliquer son raisonnement à chaque conclusion, chaque action, ou chaque résultat du système expert après résolution.

Il est très important de remarquer la séparation faite entre les connaissances et l'inférence.

- Cette séparation permet d'utiliser un codage différent, cela nous permet par exemple d'utiliser le langage naturel pour représenter les connaissances (sous forme Si... ALORS... par example).
- Cette séparation permet au programmeur de se focaliser au codage des connaissances sans se soucier trop de la façon du codage du moteur d'inférence.
- Cette séparation permet aussi de modifier les connaissances sans avoir un effet sur le codage du moteur d'inférence.
- Cette séparation permet également de pouvoir tester plusieurs types d'inférence sur la même base de connaissances.

11.4 Les organes d'un système expert

Nous avons vu dans la constitution d'un système expert que la base de connaissance est composée de la base de faits et de la base de règles.

a) La base de faits

La base de faits est la mémoire de travail du système expert. Elle est variable au cours de l'exécution et vidée lorsque l'exécution se termine. Au début de la session, elle contient ce que l'on sait du cas avant toute intervention du moteur d'inférence.Puis elle est complétée par les faits déduits par le moteur d'inférence ou demandés à l'utilisateur. Par exemples, dans le domaine médical, la base de faits pourra contenir une liste de symptômes en début de session en et un diagnostic lorsque celle-ci se terminera.

- **Le type d'un fait**

Les faits peuvent prendre des formes plus au moins complexes.Nous envisagerons que des faits élémentaires dont les valeurs possibles sont :

- Booléennes: vrai, faux.
- Symboliques : c'est -à-dire appartenant à un domaine fini de symboles.
- Réelles : pour représenter les faits continus.

Par exemple, actif est un fait booléen, profession est un fait symbolique et rénumération est un fait réel.Un système expert qui n'utilise que des faits booléens est dit d'ordre 0. Un système expert qui utilise des faits symboliques ou réels, sans utiliser des variables, est d'ordre +0. Un système expert utilisant toute la puissance de la logique du premier ordre est d'ordre 1.

Exemple :

Considérons le fait suivant : le fait pour Mr. Paul d'être né à Kinshasa lui permet d'obtenir la nationalité congolaise.

Représentation d'ordre 0

Un fait est une proposition, donc a une valeur booléenne.

SI Paul né à Kinshasa

ALORS Paul congolais

Représentation en ordre 0+

Un fait est un couple (attribut, valeur) : AV

SI lieu naissance = Kinshasa

ALORS nationalité = Congolaise

Un fait est triplet (objet-valeur-attribut) : OAV

SI (Paul lieu naissance Kinshasa)

ALORS (Paul nationalité Congolaise)

Représentation d'ordre 1

Un fait est une expression symbolique sans variable

SI ($pers lieu naissance Kinshasa)

ALORS ($pers nationalité Congolaise)

- **Metafaits et metavaleurs**

Pour qu'un système expert puisse modéliser un raisonnement humain, il est indispensable qu'il puisse raisonner sur ses propres raisonnements, réfléchir aux faits qu'il manipule, aux formules qu'il peut construire, etc. Autrement dit, il n'est pas suffisant que le système ait des connaissances, il faut qu'il ait des métas connaissances.

Il faut par exemple qu'un système expert puisse savoir si une valeur a été attribuée à un fait. Dans la négative, cette valeur pourra être demandée à l'utilisateur. Mais si l'utilisateur ne peut pas répondre, il faudra que le système puisse le savoir afin de ne pas poser éternellement la même question. La seule manière d'attribuer une valeur à un tel fait sera alors de la déduire d'autres faits.

On dira que la valeur d'un fait est :

- ✓ Connue si une valeur lui a été attribuée
- ✓ Inconnue si aucune valeur lui à été attribuée et si aucune question à son sujet n'a été posée à l'utilisateur
- ✓ Indéterminée si le système ne lui a attribué aucune valeur et si l'utilisateur a répondu «je ne sais pas» à une question concernant sa valeur.

De manière analogue, tous les faits ne doivent pas faire l'objet d'une question à la personne concernée. Il n'est pas envisageable qu'un médecin demande à son patient : «quelle maladie avez-vous?»; ni qu'un juge demande à la personne comparaissent devant lui : «à quelle peine dois-je vous condamner?»

b) La base de règles

C'est le savoir-faire de l'expert sous la forme de règles de production. Le système a accès à cette base pour résoudre le problème en cours de traitement. La règle de production se présente sous la forme :

1. SI condition ALORS Action

2. Si c1 et c2 et... en ALORS A

- Conditions (encore appelées prémisses), teste l'appartenance des faits à la base des faits.

- Action, un effet sur la base de faits (ajout ou suppression d'un fait). Ne modifie pas la base des règles.

c) Le moteur d'inférence

Un moteur d'inférence est un mécanisme qui permet d'inférer des connaissances nouvelles à partir de la base de connaissance du système.

11.5 Domaines d'application

Les systèmes experts ont été conçus pour résoudre certains types de problèmes comme en médecine, en droit, en chimie, en éducation etc. Nous citons par exemples en:

- ✓ **Médecine** : MYCIN, pour aider à la prise de décision sur les infections bactériennes du sang et la médecine interne
- ✓ **Biologie** : CRYSTAL et MORGEN, un système expert capable de reproduire de protéines par analyse aux rayons X
- ✓ **Géologie** : PROSPECTOR, étude détaillée sur les fouilles des méthodes d'exploitations
- ✓ **Chimie** : DRENDAL, qui produit de représentations structurales en chimie organique à partir d'un spectrogramme de masse
- ✓ **Education** : GUIDON, presque pareil à MYCIN, c'est pour les étudiants en médecine afin d'enseigner l'emploi des règles de MYCIN pour les stratégies adéquates de diagnostic
- ✓ **Mathématique** : SNARK, qui traite des problèmes de calcul intégrale
- ✓ **Industrie** : JONATHAN, détecte des pannes industrielles

11.6 Problèmes adaptés aux systèmes experts

Les chercheurs ont défini un ensemble informel de critères pour déterminer si un problème est adapté ou non à être résolu par la technologie système expert :

1. Le besoin d'une solution doit justifier le coût et l'effort de la construction d'un système expert.

2. L'expertise humaine n'est pas valable dans toutes les situations dont on a besoin.

3. Le problème peut être résolu en utilisant une technique de raisonnement symbolique.

4. Le domaine est bien structuré.

5. Le problème ne peut pas être résolu en utilisant des méthodes traditionnelles de calcul.

6. La coopération entre experts de domaine existe.

7. Le problème est de taille considérable.

11.7 Processus d'ingénierie de connaissance

Les personnes concernées par le développement d'un système expert sont :

- ✓ l'ingénieur de connaissance qui est un expert en langage IA. Son rôle est de trouver les outils et les logiciels nécessaire pour l'accomplissement du projet, d'aider l'expert du domaine à expliciter sa connaissance et d'implanter cette connaissance dans la base de connaissances.
- ✓ l'expert du domaine qui fournit les connaissances nécessaires liées au problème.
- ✓ l'utilisateur final dont le rôle est de spécifier l'application et de déterminer les contraintes de la conception.

En général, le travail commence par un interview entre l'ingénieur de connaissance et l'expert du domaine. L'ingénieur essaie de comprendre le domaine, d'observer l'expert pendant son travail. Une fois l'expert ait obtenu des informations complètes et précises sur le domaine ainsi que sur la résolution du problème, il pourrait entamer la tâche de la conception du système.

Il choisit la façon de la représentation des connaissances, Il détermine le type du raisonnement et la stratégie utilisée (chaînage avant ou arrière, en profondeur ou en largeur). Il conçoit de même l'interface utilisateur. Le prototype obtenu doit être capable de résoudre correctement un problème typique. Ce prototype doit être testé et affiné par l'ingénieur et l'expert du domaine en même temps.

Le prototype peut être complété au fur et à mesure en ajoutant des nouveaux éléments dans la base de connaissance. Souvent, à la fin de ce travail progressif, l'ingénieur serait amené à refaire une version plus propre qui réécrit la connaissance d'une façon plus sommaire.

11.8 Acquisition de connaissance

Dans son rôle, l'ingénieur de connaissance doit traduire l'expertise informelle en un langage formel adapté au mode du raisonnement du système. Plusieurs points doivent être soulevés concernant l'acquisition des connaissances :

1. La compétence humaine n'est pas souvent accessible via la conscience. Avec l'expérience acquise, la compétence et la performance d'un expert s'installe et opère dans l'inconscient. Par conséquence, il est difficile aux experts d'expliciter son savoir-faire.

2. L'expertise humaine prend souvent la forme du *savoir comment* plus que la forme du *savoir quoi*.

3. L'expertise humaine représente un modèle individuel ou un modèle de communauté. Ces modèles sont soumis aux conventions et aux procédés sociaux.

4. L'expertise change et peut subir des reformulations radicales.

A cause de la complexité et de l'ambiguïté posée par le problème de l'acquisition de connaissances, l'ingénieur de connaissances doit avoir un modèle conceptuel lui permettant de faire la liaison entre l'expertise humaine et le langage de programmation, ce modèle constituera ce qu'on appellera représentation de connaissances.

11.9 Architecture logicielle

L'architecture logicielle d'un système expert peut être vue comme sur la figure 4 :

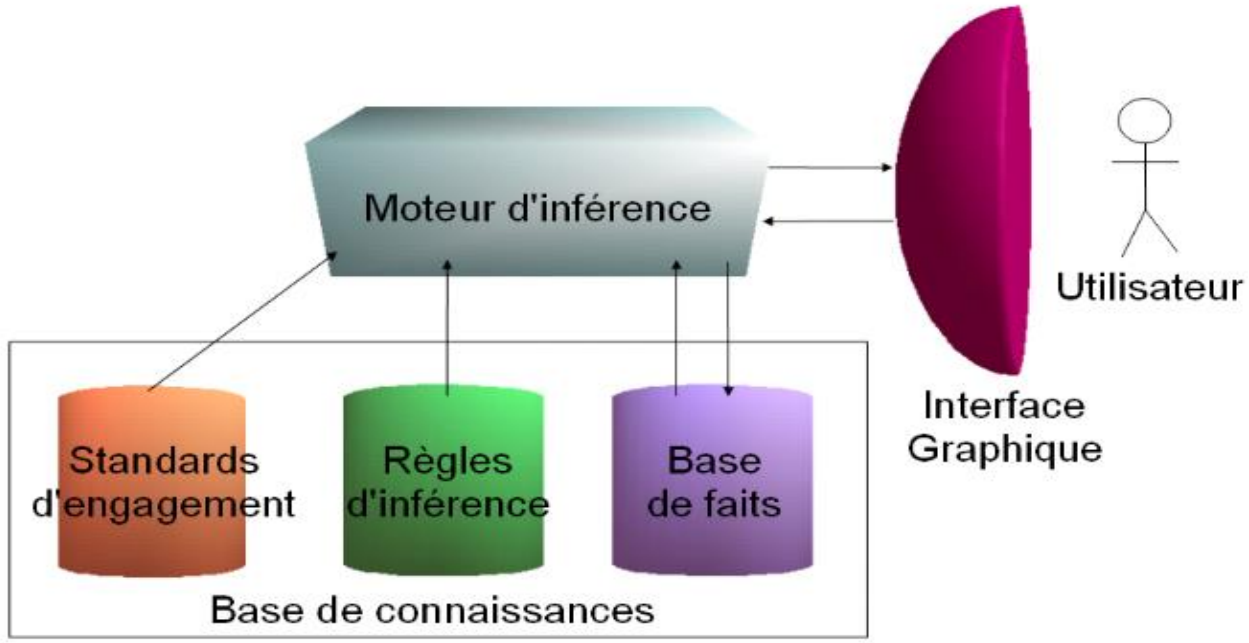


Figure 4: Architecture logicielle d'un système expert

Nous retrouvons dans ce schéma l'ensemble des composants détaillés dans la partie précédente : Le moteur d'inférence, la base de connaissances et l'interface graphique.

Chaînages avant/arrière

Le moteur d'inférence peut fonctionner selon 2 modèles : le chaînage avant et le chaînage arrière...

- ✓ **Chaînage avant**

Le principe du chaînage avant représenté dans la figure 5 ci-dessous est simple, il requiert l'accès aux prémisses (standards d'engagement) afin de déclencher les règles d'inférence adéquates définies par les metarules. L'application des règles (évaluations) donnent des résultats, ceux-ci sont évalués (par les metarules) afin de savoir si l'on a accédé à une solution finale potentielle. Si c'est le cas, on arrête et cette solution est proposée

· Si c'est le cas, la solution est proposée à l'utilisateur. S'il la valide, la solution est enregistrée dans la base de faits comme solution, sinon comme simple résultat et on continue dans le cas

suivant.

· Si cela n'est pas le cas ou si la solution est refusée, la solution est enregistrée dans la base de faits comme simple résultat et le moteur d'inférence tente d'y appliquer d'autres règles jusqu'à trouver une solution potentielle validée, ou jusqu'à ce qu'il n'y ait plus de règle.

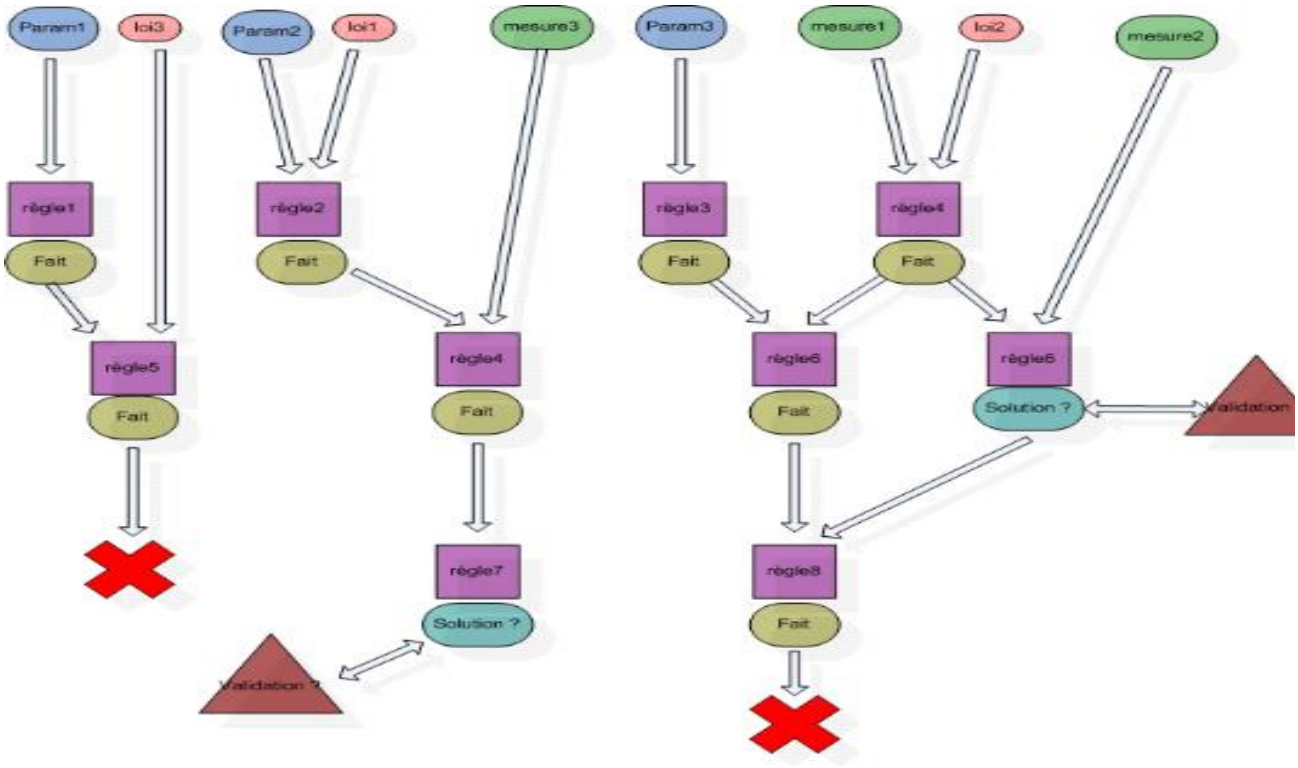


Figure 5 : Moteur d'inférence a chainage avant

✓ **Chaînage arrière**

Le principe du chaînage arrière représenté dans la figure 6 ci-dessous est plus compliqué, il s'agit dans ce cas de partir d'un effet ou d'une solution et de tenter de remonter la chaîne afin de déterminer les causes d'un effet (fait). La procédure est à partir d'un fait, de déterminer, grâce aux metarules, les règles d'inférence qui auraient pu être à l'origine de ce fait et de déterminer les paramètres les plus probables. A partir de là, on analyse les paramètres :

· Si le paramètre est un fait enregistré dans la base de faits, c'est qu'il est le résultat d'une règle (évaluation). La procédure précédemment décrite est donc relancée.

· Si le paramètre n'est pas un fait de la base de faits, on en reste là.

On relève alors tous les faits et données retrouvés. Ils représentent les causes probables de la conséquence étudiée.

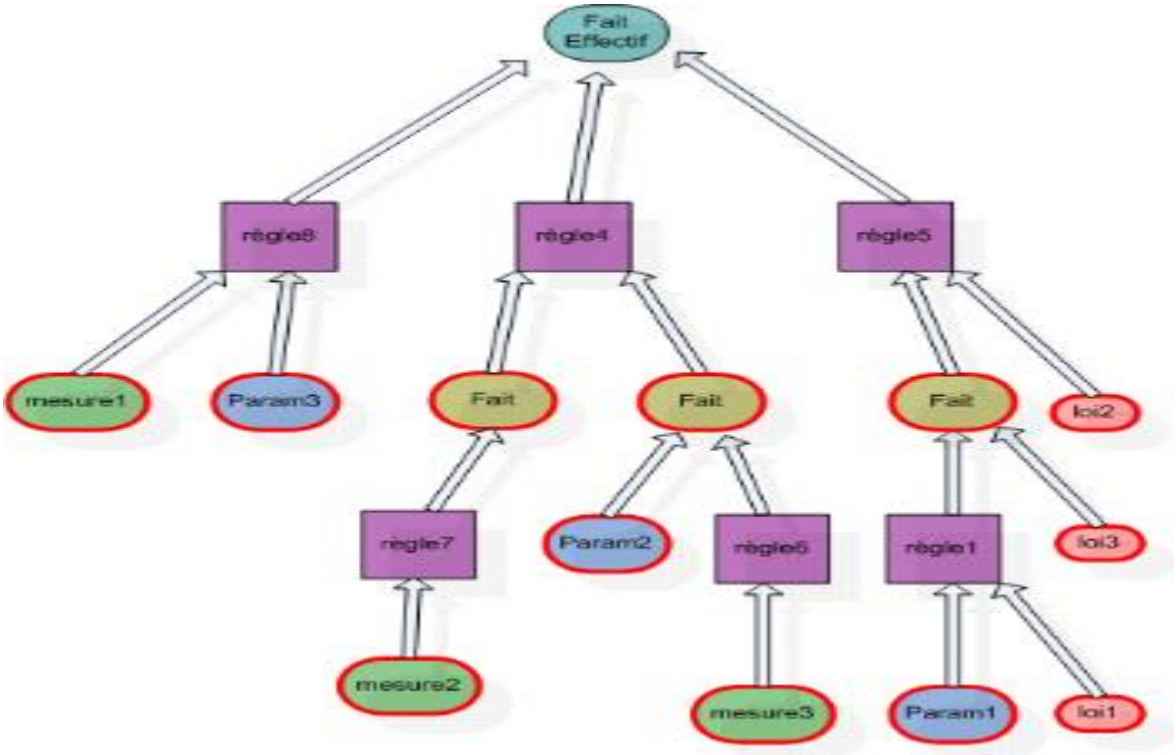


Figure 6: Moteur d'inférence a chainage arrière

Il existe un dernier mode de fonctionnement dit chaînage mixte qui combine les 2 chaînages précédents. De prime abord, il fonctionne comme le chaînage avant avec pour but de déduire un fait donnée. Mais applique un chaînage arrière sur chaque fait trouvé afin de déterminer les paramètres les plus probables et les plus optimisés. Ce mécanisme permet l'ouverture sur de nouvelles combinaisons encore non envisagées par les règles d'inférence et de déterminer les facteurs discriminants lors de la recherche d'une solution.

- ✓ **le chaînage mixte:**

Le chaînage mixte utilise le raisonnement inductif et le raisonnement déductif

Exemple

Base de connaissances :	**Base de faits :**
• **R1** : SI Tropiques ALORS Les Saintes • **R2** : SI Saint-Bart et hôtel ALORS Hôtel Paradisio • **R3** : SI dépressif ALORS Tourisme chaud • **R4** : SI tourisme chaud ALORS tropiques • **R5** : SI Les Saintes ALORS Hôtel Paradisio • **R6** : SI Les Saintes ALORS tourisme chaud • **R7** : SI P.D.G. ALORS tourisme chaud • **R8** : SI tourisme chaud et Les Saintes ALORS tourisme chaud et voilier • **R9** : SI Hôtel Paradisio ALORS Caraïbes	Les_Saintes

Base de faits état initial :

Les Saintes

Règle déclenchée **R5.**

Base de faits état 1 cycle :

Les_Saintes

Hôtel Paradisio

Règle déclenchée **R9**.

Base de faits état 2 cycles :

Les_Saintes

Hôtel Paradisio

Caraïbes

Le moteur est maintenant bloqué **en chaînage avant**. Il n'existe pas de prémisse qui contienne Caraïbes.

Il passe donc **en chaînage arrière** mais il n'existe pas de règle activable ayant une partie droite (conclusion) qui contienne Caraïbes. ***///***

Le moteur passe à nouveau **en chaînage avant**.

Les règles activables sont **R1, R2, R3, R4, R6, R7,R8**.

La règle R6 est sélectionnée et activée.

Elle permet de **déduire tourisme chaud**.

etc...

Graphe du chaînage mixte :

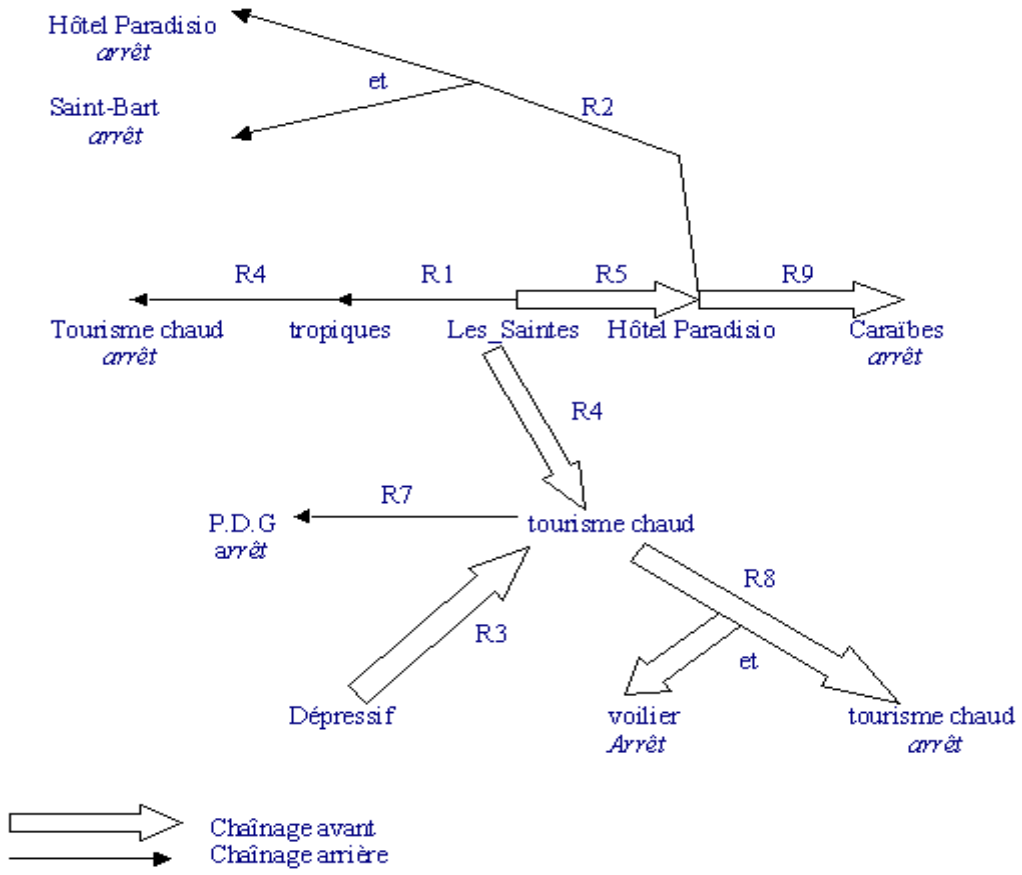


Figure 7: Moteur d'inférence a chainage mixte

12. Concluions

Les experts humains sont capables d'effectuer un niveau élevé de raisonnement à cause de leur grande expérience et connaissance sur leurs domaines d'expertise. Un système expert utilise la connaissance correspondante à un domaine spécifique afin de fournir une performance comparable à l'expert humain.

En général, les concepteurs de systèmes experts effectuent l'acquisition de connaissance grâce à un ou plusieurs interviews avec l'expert ou les experts du domaine. Les humains qui enrichissent le système avec leurs connaissances ne fournissent pas seulement leur connaissance théorique ou acadé- mique mais aussi des heuristiques qu'ils ont acquises grâce à l'utilisation de leurs connaissances.

Chaptier 03 : Des travaux existants

1. Introduction

Dans ce chapitre, qui mettait l'accent sur les travaux menés par des chercheurs dans notre domaine recherche, car ce type de recherche et de large, et nous avons choisi deux projets, qui à son tour gérer le problème des problèmes de base de données multi-niveaux.

2. Les travaux des chercheurs

Comme nous l'avons déjà dit, nous avons deux des travaux effectués sur deux comme suit :

2.1- Conception et mise en œuvre d'un Contrôleur d'inférence de base de données

Dans ce travail ***[12]*** Le problème d'inférence compromet des systèmes de bases de données généralement considérés comme sécurisés. Ici, les utilisateurs posent des séries de requêtes et déduisent des informations non autorisées des réponses qu'ils obtiennent. Un contrôleur d'inférence est un périphérique qui empêche et / ou détecte des violations de sécurité par inférence. Ce travail particulièrement intéressés par le problème d'inférence qui se produit dans un environnement d'exploitation multiniveau. Dans un tel environnement, les utilisateurs sont débloqués à différents niveaux de sécurité et ils accèdent à une base de données multiniveaux où les données sont classées à différents niveaux de sensibilité. Un système de gestion de base de données sécurisé multiniveaux (MLS / SGBD) gère une base de données multiniveaux où ses utilisateurs ne peuvent accéder aux données auxquelles ils ne sont pas autorisés.

Cependant, en fournissant une solution au problème d'inférence, où les utilisateurs émettent plusieurs requêtes et par conséquent, déduire que les connaissances non autorisées dépassent la capacité de MLS / SGBD disponible actuellement. Cet travail décrit la conception et le développement de prototypes d'un contrôleur d'inférence pour un MLS / SGBD qui fonctionne pendant le traitement des requêtes. À notre connaissance, c'est le premier prototype de contrôleur d'inférence de ce genre à être développé.

2.1.1- Gestion des inférences lors du traitement des requêtes

Ce travail décrit la conception dans le traitement des inférences lors de la requête en traitement. Il décrit d'abord une politique de sécurité pour gérer les inférences pendant le traitement des requêtes, puis discuter d'une approche pour la mise en œuvre de cette politique.

A- Politique de sécurité :

Ce travail, indique la politique de sécurité pour le traitement des requêtes

(1) Étant donné un niveau de sécurité L, E (L) est la base de connaissances associée à L. C'est-à-dire qu'E (L) consistera en toutes les réponses qui ont été publiées au niveau de sécurité L sur une certaine période et le monde réel Informations au niveau de sécurité L.

(2) Laissez un utilisateur U au niveau de sécurité L poser une requête. Ensuite, la réponse R à la requête sera diffusée à cet utilisateur si la condition suivante est satisfaite: F ou tous les niveaux de sécurité L * où L * domine L, si (E (L *) UNIONR) =>X (pour tout X) alors L * domine le niveau (X), où A => B signifie que B peut être déduit de A en utilisant l'une des stratégies d'inférence et le niveau (X) est le niveau de sécurité de X.

B- Mise en œuvre de la politique :

Dans cette section, il discute des techniques que il utilisée pour mettre en œuvre la politique de sécurité. Ils sont : la modification des requêtes et le traitement des réponses.

- Modification de la requête :

  La technique de modification de la requête a été utilisée dans le passé pour gérer la sécurité discrétionnaire. Cette technique a été étendue pour inclure la sécurité obligatoire. Dans notre conception du processeur de requêtes, cette technique sera utilisée par le moteur d'inférence pour modifier la requête en fonction des contraintes de sécurité, des réponses précédentes et des informations réelles. Lorsque la requête modifiée est posée, la réponse générée ne sera pas une violation de la sécurité.

- Traitement des réponses :

  Pour de nombreuses applications, en plus de la modification des requêtes, il faudra peut-être effectuer un traitement ultérieur de la réponse, comme la désinfection des réponses.

2.1.2- Conception de la mise en œuvre

- **Architecture de mise en œuvre :**

Sur l'architecture, la prochaine tâche était de sélectionner un système de base de données relationnel multiniveau pour la mise en œuvre. Après avoir étudié les différents systèmes disponibles, il sélectionné Secure SQL Server pour les raisons suivantes :

(1) le système était déjà disponible pour notre utilisation,

(2) Il eu des expériences de prototypage avec la version non-modulaire du SGBD relationnel de Sybase.

(3) le système a fourni les caractéristiques de sécurité de base dont il besoin.

Secure SQL Server s'exécute sur un système d'exploitation Microvax avec Ultrix.8 La conception du prototype suppose que le système d'exploitation est sécurisé à plusieurs niveaux9. Il permet de seize.

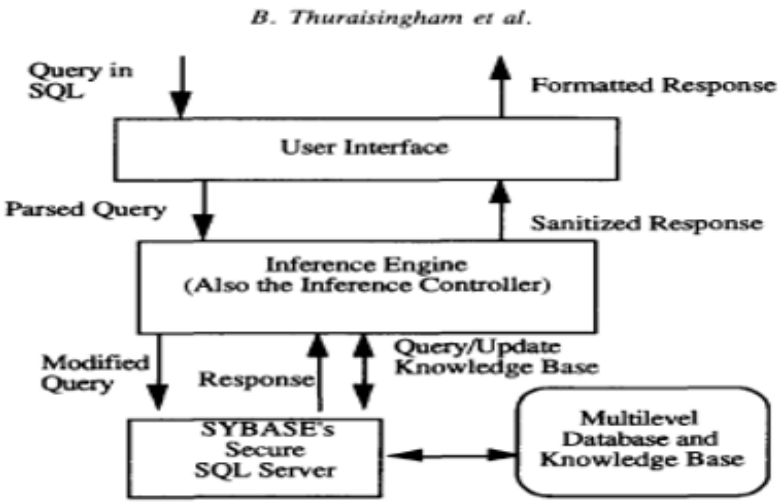


Figure 8:Implementation architecteur

2.1.3- Problèmes de représentation

Dans ce travail la conception les contraintes entrées par le SSO sont ensuite traitées par un module du contrôleur d'inférence et stockées.

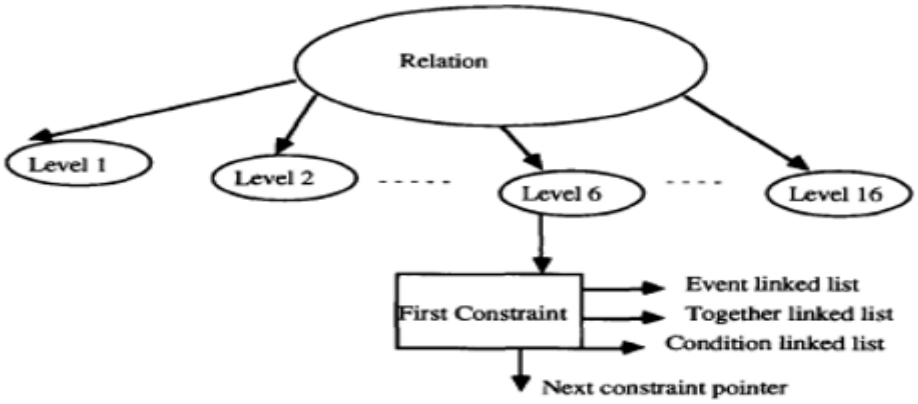


Figure 9:Constraint structure

2.1.4- Modules du processeur de requête

Ce processus demande le mot de passe et le niveau de sécurité de l'utilisateur. Étant donné qu'il suppose que le système d'exploitation est sécurisé, il utilise le mécanisme d'identification et d'authentification fourni par le système d'exploitation. En raison de cette fonctionnalité, P1 n'a pas besoin d'être un processus de confiance. Il fonctionne au niveau de l'utilisateur. P1 accepte une requête de l'utilisateur et effectue une vérification de syntaxe.

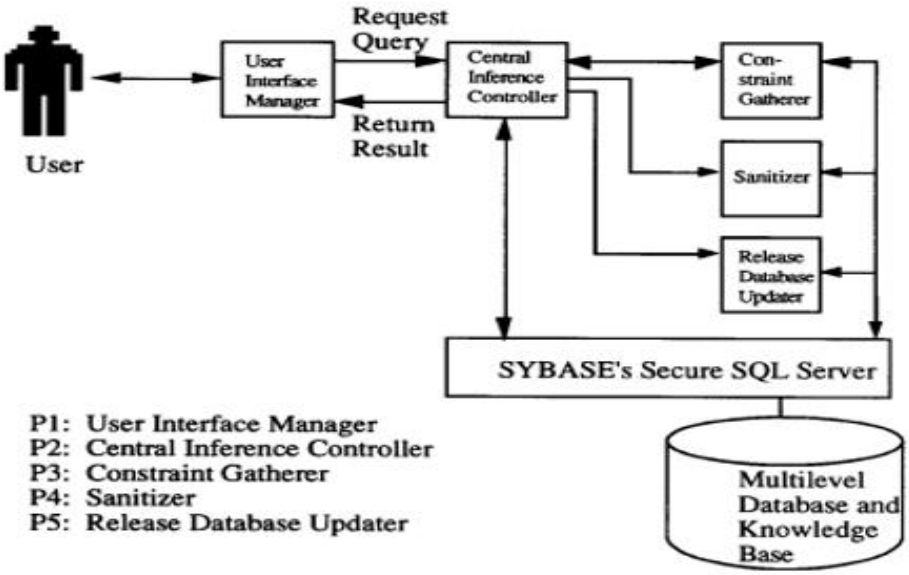


Figure 10:Major modules

2.2 - Utilisation de graphiques conceptuels pour représenter l'analyse de la sécurité par inférence de base de données

Dans ce travail ***[13]*** décrit une approche de l'analyse d'inférence de base de données basée sur des graphiques conceptuels. Le problème d'inférence de la base de données est brièvement décrit. Les approches précédentes sont résumées, suivie il présente de notre modèle d'inférence, appelé AERIE. Les notions d'une classe cible d'inférence et d'une classe de méthode d'inférence sont introduites avec des exemples donnés. Le graphique conceptuel il présente comme moyen de représenter la connaissance de l'inférence de base de données, comme première étape vers l'analyse et détectant les problèmes d'inférence de la base de données.

2.2.1-Approches à l'analyse d'inférence de base de données

L'inférence antérieure a été caractérisée dans les catégories suivantes :

1. "efforts pour découvrir des lois fondamentales qui déterminent si le potentiel d'inférences indésirables existe dans une base de données donnée.

2. efforts pour découvrir des règles d'inférence automatiques à partir de relations fondamentales entre les données appartenant à un domaine,

3. des efforts pour automatiser (via des systèmes experts) la procédure consistant à inférer des données sensibles dans un domaine spécifique. Cette liste peut être étendue avec une quatrième catégorie pour refléter un travail récent chez SRI International.
4. Efforts pour bloquer le canal d'inférence avec un «bruit» fourni par des récits de couverture plausibles.

2.2.2-AERIE Modèle d'inférence

Formulaire de chercheur d'inférence s'appelle AERIE, qui représente les effets d'inférence des activités, des entités et des relations. Pour il besoins, les entités (symbolisées par "E") sont des choses qui existent dans le monde et les activités (symbolisées par "A") sont des actions qui se déroulent dans le monde. Les relations peuvent être classées comme suit :

- ✓ Une relation entre deux entités, symbolisée par (E; E).
- ✓ Une relation entre une entité et une activité, symbolisée par (E ; A) ou (A ; E).
- ✓ Une relation entre deux activités, symbolisée par (A; A).
- ✓ Une relation entre deux relations ou plus, symbolisée par ((W 1 ; W2) ; (W3 ; W4)),
  Où chaque W i est soit «E» ou «A».

Le but d'un adversaire est d'apprendre des informations secrètes où il n'est pas facilement disponible. Nous nous référons à l'acquisition de cette connaissance comme "matérialisation" de l'information sensible.

Les informations à garder en secret seront désignées comme une cible de déférence. Les cibles d'inférence peuvent être catégorisées en classes de cible d'inférence en fonction des distinctions ci-dessus. L'approche AERIE utilise ces classes cibles pour classer les ensembles de méthodes d'inférence selon la cible qu'ils peuvent se matérialiser. Cette section décrit les classes cibles et leur relation entre elles.

2.2.3-Classes cibles d'inférence

Les classes cibles d'inférence identifient les types d'informations qui pourraient être la cible d'une attaque d'inférence. Les sept classes suivantes ont été identifiées :
Classe 1. E : La matérialisation d'une entité.
Classe 2. A : La matérialisation d'une activité.
Classe 3. (E ; E) : la matérialisation d'une relation sensible entre deux ou plusieurs entités matérialisées.

Classe 4. (A ; A) : la matérialisation d'une relation sensible entre deux ou plusieurs activités matérialisées.
Classe 5. (E ; A) ou (A ; E) : la matérialisation d'une relation sensible entre une ou plusieurs entités matérialisées et une ou plusieurs activités matérialisées.
Classe 6. (W1 ; W2) ; (W3 ; W4) ?: La matérialisation d'une relation sensible entre les relations sensibles.
Classe 7. ? C1 ; C2 ; Cn1 ?) Cn (où chaque C i est une matérialisation dans l'une des classes 1 à 6) : la matérialisation d'une règle sensible à partir des classes existantes.

2.2.4-Détection d'inférence à l'aide de graphiques conceptuels

La représentation de la connaissance choisie pour ce travail est celle des graphiques conceptuels. Les graphes conceptuels sont une notation orientée graphiquement basée sur la logique de premier ordre telle que décrite par les graphiques existentiels de Charles Peirce à partir de la fin des années 1800. Une extension des réseaux sémantiques, ils fournissent un moyen puissant et extensible de capturer les connaissances du monde réel, telles que la différence entre les types de classes et les instances d'une classe, les contraintes multiples sur un individu ou la classe et l'héritage des caractéristiques de type À partir d'un supertype. L'avantage d'utiliser des graphiques conceptuels est qu'ils permettent la modélisation de l'information sans exiger qu'elle soit codifiée dans les règles If-THEN; C'est-à-dire que la connaissance peut être appliquée de manière flexible au besoin.

2.2.5-Règles conceptuelles d'inférence graphique

Effacement : Dans un graphique négativement inclus, tout graphique peut être effacé, et tout concept généralisé. Cela est dû au fait que si quelque chose est vrai, sa généralisation est également vraie.
Insertion : Dans un contexte étrangement inclus, tout graphique peut être inséré, et tout graphique est restreint. Cela est dû au fait que si quelque chose est faux, sa spécialisation est également fausse.
Itération : Une copie de n'importe quel graphique peut être insérée dans n'importe quel contexte qu'il domine c'est-à-dire tout contexte imbriqué dans le contexte du graphique. Cela correspond approximativement à la règle A ^ (B) ? = A ^ (A ^ B).
Réitération : Tout graphique qui aurait pu être inséré au cours de l'itération peut être effacé.

Coreferent rejoindre : Deux concepts essentiels identiques dans le même contexte peuvent être joints. Ceci est une conséquence de la définition d'une ligne d'identité.
Personnes : Un concept générique peut être instancié si l'instance apparaît dans un contexte qui domine (c.-à-d.) Le concept générique. C'est parce qu'une contrainte imbriquée s'appliquera à toutes les instances qui la dominent.

3. Conclusion

Nous avons proposé ce chapter en tant que conclusion de recherché et de développements dans les bases de données d'inférence.

La première recherche se concentre sur les problèmes d'inférence et la stratégie d'inférence.

La deuxième recherche implique ou se concentre sur la représentation de la base de l'inférence de connaissances en utilisant les données cartographies.

À partir de ce résumé, nous avons bénéficié d'informations qui nous aident à terminer le quatrième chapitre, qui est la conception du contrôle d'inférence pour le base de donnée deductive.

Chapitre 04 : Conception

1. Introduction

Le mot ***inférence*** est généralement utilisé pour dire « former une conclusion à partir des premises ». Généralement, cette conclusion est crée sans une precedent approbation. L'information resultant qui se former peut être utilisée de manière innocue ou légitime ou elle peut être utilisée à des fins clandestines avec des harmonies sinistres menaçant la sécurité du système. Il est possible que l'utilisateur ait des ***informations sensibles*** (dont il n's pas le droit d'y accéder) à partir des données de la base de données du système. Cet accès aux informations sensibles ou informations non-autorisées présente un problème majeur sur la ***sécurité*** des données et des entreprises.

Dans ce chapitre nous nous sommes intéressés aux problems d'inférence dans une base de données multi niveaux où l'utilisateur doit faire face à différents niveaux de sécurité. Nous allons limiter et réduire l'accès aux différents types de données.

2. Architecteur du notre système

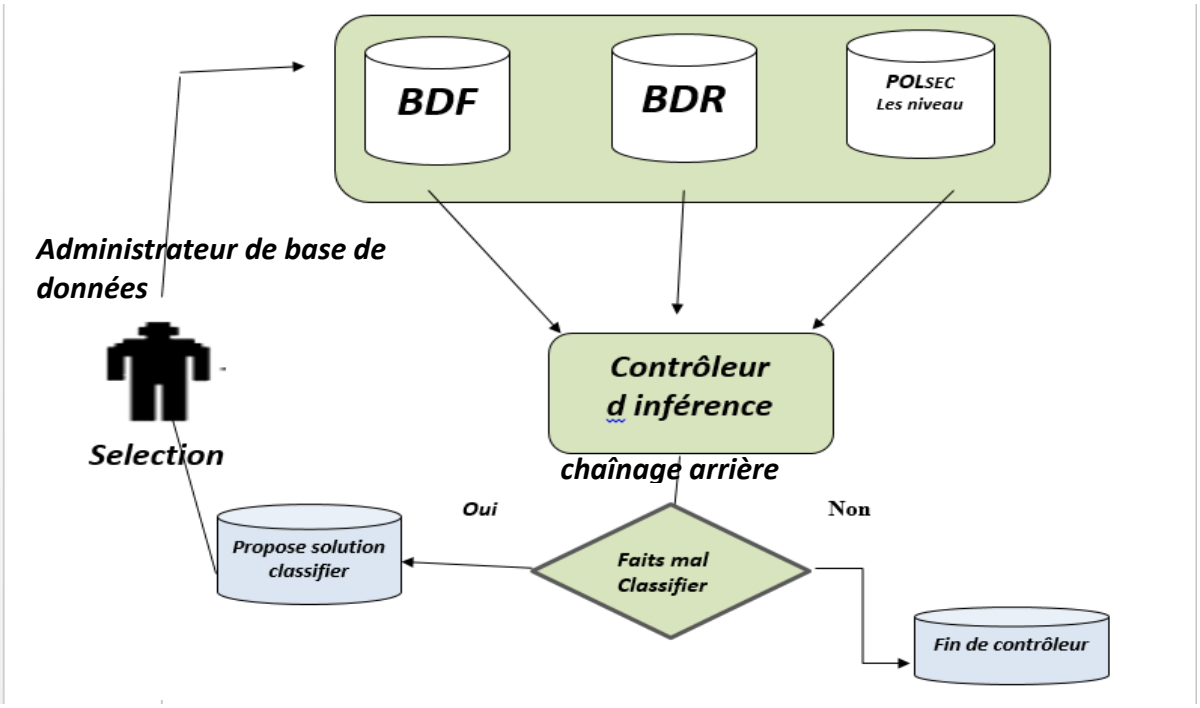


Figure 11: Architecteur du notre système

Dans notre travail nous allons devleppement un ***système expert***. Chaque système expert dispose de sa proper base de connaissance (ou Base de données). La base de connaissance contient la base de règles et la base de faits.

- **Base de Règles:** modélise la **connaissance** du domaine considerer.
- **Base de Faits** : contient les informations concernant le problème traité.

Vu plusieurs avantages nous allons adopter les ***bases de données deductive*** (détaillées dans le chapitre 1) pour la conception de notre ***système expert***.
Une base de données deductive est un système de base de données qui comprend des installations pour definer des ***règles*** deductive. Ces règles peuvent ensuite être utilisées pour inférer ou déduire des informations supplémentaires à partir des ***faits*** stockés dans la base de données (base de faits). Ce type de base de données supporte l'accès aux different niveaux (classes), ce qui n'offre pas ***une sécurité*** d'accès aux données qui peuvent être parfois ***confidentiel***. L'accès de l'utilisateur est déterminé au niveau de la class ou niveau accès. Une classification peut être aussi applique sur les different niveaux de la base de données comme: les relations, les attributes et les elements spécifiant chaque niveau.
Dans un système de base de données deductive, les règles sont généralement specifies dans un langage declarative, c'est-à-dire une langue qui fournit un mécanisme pour specifies ce que nous voulons atteindre plutôt que la façon dont nous voulons l'atteindre. Le modèle utilisé pour les bases de données deductive est basé sur le domaine de la programmation logique et le langage Prolog (expliqué plus en detail dans ce document).
Dans architecture propose nous avons une base de données deductive avec plusieurs niveaux. Chaque niveau posed ses propres faits et ses propres règles.
Nous envisages à travers notre proposition de specifies les informations de chaque niveau et affair un dirge de sécurité. De ce fait, l'utilisateur n'a accès qu'a son niveau ou le niveau inférieur d'une manière croissante ou décroissante (figure 12 ci-dessous):

- **AU ≤ N** ; où **AU : Accès Utilisateur, N : Niveau**

Par example:

- L'utilisateur A a accès au niveau N1 ;
- L'utilisateur B a accès au niveau Ni ;
- L'utilisateur C a accès au niveau Nn.

  Dans la base de données l'accès aux autres niveaux est comme suit:

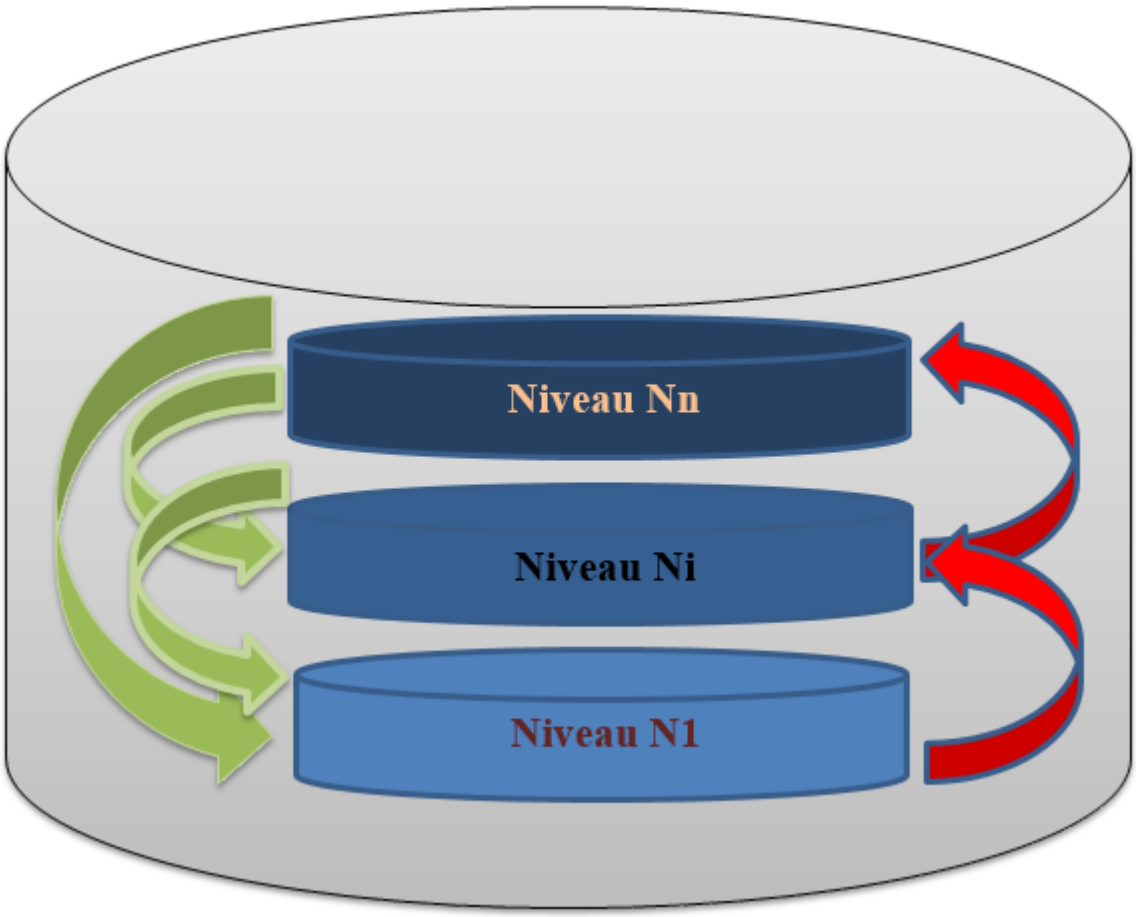


Figure 12: Architecture de base de donnée

On suppose généralement que ensemble des niveaux de sécurité former un réseau partiellement ordonné avec:

Non classifié < Confidentiel < Secret < Top Secret

Comme nous l'avons déjà mentionné, chaque niveau dispose de sa proper base de fait et base de règles.

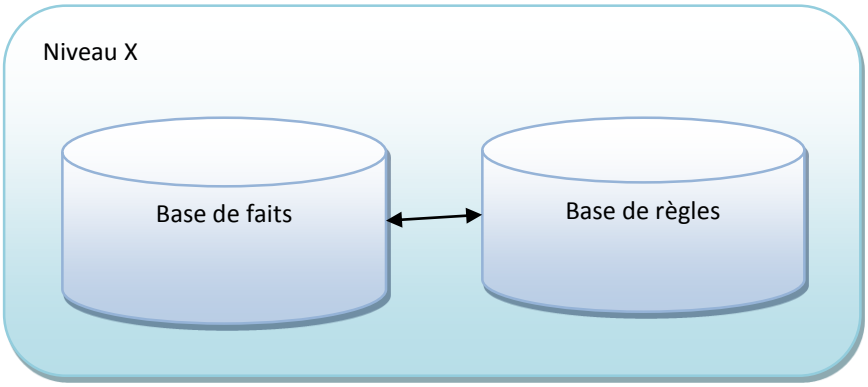


Figure 13: Base de faits et regle

3. Représentation des connaissances

Le choix d'un formalism pour la representation des connaissances se fait en general permit trios categories:

- ✓ ***Les formalisms logiques***
- ✓ Les réseaux sémantiques
- ✓ Les hiérarchies d'objets structurés.

Ces formalisms entrant dans le cadre des representations declaratives, mais, ils font apple à des representations procedurals lorsqu'il s'agit de maniple les connaissances declaratives pour trover une solution à un problème donné. Il arrive aussi qu'une partie des connaissances du domaine soit à caractère particulièrement procedural. Certains formalisms permettent, dans ce cas, de les incorporer dans la representation declarative choisie sous former d'attachement procedural.

Dans notre cas nous allons adopter le premier formalism qui est: ***Les formalisms logiques.***

Les formalisms logiques et surtout la logique des predicates joint un role important dans tout ce qui relive de la representation des connaissances et ceci est dû à plusieurs aspects de ces formalisms:

- ✓ D'abord, ils sont les premiers –chronologiquement plant- formalismes utilisés en IA notamment pour les applications de demonstration automatique de théorèmes ;
- ✓ Certains formalismes peuvent être vus comme une variant syntaxique d'une partie de la logique de prédicats ou l'une de ces extensions ;
- ✓ Ils sont des formalismes pavement syntaxiques dont la sémantique est rigoureusement définie ;
- ✓ Ils bénéficient d'une base mathématique solide en termes de mécanismes de raisonnement qui procèdent uniquement par manipulation symbolique.

Parmi les différentes logiques nous avons choisis la logique des prédicats.

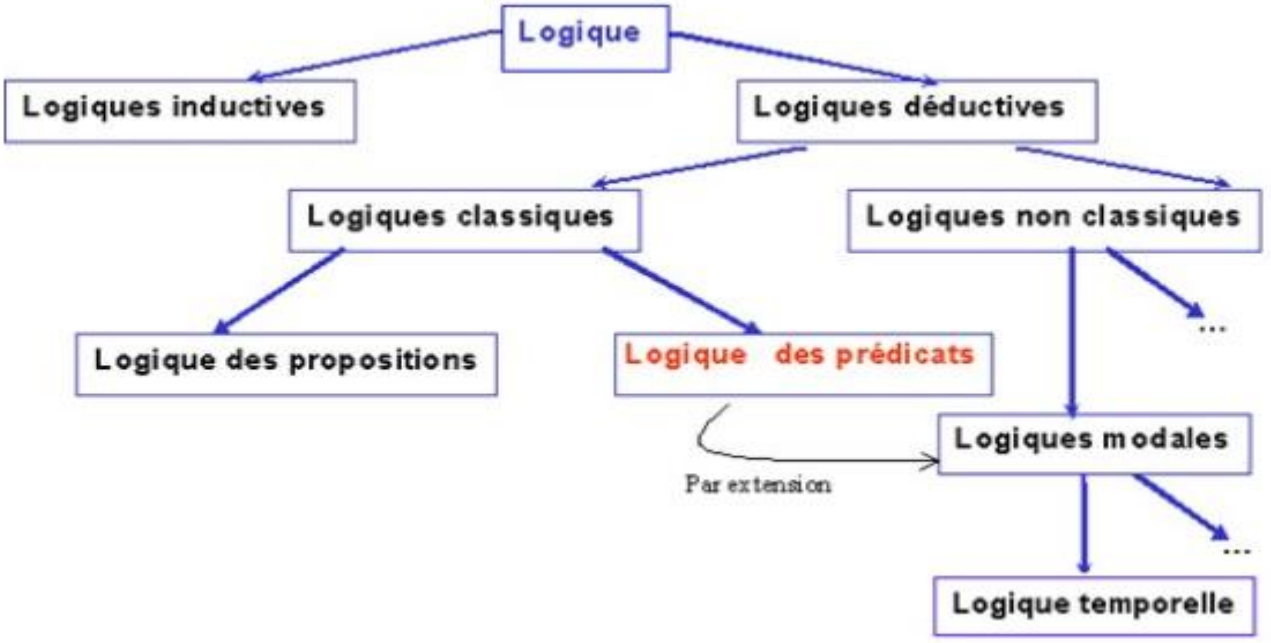


Figure 14: Représentation des connaissances

Logique des prédicats:

La logique des prédicats est le resultant des travaux de Free sur les systems formals vers la fin du 19ième siècle. Intelligence Artificial, a su en tier profit en l'utilisant comme un formalisme de représentation de connaissance dans de nombreuses applications où la logique propositionnelle se déclarait insuffisante. L'étude de la logique de prédicats comme un formalisme de représentation de connaissances comprend trios aspects:

- ✓ L'aspect syntaxique concerne la definition de la syntax correct des formules (dites bien formées) qui vont nous permettre d'écrire les énoncés concernant les faits du monde réel;
- ✓ L'aspect sémantique concerne le calcul des valeurs de vérité des formules qui représentent ces faits en se basant sur l'interprétation des différents elements constituent ces formules ;

L'aspect raisonnement s'occupe de l'établissement de preuves formelles de la validité des formules (et donc des faits qu'elles représentent) en se basant sur des règles d'inférence valides et des formules initiales valides. ***[14]***

4. Le moteur d'inférence

Dans notre travail nous visons d'ajouter une extension à cette approche qui consiste à autorisé le client ou l'utilisateur d'accéder aux niveaux inférieurs.

L'accès contrôlé et limité aux niveaux doit être effectué par ***un contrôleur d'inférence***. Une approche descendante (qui reste a confirmé) est également connue sous le nom de chaînage en arrière ou de résolution descendante. La figure 15 suivante représente architecture general de notre moteur d'inférence.

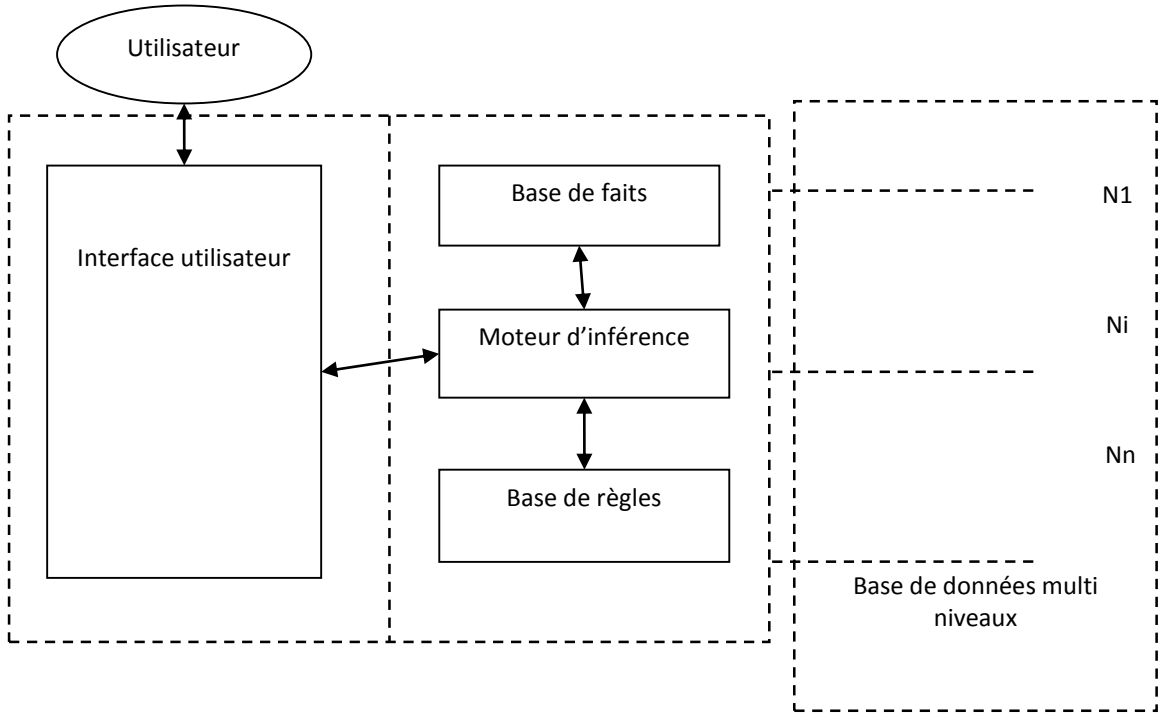


Figure 15: architecteur Le moteur d'inférence

- **<u>Politique de sécurité :</u>**

Dans cette partie nous allons decrier notre politique general de sécurité que nous allons applique à notre base de données.

a. Soit ***S*** le niveau de sécurité, ***BD(S)*** est la sécurité associée à la base de données. C'est-à-dire que ***BD(S)*** consistera toutes les réponses qui vont être diffusées sur le niveau de sécurité S sur une certaine période et information.

b. Soit un utilisateur ***U*** au niveau de sécurité ***S*** envoi une requête. Ensuite, la réponse ***R*** à la requête sera diffusée à cet utilisateur si la condition suivante est satisfaite:

- Pour tous les niveaux de sécurité ***S**** où ***S**** domine ***S***,
  Si ***(BD (S*) UNION R) ~ X*** (pour tout ***X***) alors ***S**** domine le niveau (***X***),
  Où A ~ B signifie que B peut être déduit de A en utilisant l'une des stratégies d'inférence et que le niveau (***X***) est son niveau de sécurité.

- **<u>Choix de la stratégie d'inférence:</u>**

Dans la partie précédente nous avons mentionné que, dans un système de base de données deductive, les règles sont généralement spécifiées dans un langage déclaratif qui est basé sur le domaine de la programmation logique et le langage Prolog.
L'inférence dans le domaine de la programmation logique et surtout le langage Prolog se base sur evaluation **top-down** (***<u>chaînage arrière</u>***) où l'ordre des règles est important [REF].
De ce fait, nous allons adopter le chainage en arrière (Detaille dans le chapitre 2). Dans notre processes, selon le principe du chaînage en arrière, nous allons construire un ***arbre d'induction (ET/OU)*** pour remonter aux causes selon l'algorithme suivant:

Chainage-arrière *(BF, BR, f)*
SI ***f*** appartient-à ***BF*** ALORS renvoyer *VRAI*
POUR TOUT ***r*** dans ***BR***, activable (***r***) = *VRAI* // Initialisation
conflit <- ∅ //Chercher les règles déclenchables
POUR TOUT ***r*** dans ***BR*** //Recherche des règles déclenchables
SI activable (***r***) = *VRAI* &&
f appartient-a Conclusions (***r***) &&
POUR TOUT ***p*** dans Prémisses (***r***)
p appartient-a ***BF*** &&
p non contradictoire avec ***BF***
ALORS AJOUTER ***r*** a ***conflit***
FIN POUR TOUT
SI ***conflit*** = ∅ ALORS renvoyer *FAUX* // Pas de règles pour prouver ***f***
r <- CHOISIR (***conflit***) // Point de choix
activable (***r***) = *FAUX* // Eviter les bouclages sur les règles
SOIT ***p1***, …, ***pn*** les Prémisses(***r***) // Appel recursive sur les premises de ***r***

renvoyer CHAINAGE-ARRIERE (***BF***, ***BR***, ***p1***) &…& CHAINAGE-ARRIERE (***BF***, ***BR***, ***pn***)

Cet algorithme va nous générer l'arbre d'induction suivante:

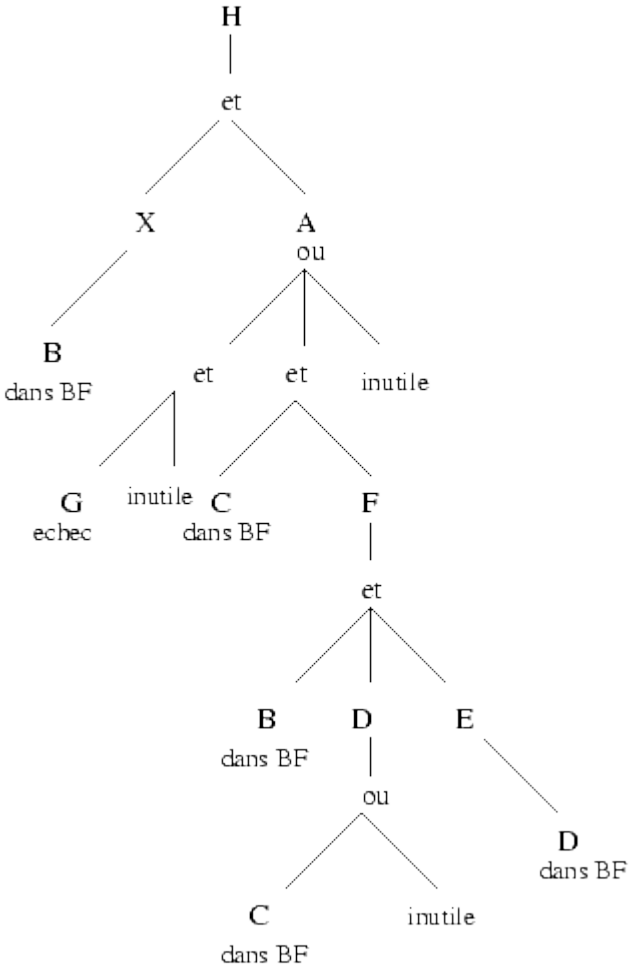


Figure 16: Chainage-arrière

5. Problème d inférences et Niveau de classification

Un problème d inférences il fait tout d'abord attribuer un niveau de classification à chaque fait déduit, pour cela on applique le principe suivant : soit *c*

P1 (a1) et P2 (a2) et .....Pn (an)

Une clause appartement à BDR (la Base de Règles) et soit *niv* le niveau de classification de cette clause, supposons qu'il existe une instance clause de *C* de la forme :

P1(A1) et P2(A2) et……..et Pn (An)

Tel que chacun des *Pi(Ai)* soit un fait appartement a BDF le fait *Q(B)* est déduit avec le niveau *niv'* de fini par : *niv '= lub (niv, niv1, niv2,……..nivn)*

Chacun de *nivi* représentent le niveau de classification de Pi(Ai),et *lub* est la formation calculant le bonne supérieur de l'ensemble des niveau des sécurité passés en paramètre.

Le niveau niv' ccerregrand à une classification implicite du fait Q(B)

Le fait Q(B) peut se trouver explicitement classification dans la base avec un niveau inférieur ou égale à niv' sans qu'il a de problème d'inférence.

Alors il y a un problème d'inférence dans la base si l'une des deux solutions suivantes existe.

1)-un fait est explicitement classification avec deux niveaux différents niv1 et niv2

Avec niv1<niv2 ou niv2<niv1.

2)-un fait explicitement classification a une (certain niveau niv1 et peut être également déduit avec une classification explicite niv2 tel que niv2<niv1.

6. l'algorithme de détection d'inférence

Préposé se dé compose en deux phases attribution un niveau de classification à chaque fait déduit

(F, D, F', D') tous ces ensemble sont constitués de couples (f, niv)

F : est ensemble de tous les faits déduits.

D : est ensemble de nouveaux faits déduits par dernière étape.

On définir aussi :

NF : l'ensemble de nouveaux faits déduits à partir de l'ensemble F.

FMC : l'ensemble de faits male classifier, il s'agit de tous les faits qu' permet être déduits avec deux classification diffèrent niv, et niv' .

On à alg ANCFD (F, D)= (F', D') si et si

$$\begin{cases} F' = (F \cup NF) - FMC \\ D' = NF - FMC \end{cases}$$

NF et FMC sont définis de la forme suivante.

NF= (f, niv) tel que :

Il existe une instance d'une clause (c, niv) de BDR

Ou c'est la forme : f1 et f2 et ……et Fn reçoit f

Et niv= lub (niv1, niv2, ……. Niv n)

FMC = (f, niv) tel que :$(f, niv) \in (F \cup NF)$ et $niv' = (f, niv') \in (F \cup NF)$ et $niv' < niv$.

- ✓ **<u>De la algorithme ANCFDn</u>**

Consiste alors à calculer n fois la fonction ANCF sur ensemble BDF c'est dire ANCFN (BDF, DBR) avec n tel que

ANCFNn (BDF, BDF)=< F, Q.

S'arrête donc lorsque la fonction ANCFN ne permet plus de générer (…) faits élémentaire.

7. Elémination des inférences non autorisé

L'orsque problème d'inférence déduction plusieurs solution se présente pour y remédier.

1er.sol : dé classification.

2émes.sol : sur classification au certain engouement alors il présenter l'ensemble P a l'administrateur de BDD pour décode de décalcification ou sur classification.

- ✓ Après la décision de l'administrateur et la modification la base de données en conséquence de la décision ; il faut contrôler autre fois la base de donnée jusque l'absences des fait mal classifiées.

8. Conclusion

A la fin de ce chapitre, nous concluons que un moteur d'inférence est un système logiciel qui est conçu pour tirer des conclusions en analysant les problèmes à la lumière d'une base de données de connaissances d'experts, il se appuie sur Il atteint des résultats logiques fondées sur les données les locaux établit.

Parfois inférence moteurs sont également capables d'aller au-delà de traitement logique stricte, et utilisent des calculs de probabilité de parvenir à des conclusions que la base de connaissances ne fonctionne pas strictement, mais implique ou borne à suggérer.

Chapitre 05 : Implémentation

1. Introduction

Dans le chapitre precedent de ce memoir, nous avons propose une architecture pour la conception de notre système, pour cela nous avons définit une politique de maintenance et nous avons choisis le chainage en arrière comme stratégie d'inférence. Afin d'illustrer les différentes idées et concepts inclus dans architecture proposée, nous allons utiliser cette architecture comme base pour une etude de cas dans un environment réel. Le but est de dérouler les principaux aspects de notre architecture sur un example concret afin de montrer la faisabilité et la mise en évidence de nos idées.

Pour cela, nous procédons comme suit : nous commençons par limiter le cadre de notre application. Par la suite, nous décrivons brièvement la plate-forme que nous avons adoptée pour l'implémentation de notre architecture, pour montrer comment nous l'exploitons dans le cadre de notre travail. Les résultats obtenus à partir de l'implémentation de notre étude de cas sont présentés dans la fin de ce chapitre.

2. Présentation de l'étude de cas

Afin de montrer la validité, la fiabilité et l'extensibilité de notre architecture, nous avons réalisé une étude de cas. D'où nous allons appliquer notre approche sur une clinique médicale.
Dans notre système nous avons trois niveaux (figure 17).
Le premier est celui d'un client où le client connecté pourra avoir des informations concernant la clinique ou les patients de cette dernière. Le client ne pourra pas avoir aucune information concernant les personnels ni les médecins.
Le deuxième niveau, concerne les personnels de la clinique. Les personnels auront accès à leur niveau et au niveau inférieur celui des clients. La base de fait et de règle de ce niveau est privée à ce niveau ; comme décrit dans le chapitre précédent chaque niveau possède sa proper base de fait et base de règle. La base de fait de ce niveau contient les numéros d'immatriculation de chaque personnel et leurs noms.
Le troisième niveau est celui des médecins, la base de fait de ce niveau contient elle aussi leurs immatriculations et leurs noms. Le médecin sur ce niveau aura accès aux deux autres niveaux inférieurs.

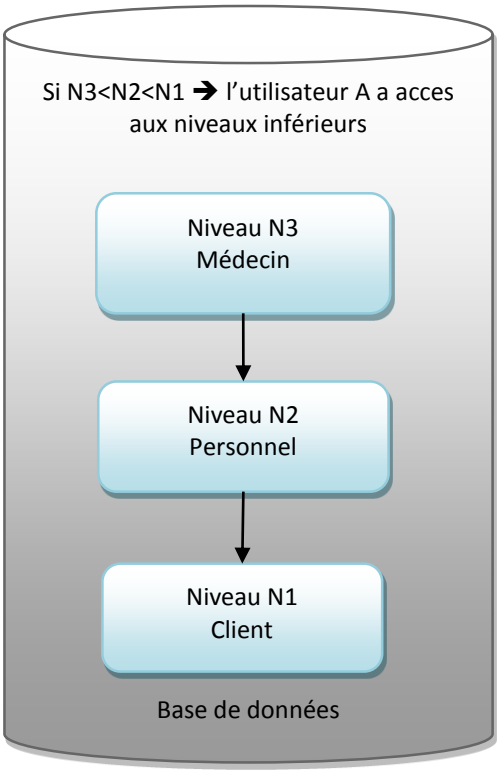


Figure 17: Présentation de l'étude de cas

npgm.pl

File Edit Browse Compile Prolog Pce Help

npgm.pl

```
%Prolog fonctionne automatiquement en chainage en arrière
%le debut ?- clinique.
%
%
%
%Base de fait qui veux dire
symp(fiévre). %la fièvre est un symptome
symp(nosée). % nosée est un symptome
symp(fatigue). %la fatigue est un symptome
symp(mal_à_la_tète). %mal_à_la_tète est un symptome
symp(eruption_cutanée). %eruption_cutanée est un symptome
symp(des_frissons). %frissons est un symptome
symp(mal_au_corps).%mal au corps est un symptome
symp(saignement). %saignement est un symptome
symp(nez_qui_coule). %nez qui coule est un symptome
symp(conjonctivite). %conjonctivite est un symptome
symp(éternuements).%éternuements est un symptome
symp(la_toux). %la toux est un symptome
symp(glandes_enflées).%glandes_enflées est un symptome
symp(maux_de_gorge).%maux_de_gorge est un symptome
%symp(rhume).
```

Figure 18: Base de faits

```
%%%%  %Niveau 2 les faits veux dire
acc(jamal). % jamal a accès au niveau 2 ====> c'est une relation entre la personne et
 l'accès
acc(yacine). % yacine  a accès au niveau 2 ====> c'est une relation entre la personne
 et l'accès
acc(sami). % sami  a accès au niveau 2 ====> c'est une relation entre la personne et
l'accès
acc(ali). % ali a accès au niveau 2 ====> c'est une relation entre la personne et l'a
ccès


app(n001,n2). % le matricule n001 appartient au niveau 2
app(n002,n2).  % le matricule n002 appartient au niveau 2
app(n003,n2). % le matricule n003 appartient au niveau 2
app(n004,n2). % le matricule n004 appartient au niveau 2
```

Figure 19 : Base de faits niveau 2

```
%!      %% Niveau 3
mcode(m1,n3). % le matricule m1 appartient au niveau 3
mcode(m2,n3). % le matricule m2 appartient au niveau 3
mcode(m3,n3). % le matricule m3 appartient au niveau 3
mcode(m4,n3). % le matricule m4 appartient au niveau 3

%
med(aimen). % aimen est un médecin
med(wahab). % wahab est un médecin
med(charaf). % charaf est un médecin
med(amin). % amin est un médecin
```

Figure 20 : Base de faits niveau 3

```
regle1(T):-
        (   (T == '1')-> writeln('Niveau 1'),hypo(niveau1));
        % d'ici nous commençons le chainage en arrière pour appliquer la règle
        (   (T == '2')-> writeln('Niveau 2'),hypo(niveau2));
        % d'ici nous commençons le chainage en arrière pour appliquer la règle
        (   (T == '3')-> writeln('Niveau 3'),hypo(niveau3));
        % d'ici nous commençons le chainage en arrière pour appliquer la règle
        (   writeln('Fausse demande'),
        erreur).

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Hypothesis that should be tested (!) inside a rule will prevent Prolog
% from backtracking any predicates behind the cut

hypo(niveau1):-niveau1,!. % cette règle veux dire si l'utilisateur a choisis le numér
o 1 il va va aller au niveau 1

hypo(niveau2):-niveau2,!. %% cette règle veux dire si l'utilisateur a choisis le numé
ro 2 il va va aller au niveau 2

hypo(niveau3):-niveau3,!. %% cette règle veux dire si l'utilisateur a choisis le numé
ro 2 il va va aller au niveau 2
hypo(message):-erreur,!. /* aucune informations*/
```

Figure 21 : Base de regle

3. Environnement de développement

a. Langages de programmation

Prolog

Prolog (Programmation Logique) est un langage de programmation basé sur la logique du premier order, il a été inventé au début des années 70 par Alain Colmerauer à Marseille dans le but de pouvoir faire le traitement de la langue naturelle, mais il s'est vite aperçu que ce langage pouvait avoir un champ application beaucoup plus large. ***[15]***

Le langage Prolog effectue un raisonnement guidé par les buts (Chainage arrière) et son principe de résolution (par SLD-résolution) va avec la stratégie en profond d'abord.

Chaque étape de résolution doit prendre une clause négative (initialement, la question) et une clause définie (prise dans le programmer):

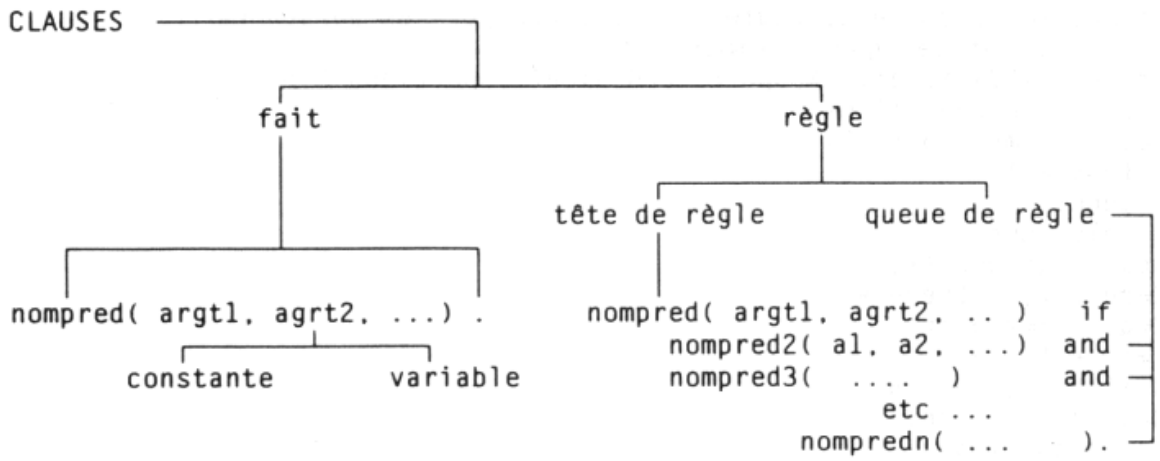
SLD-resolution (*Linear resolution for Definite clauses with Selection function*):

Prendre un littéral de la clause négative (lequel?) et tenter une unification avec le littéral positive d'une clause définie (**unification le plus general**)

Si une telle unification est trover alors remplacer le littéral choisi de la clause négative par les éventuels littéraux negatives de la clause définie qui a réussi l'unification (*Linear*)

Si l'unification échoue, reporter cet échec à l'unification de niveau supérieur.

Si la clause négative est vide: succès!



Version choisis de Prolog:

Parmi plusieurs versions de Prolog, nous avons choisis de faire notre implementation sur la version ***SWI-Prolog*** (figure 22). Cette dernière que nous avons choisie est une version développement SWI-Prolog **7.5.4** for Microsoft Windows (64 bit).

Notre choix est motivé par les points suivants:

Fonctionne sous Linux, Windows et MacOS.

Disponible gratuitement (licence GPL) au département informatique de l'Université de Psychologie d'Amsterdam http://www.swi-prolog.org/download/stable

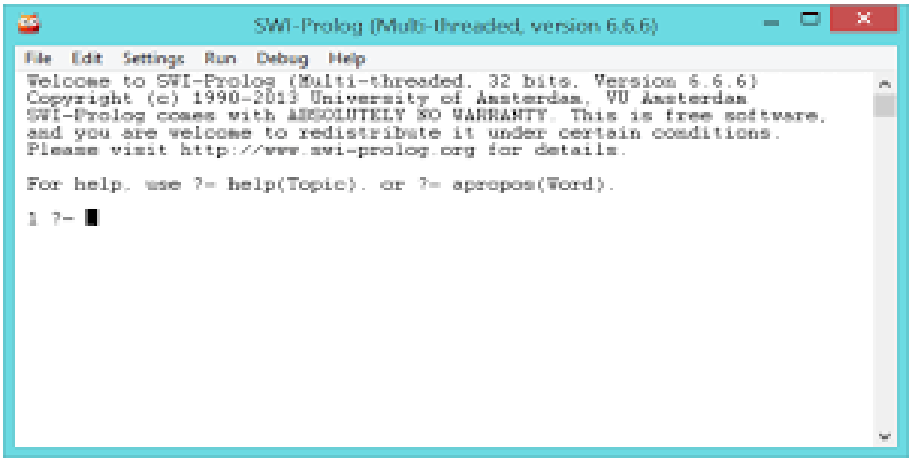


Figure 22: Interface SWI-Prolog

Dans la version que nous avons choisie la connexion avec java peut être facilement effectuée. La bibliothèque XPCE a été créée en 1985 pour fournir une interface graphique à SWI-Prolog. XPCE est très facilement utilisable avec Prolog mais peut-être utilisé avec d'autres langages (Lisp par exemple). Les codes XPCE sont multiplateformes Unix/Windows.

XPCE a été redésigné par la compagnie SUN (java) [REF]. XPCE est distribué avec le système SWI-Prolog, c'est une bibliothèque de base, aucune installation spéciale n'est requise.

4. Résultats expérimentaux

Afin d'exécuter notre programmer il va falloir taper le mot « clinique. » dans la console Prolog. En deuxième temps l'interface suivante s'affiche.

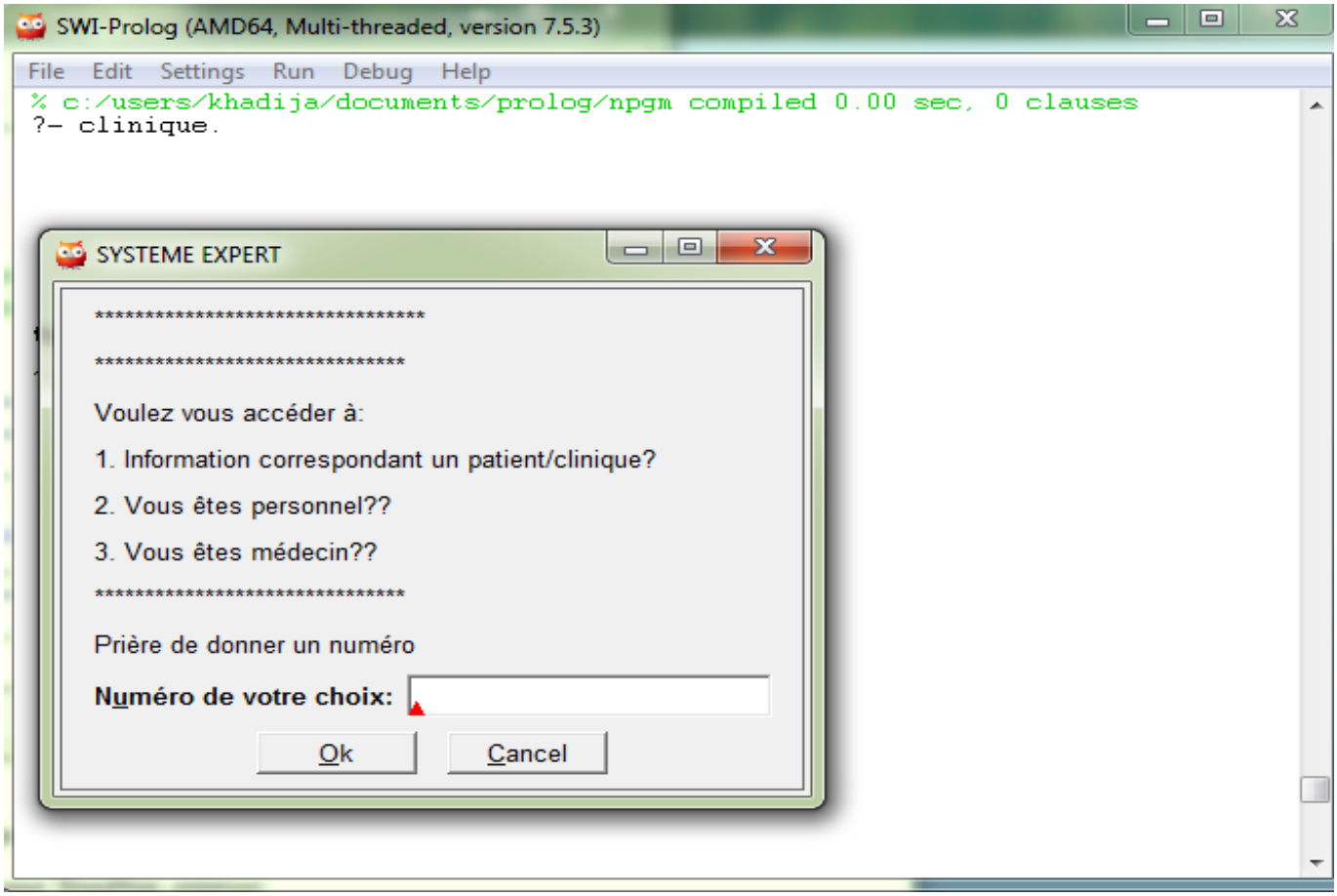


Figure 23: Résultats expérimentaux

L'utilisateur doit choisir un numéro correspondant son choix. Si l'utilisateur se trompe de numéro la fenêtre suivante s'affiche avec une fenêtre erreur:

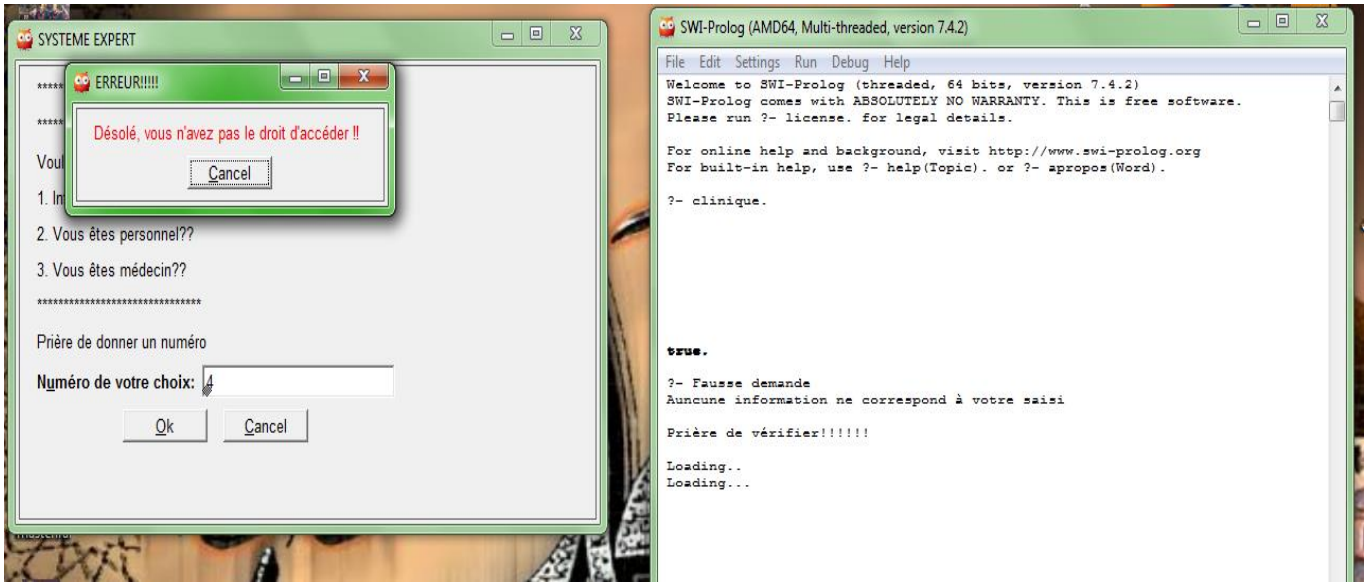


Figure 24: les message d'erreurs

Dans le cas ou l'utilisateur est un client (extérieur) une fenêtre s'affiche ou le client va pouvoir consulter les informations concernant la clinique ou les différents clients de la clinique.

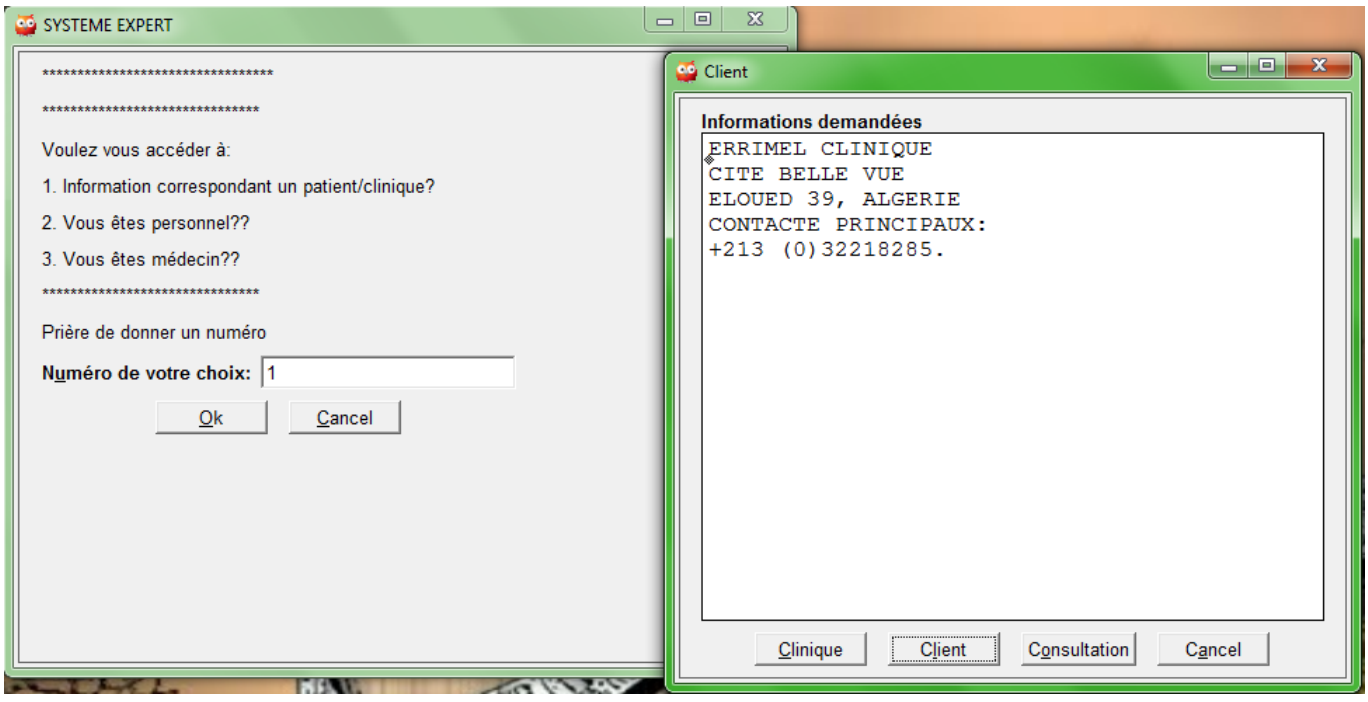


Figure 25: clients de la clinique

Dans le cas où l'utilisateur est une personnel dans la clinique sera une fenêtre pour entrer dans son secret de nom et le numéro.

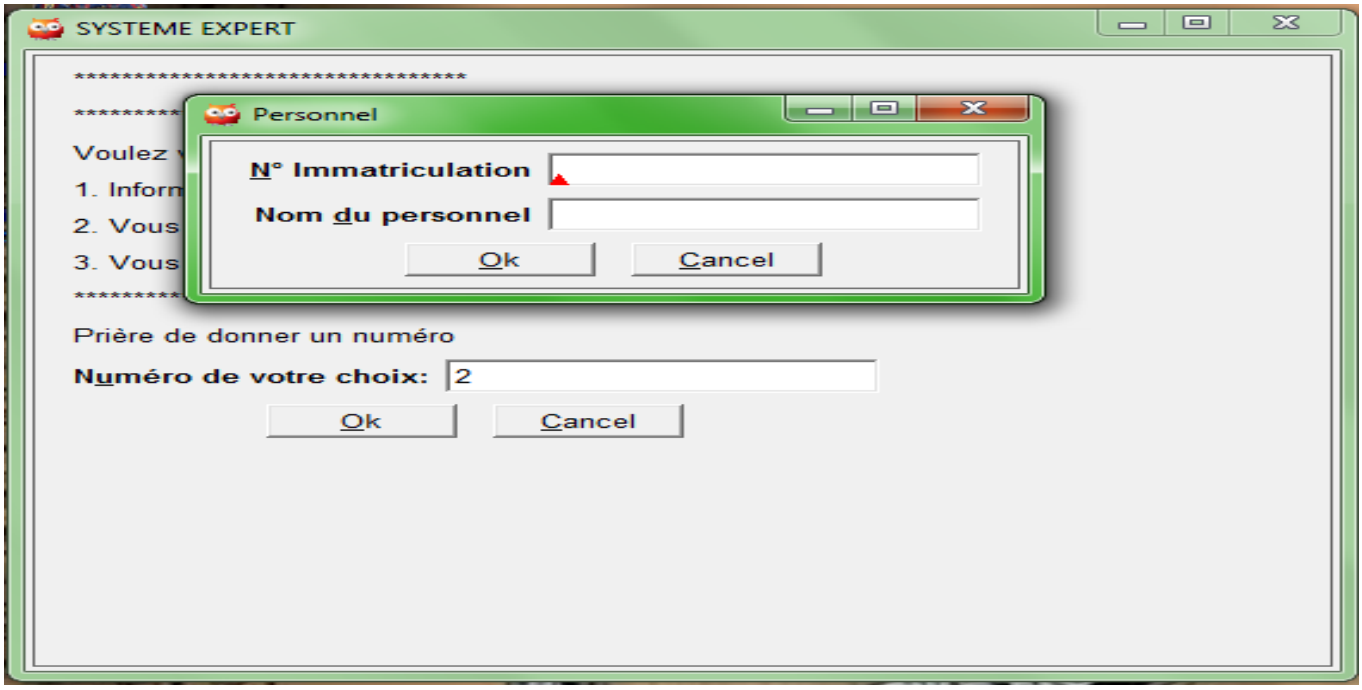


Figure 26:personnel de la clinique

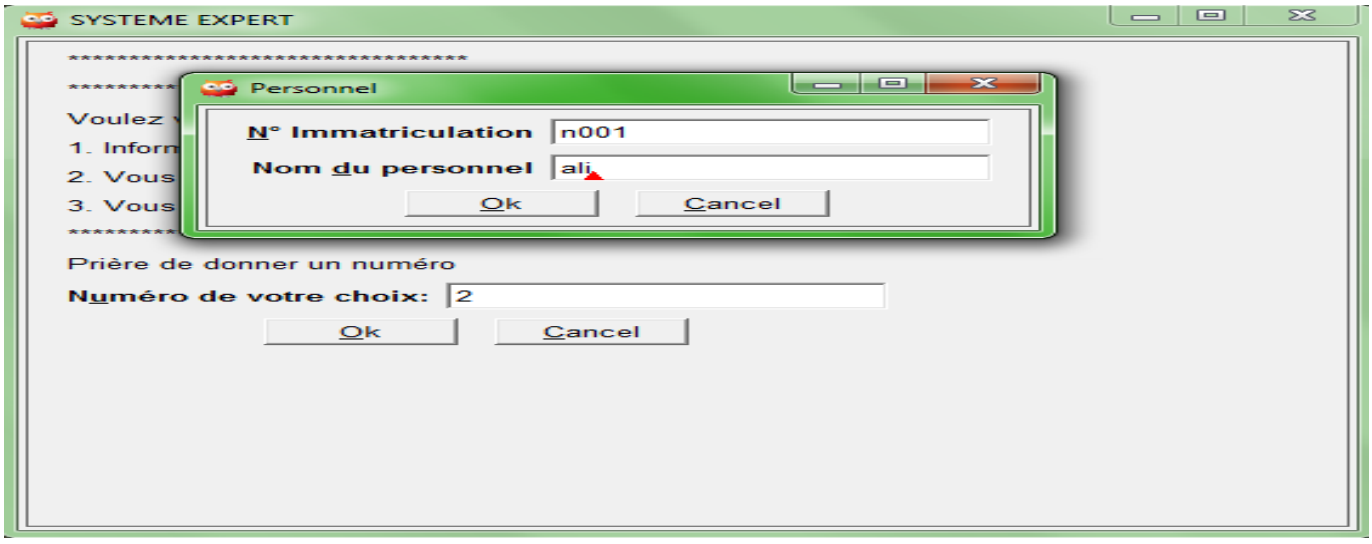


Figure 27:La vie privée du personnel de la clinique

Lors de l'entrée des personnel se trouvent sur leurs informations ainsi que les clients privés.

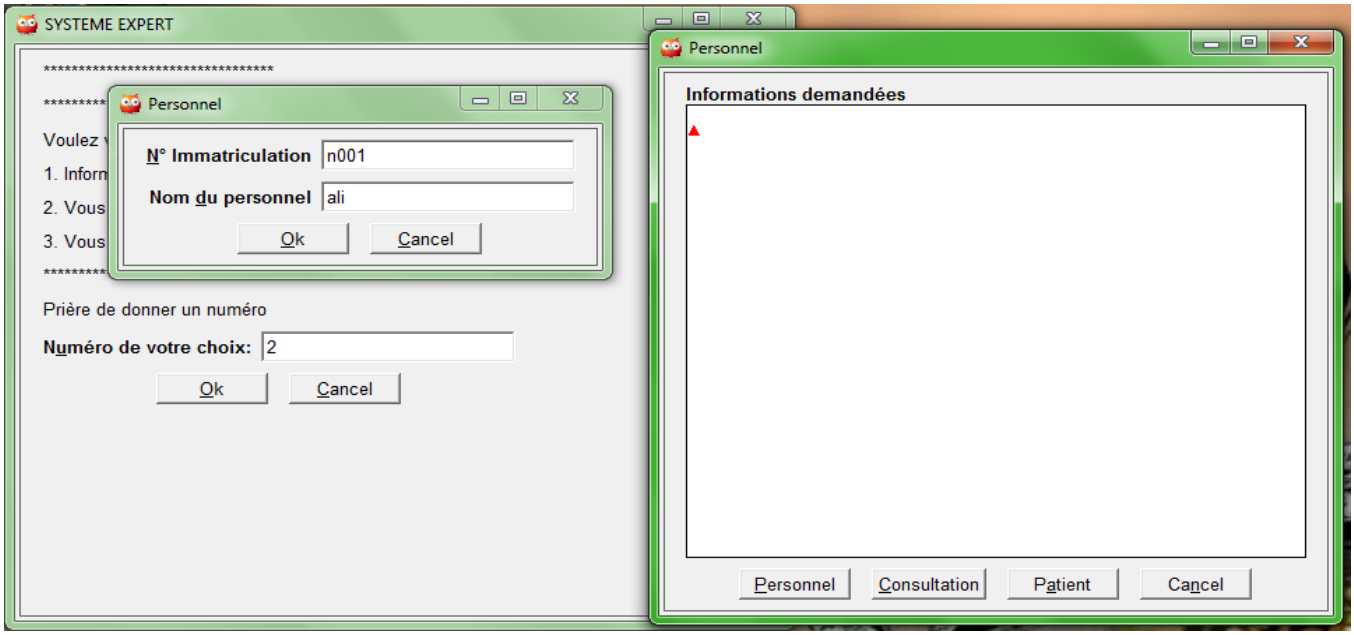


Figure 28:Les informations du personnel de la clinique

Dans le cas où l'utilisateur est le médecin sera une fenêtre spéciale pour insérer le nom et le numéro de votre médecin.

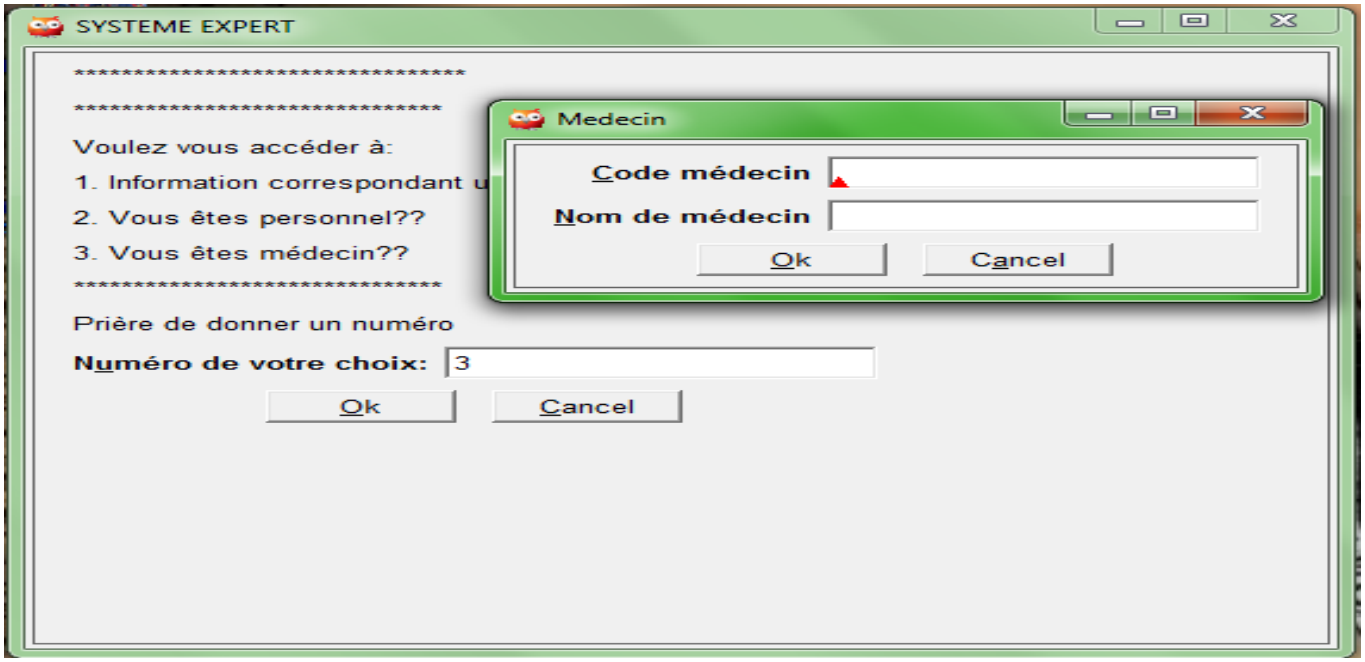


Figure 29: Clinique des Médecins

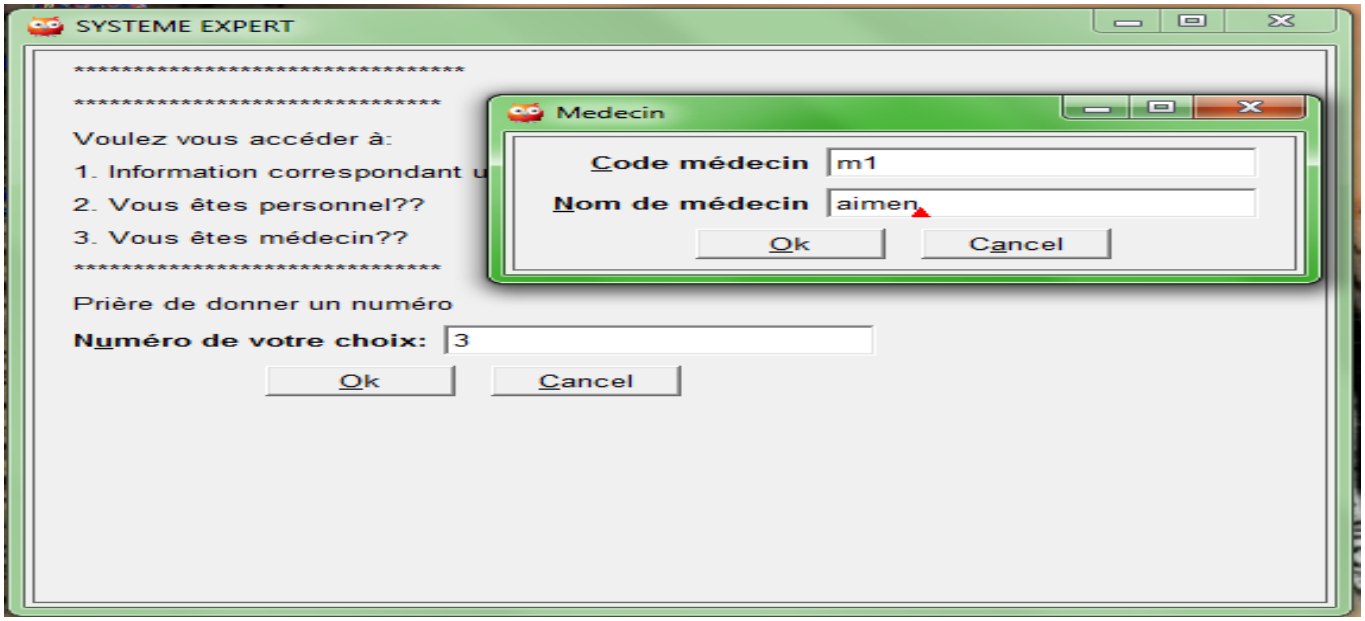


Figure 30:Clinique des Médecins de confidentialité

Lorsque vous entrez le médecin peut voir toute information qu'il veut, que ce soit au niveau des clients ou au niveau du personnel parce qu'il a tous les pouvoirs.

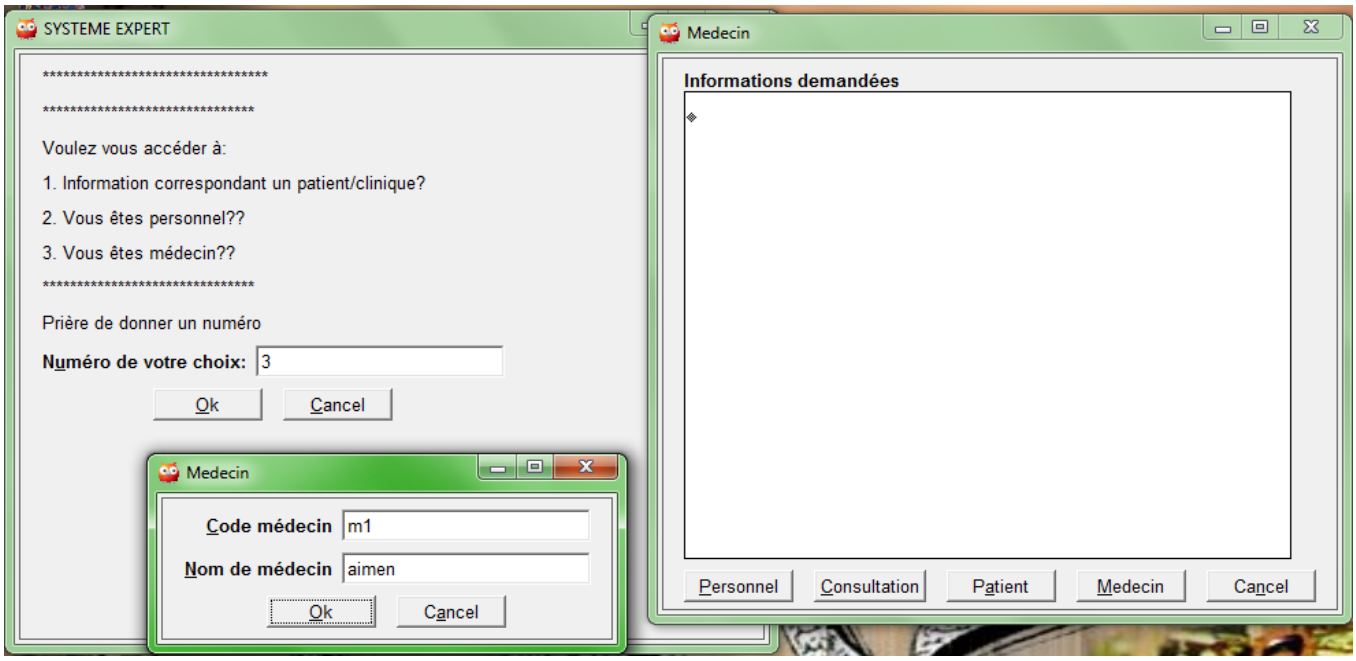


Figure 31:Médecins information clinique

Lorsque vous executes le programmer par l'utilisateur dans le cas et la qualité de information au niveau de 1, par exemple, similaire à d'autres informations au niveau de 3, le programmer va nous montrer cette fenêtre:

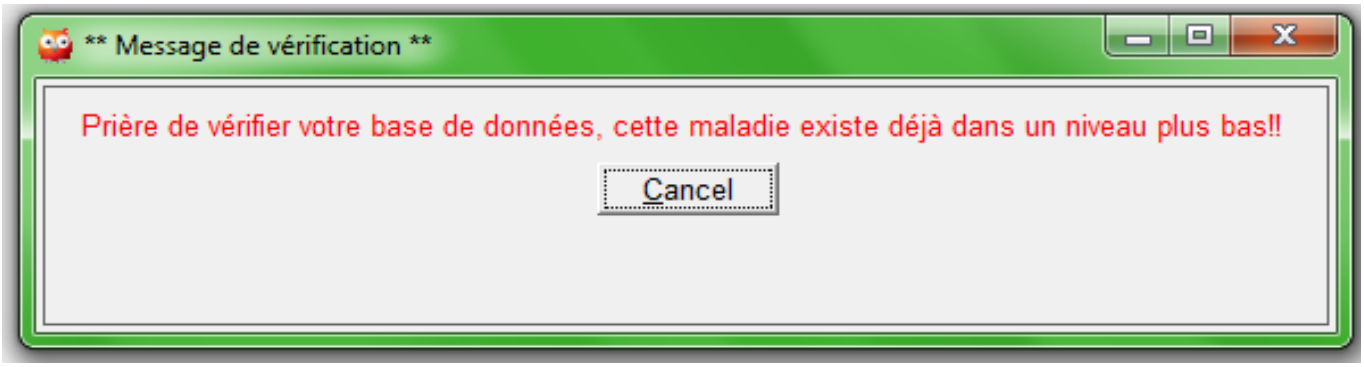


Figure 32: verifier la base de données

5. Conclusion général

Dans le travail présenté précédemment, nous avons mis au point un programme pour surveiller la base de données à plusieurs niveaux. Nous avons utilisé pour cela, les outils XPCE et JPL dans le langage de programmation Prolog.

Nous avons programmé les différents niveaux de la base de données déductive avec leurs différents faits et règles. Un point objectif acceptable est atteint, le programme peut être appliqué et mis en œuvre assez facilement et les résultats obtenus s'affichent clairement.

Comme perspective, nous proposons d'améliorer ce programme mieux pour gérer plus de peine et de correction d'erreurs Suggestions et plus.

Bibliographies

[1] http://docplayer.fr/1377960-Bases-de-donnees-deductives.html

[2] http://georges.gardarin.free.fr//liver_BD_contenu/15-BDDedu.pdf.

[3] liviOer Perrin IUT Nancy-Charlemagne Département Informatique Université Nancy 2 Olivier.Perrin@loria.fr.

[4] P. Jackson. Introduction to Expert Systems. Addition-wesley, 1986, p50.

[5] H. Farreny : les SE, principes et exemples (Cepadues, 1985).

[6] Alain BONNET, Jean-Paul HATON, Jean-Michel TRUONG, système experts : vers la maîtrise technique, Inter Editions, 1986, p12-16.

[7] R. Forsyth. Expert Systems : Principles and Case Studies. Chapman and Hall, 1984, p10-16.

[8] Michel le SEAC'H, développer un système expert, chez Editests, 1989, p50-5.

[9] Michelle SEACH, op.cit, p2.

[10] www.igm_univ-mlv.fr.

[11] PHIL. REITZ, les systèmes experts, cours Dea Ing IA, département ARC, LIRMM, 1993, p10-15.

[12] Bhavani Thuraisingham , William Ford, Marie Collins and Jonathan O'keeffe The MITER Corporation, Burlington Road, Bedford, MA 01730, USA.

[13] Harry S. Delugach Thomas H. Hinke.

[14] Pierre M. Nugues, Language Processing with Perl and Prolog: Theories, Implementation, and Application, page 375, 2014.

[15] Jan Wielemaker, Anjo Anjewierden programming in XPCE/Prolog, université d'Amesterdam 2002.