République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université de 8Mai 1945 – Guelma-
Faculté des Mathématiques, d'Informatique et des Sciences de la matière
Département d'Informatique

Mémoire de projet de fin d'études Master
Filière : Informatique
Option : Systèmes Informatiques

Thème :

Exploitation des bases de données graphes pour le stockage et l'interrogation des données des processus métiers

Encadré par :
Dr. KHEBIZI Ali

Présenté par :
BOUZIANE Bochra

SEPTEMBRE 2021

REMERCIEMENTS

Nous ne nions jamais que nous avons rencontré de nombreuses difficultés au cours de notre cursus universitaire, mais tout cela a abouti à la réalisation de ce travail modeste qui a couronné notre parcours.

Pour cela et surtout, je dois dire, louange à Dieu, qui m'a donnée le courage et la patience nécessaire pour réaliser ce travail.

Je tients de méme, à remercier mes parents et toute ma famille qui ont été à mes cotés tout au long de mes études ainsi que pour leur soutien moral et matériel qu'ils n'ont cessé de me fournir, particulièrement durant la réalisation de mon présent mémoire.

J'adresse, aussi, mes forts remerciements et ma grande gratitude à mon encadreur monsieur **KHEBIZI Ali** pour sa disponibilité et qui n'a pas cessé de me soutenir, de me guider, de me conseiller et de me prodiguer tous les soutiens en besoin tout au long de la réalisation de mon travail.

Je remercie également tout le personnel de notre département informatique de l'université de 8 mai 1945, particulièrement nos enseignants qui nous ont fournis toutes les connaissances attendues pour notre formation.

DEDICACES

Je dédie ce modeste travail

A mes chers parents pour leur soutien continue et leur aide morale et matérielle qu'ils n'ont cessé de me fournir.

A mes frères qui ont suivi avec grande attention toutes les étapes de mes études.

A tous mes proches et connaissances qui m'ont donnée tous les encouragements utiles.

Résumé

Compte tenu de la quantité massive et volumineuse des données qu'apportent les processus métiers , et qui sont à leur tour une partie inévitable du Systèmes Informations et des entreprises actuelles *(industries manufacturières)* le BPM *(Business Process Management)* des processus métiers est devenu une tâche cruciale qui offre un ensemble d'étapes pour la gestion des processus métiers. Et cela doit faire face à de multiples facteurs/enjeux :

(i) la nature de ces données.

(ii) Le temps d'analyse et de traitement de ces immenses données.

Ainsi, la gestion des processus métiers participe à l'amélioration des entreprises sur les quatre critères de la performance qui sont : les coûts ; la qualité ; les délais de production et la flexibilité.

Ainsi, les processus métiers qui étaient seulement une séquence ordonnée et chronologique de tâches *(activités)* et qui ne prenaient pas en considération le terme **données** et qui sont destinées à produire un résultat à valeur ajoutée pour les clients, les actionnaires et les employés de l'organisation avaient leur rendement un peu réduit par rapport à ce qu'il existe actuellement *(l'ajout du terme **données**)*.

Dans ce travail nous exploitons les bases de données orientées graphes pour le stockage et l'interrogation des données des processus métiers où ils sont modélisés sous forme d'automates d'états finis déterministes utilisant les données pour rendre plus performants et plus efficaces les résultats des processus métiers.

Ces dernières (les **données**) sont stockées et manipulées par des bases de données orientées graphes.

Aussi, nous avons opté pour les bases de données orientées graphes compte-tenu de leurs nombreux avantages (flexibilité, connectivité...) dans des multiples domaines utilisateurs des processus métiers.

La modélisation des données par les bases de données orientées graphes a été faite au moyen de l'outil Neo4j qui nous offre une interface performante pour la modélisation des processus métiers et ces données par des bases de données orientées graphes. En outre, il est fait constat que ces bases de données orientées graphes enlèvent le terme de jointure qui n'existe plus dans ces dernières *(BDDG)* contrairement à l'ancienne bases de données *(relationnelle)*.

Abstract

Given the massive and voluminous amount of data that business processes brings, and which in turn is an inevitable part of the information systems and current businesses *(manufacturing industries)* BPM *(Business Process Management)* of business processes has become a crucial task that offers a set of steps for the management of business processes. And this has to face multiple factors / challenges :

(i) the natures of these data.

(ii) The time to analyze and treatment of this immense data.

Thus, the management of business processes contributes to the improvement of companies on the four performance criteria which are : costs ; the quality ; the deadlines of production and flexibility.

Thus, the business process that were only an ordered and chronological sequence of tasks *(activities)* and that did not take into consideration the term **data** and that are intended to produce a value-added result for customers, shareholders and employees of the organization, had their performance somewhat reduced compared to what it currently exists*(the addition of the term **data**)*.

In this work we exploit graph-oriented databases for storing and querying Business process data where they are modeled as deterministic finite state automata using the data to make the business process results more efficient and effective.

The latter **(data)** are stored and manipulated by graph-oriented databases.

Also, we have opted for the graph-oriented databases because of their many advantages (flexibility, connectivity ...) in multiple areas of business process users.

The modeling of the data by the graph-oriented databases was made by means of the Neo4j tool which offers us a powerful interface for the modeling of the data of the business process by graph-oriented databases. Moreover, it is noted that these graph-oriented databases remove the term of joint which does not exist any more in these last (BDG) contrary to the old BDD (relational).

Table des matières

Table des figures ix

Liste des tableaux xi

1 Introduction générale 1
Introduction générale 1

I État de l'art 3

2 Les Processus métiers 4
2.1 Concepts de base des processus métiers 4
2.1.1 Notion de Processus Métier 5
2.2 Instances et traces d'exécution d'un PM 6
2.2.1 La gestion des PM 7
2.3 Cycle de vie des PM 8
2.3.1 Phase Conception et Analyse 8
2.3.2 Phase Configuration 9
2.3.3 Phase Mise en œuvre 9
2.3.4 Phase d'évaluation et Supervision 10
2.4 Les modèles de représentation des PM 10
2.4.1 Le langage BPMN 10
2.4.2 Business Process Execution Language (BPEL) 11
2.4.3 Automate d'états finis AFD 13
2.4.4 Réseaux de Pétri 14
2.4.5 Diagramme UML 16
2.5 Comparaison des modèle de représentation des PM 17
2.6 Les données des PM 19

3 Gestion des données des processus métiers 21
3.1 Techniques de stockage et d'interrogation des données 21
3.2 Les bases de données classiques 23
3.2.1 Les bases de données hiérarchiques 23
3.2.2 Les bases de données relationnelles 25
3.2.3 Les bases de données orientées objets 27
3.3 Les bases de données NoSQL 28
3.3.1 Les bases de données XML 29
3.3.2 Les bases de données orientées documents 30
3.3.3 Les bases de données orientées clé-valeurs 31
3.4 Les bases de données orientées graphes 31
3.4.1 Rappel sur les graphes 31
3.4.2 Fondement des bases de données graphes 32
3.4.3 Avantages 33

3.5 Discussion sur les modèles de stockage des données . . . 33
3.6 Conclusion . . . 35

4 Problématique et Travaux connexes 36
4.1 Problématique . . . 36
4.2 Motivations . . . 38
4.3 Types de données des PM . . . 39
4.3.1 Données descriptive (*schéma*) . . . 40
4.3.2 Données historiques . . . 40
4.3.3 Données d'exécution . . . 40
4.3.4 Données de supervision . . . 41
4.4 Étude comparative des travaux connexes . . . 41
4.4.1 Dans le domaine de la recherche académique . . . 41
4.4.2 Dans le domaine industriel (Industrie du logiciel) . . . 51

II Contribution et Implémentation de l'approche 60

5 Conception de l'approche 61
5.1 Modèle de représentation des PM . . . 61
5.1.1 Utilisation des AFD pour représenter les protocoles des PM . . . 62
5.1.2 Quelques exemples de PM modélisés par des AFD . . . 62
5.1.3 Motivations pour le choix du modèle de PM . . . 66
5.2 Concepts associés au modèle utilisé . . . 66
5.2.1 Instances de processus . . . 67
5.2.2 Exécution et chemin d'exécution . . . 67
5.2.3 Trace d'exécution d'une instance de PM . . . 68
5.3 Enrichissement du modèle de PM par les données . . . 69
5.3.1 Objectifs . . . 69
5.3.2 Formalisation de l'enrichissement . . . 69
5.3.3 Illustration de l'enrichissement des BP . . . 70
5.4 Jonction entre le modèle enrichi et les BDD graphes . . . 71
5.4.1 Spécification de la fonction de mapping . . . 72
5.4.2 Mise en œuvre du modèle enrichi par les données . . . 72
5.5 Conclusion . . . 73

6 Implémentation et Expérimentation de l'approche 76
6.1 Présentation de l'environnement de travail . . . 76
6.1.1 Les APIs Java . . . 76
6.1.2 Neo4j . . . 76
6.1.3 Cypher . . . 77
6.2 Architecture générale et les fonctionnalités du système . . . 78
6.3 Déploiement de Neo4j pour la gestion des données des PM . . . 79
6.3.1 Graphe des PM . . . 79
6.3.2 requêtes sur les graphes des PM . . . 79
6.4 conclusion . . . 84

7 Conclusion Générale **85**

Bibliographie **87**

Table des figures

2.1 Processus Métier du traitement d'une commande Client 5
2.2 Cycle de vie d'un processus métier 9
2.3 Eléments pour modéliser les PMs par des diagrammes BPMN 11
2.4 Processus de commande modélisé par un diagramme BPMN 12
2.5 Processus métier de suivi des soumissions d'articles avec BPEL 13
2.6 Eléments de base d'un AFD 14
2.7 Processus de gestion des commandes modélisé par un AFD 14
2.8 Eléments de base pour modéliser un PM par un RDP 15
2.9 Représentation d'un processus de commandes par un RDP 15
2.10 Inscription en ligne modélisée par un diagramme d'activités 16
2.11 Diagramme de séquences pour l'ajout d'un patient 17
2.12 Représentation d'un DFD exprimant les acteurs et les données 19

3.1 évolution et historique des bases de données existantes 23
3.2 Exemple d'une BDD hiérarchique dans domaine de la santé 24
3.3 Les deux composants d'un SGBDR 25
3.4 Exemple d'une BDD relationnelle du domaine de la scolarité 26
3.5 Une BDD graphe 33
3.6 EXEMPLE 2 BDD graphe 34

4.1 Problématique de la gestion des données des PM 38
4.2 Répartition des principaux outils BPMS 52
4.3 Exemple d'un processus métier en utilisant Oracle BPM 53
4.4 Exemple d'un processus métier en utilisant IBM BPM 54
4.5 Exemple d'un processus métier en utilisant SAP NetWevear 55
4.6 Architecture de l'outil Bonita soft BPM 56
4.7 Exemple d'un processus métier en utilisant Bonita soft 57
4.8 interface de l'outil JBoss jbpm 57
4.9 Exemple d'un processus métier en utilisant JBoss jbpm 58

5.1 AFD du PM de gestion des Commandes Client 63
5.2 Protocole Opérations électorales modélisé par un AFD 64
5.3 AFD modélisant le PM e-learning 65
5.4 réalisation d'un processus de commande par un AFD avec les données 73
5.5 Exemple de modèle comportemental d'un processus e-gouvernement par un AFD avec les données 74
5.6 Exemple de modèle comportemental d'un processus e-learning par un AFD avec les données 75

6.1 Architecture générale de notre système 79
6.2 Le navigateur Neo4j avec l'exemple du e-learning 80
6.3 Le navigateur Neo4j avec l'exemple du e-commerce 80
6.4 Le navigateur Neo4j avec l'exemple du e-gouvernement 81

6.5 exemple de la requête 1 81
6.6 exemple de la requête 2 82
6.7 exemple de la requête 3 82
6.8 exemple de la requête 4 83
6.9 exemple de la requête 5 83
6.10 exemple de la requête 6 84

Liste des tableaux

2.1 Quelques instances et leurs traces d'exécution pour le traitement de commandes. 7
2.2 Etude comparative des modèles de représentation des PM 18

4.1 Tableau d'évaluation des travaux existants 51
4.2 Tableau des bases de données manipulées 58

5.1 Instantiation des données des activités du protocle de elearning 72
5.2 Tableau de correspondances des concepts d'AFDe et des concepts des BDDG 72

abreviations

<**AFD**> <Automate d'états Fini Déterministe>
<**AMBER**> <Architectural Modelling Box for Entreprise Redesign>
<**API**> <Application Programming Interface>
<**BDD**> <Base de données>
<**BDDG**> <Base de données graphe>
<**BDOO**> <Base de Données Orientée Objet>
<**BDDR**> <Base de données relationnelle>
<**BP**> <Business Process>
<**BDM**> <Business Data Model>
<**BPEL**> <Business Process Execution Language>
<**BPM**> <Business Process Management>
<**BPMN**> <Business Process Modeling and Notation>
<**BPMN-Q**> <>
<**BPMS**> <Business Process Management System(software)>
<**BRM**> <Business Rules Management>
<**CE**> <Composition Environnement>
<**DFD**> <Diagramme de Flux de Données>
<**DFG**> <Directly Follow Graph>
<**ELT**> <Extract, Transform and Load>
<**IBM**> <International Business Machines>
<**IBUPROFEN**> <Improvement and BUsiness Process Refactoring OF Embedded Noise>
<**jBPM**> <Java Business Process Mnagement>
<**JEE**> <Java Entreprise Edition>
<**JSON**> <JavaScript Object Notation>
<**LTS**> <Labelled Transition System>
<**NoSQL**> <Not only SQL>
<**ODMG**> <Object Database Management Group>
<**OLAP**> <On-Line Analytical Processing>
<**OLTP**> <OnLine Transactional Processing>
<**OQL**> <Object Query Language>
<**PM**> <Processus Métier>
<**PAIS**> <Process Aware Information System (systèmes d'information contients des processus métiers)>
<**RdP**> <Réseau de Petri>

<**SGBDG**> <Système de Gestion de Bases de Données Graphe>
<**SGBDR**> <Système de Gestion des Bases de Données Relationnelles>
<**SGF**> <Système de Gestion de Fichiers>
<**SGML**> <Standard Generalized Markup Language>
<**SI**> <Systèmes d'information>
<**SQL**> <Structured Query Languages>
<**STE**> <Systèmes de Transition Etiquetés>

<**UI Designer**> <Interfaces Utilisateur Designer>
<**UML**> <Unified Modeling Language>
<**WSDL**> <Web Services Description Language>
<**XLANG**> <XML LANGuage>
<**XML**> <eXtensible Markup Language>
<**XSLT**> <eXtensible Stylesheet Language Transformations>

Chapitre 1

Introduction générale

Introduction générale

L'omniprésence des systèmes d'information a engendré une explosion extraordinaire de la masse de données générée par les systèmes transactionnels qui les supportent. La conséquence immédiate de ce phénomène est l'apparition de nouveaux besoins en termes de stockage et d'interrogation de ces données, qui sont devenus cruciaux. En effet, nous constatons ces dernières années une grande évolution des technologies permettant la gestion des données. Dans cette perspective, une nouvelle approche qui offre l'avantage de prendre en charge d'autres types de données, à part ceux dont la structure est régulière (*sous forme de tables*), a été proposée et mise en œuvre par différents systèmes commerciaux. Il s'agit des approches **NoSQL** (*i.e. Not only SQL*). Parmi ces approches, nous distinguons essentiellement les bases de données orientées graphe qui sont utilisées dans de nombreux domaines de la vie courante, tels que les réseaux sociaux, les réseaux d'ordinateurs de capteurs, ou encore les systèmes biologiques (ADN). Le paradigme des bases de données orientées graphes offre l'avantage de prendre en charge des données dont les caractéristiques sont les suivantes.

1. Le stockage des données est fait d'une manière souple contrairement aux données SQL qui fait avec précaution selon un schéma fixé avec contraintes.
2. Dynamicité accrue des schémas des données manipulées.
3. volume important des données.
4. flexibilité de ces schémas dont les évolutions sont fréquentes.

D'un autre point de vue, la grande majorité des outils BPM actuels proposent aux utilisateurs de choisir le type de SGBD leur permettant de gérer les données afférentes aux processus métiers (PM). Les utilisateurs n'ont qu'a choisir une SGBD parmi le spectre des SGBDR offerts par le système. Néanmoins, les PM sont des spécifications particulières qui sont caractérisés par :

- des interactions inter-processus pour réaliser des objectifs de gestion intra et inter-entreprises.
- Les changements qui surgissent dans l'environnement des organisations imposent des évolutions des schémas des PM. Autrement dit, les processus métiers doivent êtres flexibles.
- Il a été constaté que les requêtes d'interrogation des bases de données relationnelles impliquant des jointures de plus de cinq tables relationnelles sont très couteuses en ressources (*temps, performances*).

Partant de ces considérations, les développeurs et les gestionnaires de processus métiers ont la possibilité de profiter pleinement des avancés réalisées par les bases de données orientées graphes pour la prise en charge des données de leurs processus métiers.

Cette vision pragmatique et prometteuse ne peut être réalisable sans une modélisation adéquate des processus métiers sous forme proche du modèle des graphes, tel que exigé par la technologie des bases de données graphes. En ce sens, les Processus métiers sont dotés d'une grande variété de modèles de représentation et chaque modèle est caractérisé par un niveau d'expressivité particulier. D'une manière générale, les systèmes de transition étiquetés sont largement utilisés dans la littérature de recherche. Dans notre travail, nous avons choisi de représenter les processus métiers par des AFD (automates d'états finis déterministes). Ce choix permettra de concrétiser la perception des processus métiers, en tant que graphe et par conséquent ouvrira la voie pour la prise en charge des données associées par les systèmes de base de données graphe.

Ce projet de fin d'études vise à assurer la jonction entre les processus métiers et la gestion de leurs données sous formes de bases de données graphes. Dans notre travail , nous modélisons les processus métiers sous forme d'automate d'états fini déterministe et nous exploitons la technologie des bases de données orientées graphes pour le stockage et l'interrogation des données des processus métiers. Dans cette perspective, l'outil **Neo4j** est exploité en tant que SGBD orienté graphe pour l'implémentation de notre approche et le langage **Cypher** est déployé pour l'interrogation des données. La suite de ce mémoire est structurée en deux grande partie. La première partie est un état de l'art du domaine et contient les trois chapitres suivants :

- Dans le premier chapitre, on va abordé le domaine des processus métiers ; en exposant les concepts associés, les modèles de représentation et aussi leur cycle de vie. Les outils BPMS commerciaux les plus répandus sont présentés et une brève comparaison entre les modèles de représentation des PM est dressée.
- Le deuxième chapitre est un panorama des différentes techniques des bases de données. On commence par présenter les bases de données classiques (relationnelles et objets) puis on aborde les bases de données non relationnelles. On s'intéresse particulièrement aux bases de données graphes. Une évaluation des différentes techniques est exposée en fin du chapitre.
- Le troisième chapitre expose la problématique de recherche traitée dans ce mémoire. On se focalise sur les aspects qui motivent le problème, et après examen approfondi de l'état de l'art on met en évidence le déficit constaté.

La deuxième partie constitue notre contribution. Elle contient la modélisation de l'approche et son implémentation. Cette partie englobe les deux chapitres suivants :

- Le quatrième chapitre est dédié à la conception de notre approche et à la modélisation du système à implémenter.
- le cinquième chapitre est implémentation de l'approche proposée sous l'environnement Neo4J/Cypher.

On termine le mémoire par une conclusion générale et des perspectives pour d'éventuels travaux futurs.

Première partie

État de l'art

Chapitre 2

Les Processus métiers

Introduction

Les systèmes d'information **(S.I)** contemporains couvrent les différents secteurs d'activités et touchent, pratiquement, toutes les activités humaines. Ils gèrent les informations relatives à l'administration, à l'industrie et au commerce et services. Ces S.I portent sur deux dimensions complémentaires, les données et les traitements. La composition de ces deux éléments permet de prendre en charge les différentes règles de gestion manipulées par les organisations et d'actionner l'ensemble des ressources nécessaires à la réalisation d'un objectif de gestion particulier. Couramment, on parle du concept de **processus métiers**.

Dans ce chapitre nous allons introduire les processus métiers et leur cycle de vie et nous allons aborder la technologie Business Process Management **BPM**. D'autres notions de base liées aux processus métiers, telles que les traces d'exécutions, les instances et les modèles de représentation des PM seront exposées et illustrées avec des exemples.

2.1 Concepts de base des processus métiers

L'utilisation des processus métier (PM) fait appel à un ensemble d'informations pour répondre à un besoin de gestion spécifique (*achat de produits auprès d'entreprises*) .... Ainsi, les PM sont à la base de toute action de gestion au niveau de n'importe quelle organisation. En effet, ce sont ces PM qui permettent à titre d'exemple, de livrer les commandes aux clients, d'assurer le réapprovisionnement en produits, en cas de rupture de stock, de suivre les transactions de paiement des clients, ... En effet, les processus métiers sont au cœur des SI de l'entreprise. Ils permettent de faire un diagnostic interne de toutes les ressources de l'entreprise et une analyse externe des besoins des clients et également de la concurrence du marché.De manière plus générale, ils couvrent les différentes dimensions et aspects opérationnels et décisionnels de toute l'organisation. Ainsi, ils visent à améliorer les performances globales de l'entreprise, en répondant aux questions suivantes :

— Quelle est la stratégie de l'entreprise ?

— Quels sont les objectifs que l'entreprise veut atteindre ?

— comment analyser et superviser les traces des PM afin d'améliorer les performances globales de l'entreprise ?

Dans cette section on va se focaliser sur la définition de PM et la technologie de leur gestion.

2.1.1 Notion de Processus Métier

Le concept de **processus métier** fait intervenir deux notions de *processus* et de *métier*, que nous allons expliciter dans ce qui suit.

Processus Un processus désigne la description d'un système d'activités qui utilise des ressources pour transformer les éléments d'entrée en éléments de sortie. [1]

Métier Un métier est d'abord l'exercice par une personne d'une activité dans un domaine professionnel, en vue d'une rémunération. Aussi, il désigne le degré de maitrise acquis par une personne ou une organisation du fait de la pratique sur une durée suffisante de cette activité [2].

A présent, nous pouvons énoncer une première définition d'un PM par la définition suivante 2.1.

Définition 2.1 *Le processus métier consiste en un ensemble d'activités qui sont exécutées en coordination dans un environnement organisationnel et technique. Ces activités réalisent conjointement un objectif de gestion. Chaque activité est mise en œuvre par une seule organisation, mais elle peut interagir avec des processus métier exécutés par d'autres organisations.*

Exemple 2.1 *Exemple d'un Processus métier*

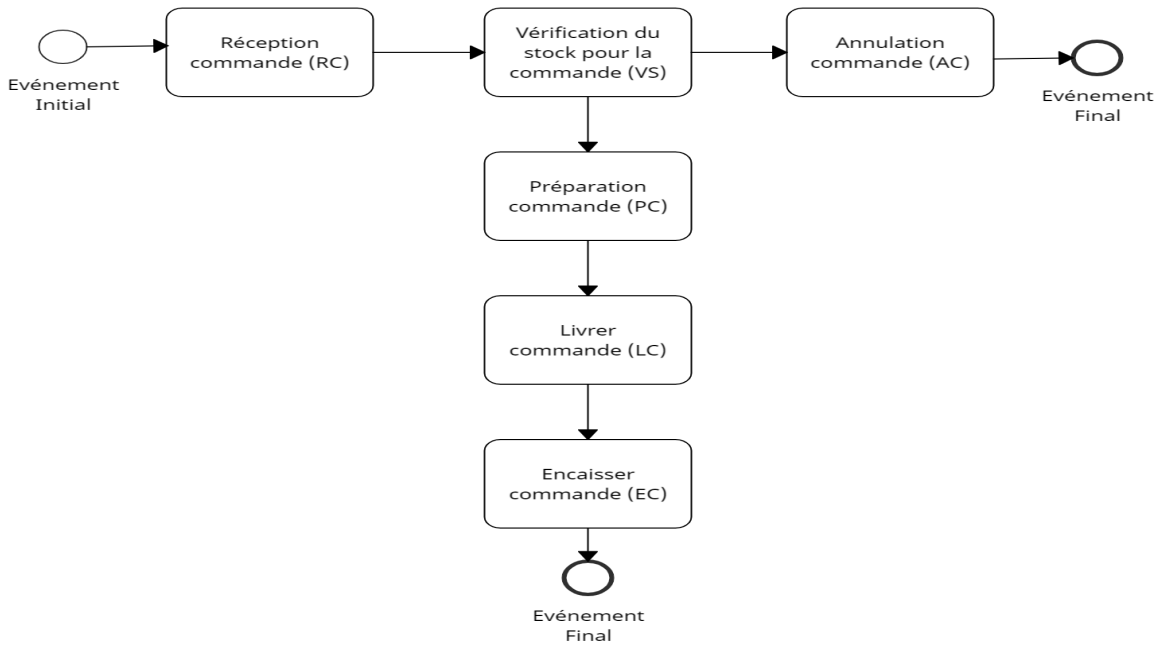

FIGURE 2.1 – Processus Métier du traitement d'une commande Client

La figure 2.1, ci-dessous illustre un PM simple de gestion des commandes clients par une entreprise commerciale. Les phases et les étapes de ce processus sont clairement identifiées.

Cet exemple montre un PM pour la modélisation de traitement d'une commande par un processus métier, où il illustre une séquence d'activités à partir du début de la commande jusqu'à la livraison et l'encaissement de cette commande ou bien l'annulation d'une commande. On peut observer que ce processus passe par plusieurs étapes lors de l'exécution des activités suivantes : recevoir commande, vérification du stock pour une commande, préparation commande, livrer commande et à la fin encaisser commande. Le processus se termine, par un événement final qui est, soit l'encaissement de la commande ou l'annulation commande, comme le montre la figure 2.1.

Après cette définition des processus métier nous pouvons, à présent introduire la gestion des processus métier qui couvre non seulement la modélisation des processus métier, mais également des activités supplémentaires, telles que l'exécution et la supervision des PM. [3]

2.2 Instances et traces d'exécution d'un PM

Une instance de processus métier représente un cas concret dans l'activité opérationnelle d'une entreprise. Elle est constituée d'instances d'activités réalisées par l'instance en question. Chaque modèle de processus métier agit comme un modèle pour un ensemble d'instances de processus métier, et chaque modèle d'activité agit, à son tour, comme un modèle pour un ensemble d'instances d'activités.

Lors de l'exécution d'un processus métier, des "instances" sont créées par des évènements "initiaux" (lancement ou activation de l'instance) du processus. Une fois, ces instances sont exécutées, un processus métier pouvant avoir simultanément plusieurs instances en cours d'exécution, et d'autres qui sont déjà achevées (complètes).

Pour illustrer ces deux concepts, très importants dans la gestion du cycle de vie des **PM**, nous allons donner quelques exemples réels de traces et d'instances, en se basant sur le processus de traitement des commandes client, représenté par la figure 2.1.

Exemple 2.2 ***les instances d'exécution***
Chaque instance d'exécution correspond à un cas réel d'invocation du processus par un client particulier. Ainsi chacune peut avoir atteint un niveau d'exécution qui lui est spécifique. A titre d'exemple, considérons les quatre instances suivantes.

- *trace $\mathcal{T}_1$: exécutée par le client Mohamed et ayant réalisé les 2 activités RC et VS.*
- *trace $\mathcal{T}_2$: exécutée par le client Bochra et ayant réalisé les 3 activités RC et VS et AC .*
- *trace $\mathcal{T}_3$: exécutée par le client Rami et ayant réalisé les 3 activités RC, VS et PC.*
- *trace $\mathcal{T}_4$: exécutée par le client Malika et ayant réalisé les 4 activités RC,VS, PC et LC.*

A présent, nous pouvons spécifier les traces des exécutions historiques réalisées par chacune des traces $\mathcal{T}_1 \ldots \mathcal{T}_4$ précédentes.

Exemple 2.3 ***les traces d'exécution***
La table 5.1 suivante expose quatre traces d'exécution associées aux quatre instances de l'exemple 2.2

Instance de Processus	Traces d'exécution
$\mathcal{T}_1$	RC.VS
$\mathcal{T}_2$	RC.VS.AC
$\mathcal{T}_3$	RC.VS.PC
$\mathcal{T}_4$	RC.VS.PC.LC

TABLE 2.1 – Quelques instances et leurs traces d'exécution pour le traitement de commandes.

2.2.1 La gestion des PM

De nos jours, la gestion des processus métiers (Business Process Management : BPM) a pris une importance capitale dans la gestion de toute entreprise. Par ailleurs, les systèmes d'information contemporains sont axés, de manière fondamentale, sur ce concept. En effet, on parle de systèmes d'information conscients des processus métiers ou Process Aware Information System (PAIS).

La gestion des processus métiers est une approche de gestion des ressources et de mise en œuvre des stratégies d'une organisation qui se focalise sur la maitrise de l'information en vue d'une meilleure prise de décision, tout en prenant en compte les besoins réels des clients. En effet, le but est de favoriser l'efficacité opérationnelle, tout en abordant la question de l'évolution continue des processus métiers et en avantageant le point de vue ***métier*** sur le point de vue ***technique***. Cette façon de voir l'entreprise rend les processus de plus en plus efficaces et capables de s'adapter aux éventuels changements et mutation qui surgissent dans le macro ou micro environnement de l'entreprise.

Nous énonçons, dans ce qui suit, la définition exacte du BPM.

Définition 2.2 *Business Process Management [3]*
La gestion des processus métiers (BPM) est l'utilisation de méthodes, techniques et systèmes logiciels pour concevoir, exécuter, contrôler et analyser des processus opérationnels faisant intervenir des hommes, des applications, des documents et d'autres sources d'information.

De ce qui précède, on observe que la gestion des processus métier comprend des concepts, des méthodes et des techniques pour soutenir la conception, l'administration, la configuration, la mise en œuvre et l'analyse des processus métier.

Actuellement, les processus métiers tirent profit des avancées technologiques et sont pris en charge par des logiciels spécifiques, appelées : les systèmes de gestion de processus métiers, ou Business Processes Management Systems (BPMS), définis ci-dessous.

Définition 2.3 *Systèmes de Gestion de Processus métiers (BPMS) [3]*
BPMS signifie "Business Process Management Software" ou "Business Process Management System".

C'est un système logiciel générique qui est piloté par une représentation de processus explicite pour coordonner la mise en œuvre des processus métiers et pour la gestion de leur cycle de vie.

Le principal avantage de BPMS est que ses utilisateurs ont l'opportunité de participer activement à l'amélioration des processus métiers grâce à des outils simples et intuitifs. Tous ces avantages sont possibles grâce aux trois composants : les moyens de construction des processus, leur exécution (plateforme) et leur surveillance (*bibliothèque de processus*) [4].

Dans la section suivante, nous allons aborder les différentes étapes de la vie d'un BP.

2.3 Cycle de vie des PM

La gestion des différents aspects liés à l'exécution des PM rencontre les problèmes et les contraintes suivantes :

- complexité dans leurs utilisations.
- le risque d'avoir une modélisation trop complexe, avec un modèle abstrait qui est pratiquement impossible a comprendre, d'où le besoin d'optimiser le processus pour des soucis de lisibilité.
- la croissance de journaux d'événements limite l'extraction des processus à grande échelle et présente des difficultés en raison de manque de capacité de gestion de données.

Pour tenter de remédier à ces limitations, les processus métiers, doivent obéir à un cycle de vie qui permet d'assurer leur bon fonctionnement.

Le cycle de vie des processus métiers est illustré à la figure 2.2.

Il se compose de phases liées les unes aux autres. Les phases sont organisées selon une structure cyclique, montrant que leurs dépendances logiques n'impliquent pas un ordre temporel strict dans lequel les phases doivent être exécutées. De nombreuses activités de conception et de développement sont menées au cours de chacune de ces phases [3].

Dans ce qui suit, on donnera une brève explication de chacune des phases du cycle de vie des PM.

2.3.1 Phase Conception et Analyse

Le cycle de vie des processus métiers commence par la phase de conception et d'analyse, au cours de laquelle des enquêtes sur les processus d'affaires et leur environnement organisationnel et technique sont menées. Sur la base de ces enquêtes, les processus métiers sont identifiés, examinés, validés et représentés par des modèles de processus métiers.

Dans cette phase, on utilise différents modèles pour la représentation abstraite de ces processus métiers. L'objectif de cette modélisation est d'offrir des spécifications explicites exprimées dans une notation graphique qui facilite la communication entre ces processus, d'une part, est de permettre d'offrir un moyen de dialogue entre les différents partenaires, afin de les affiner et les améliorer, d'autre part.

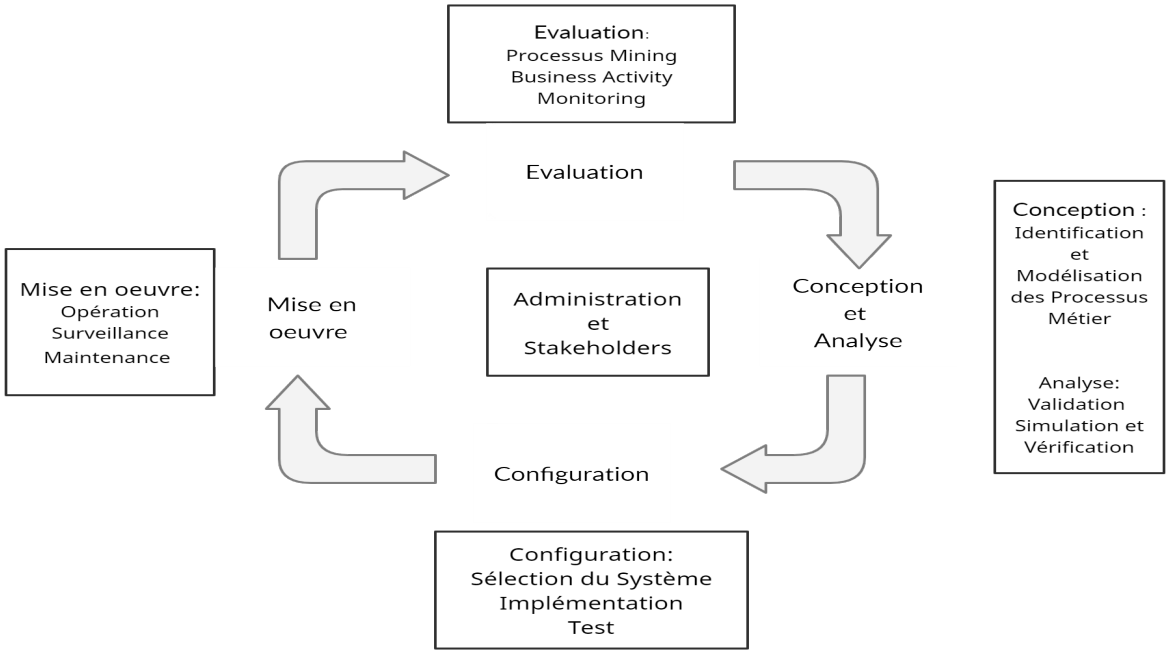

FIGURE 2.2 – Cycle de vie d'un processus métier

2.3.2 Phase Configuration

Une fois le modèle de processus métier conçu et vérifié, celui-ci doit être configuré en fonction de l'environnement organisationnel de l'entreprise. Cette configuration comprend les interactions des employés avec le système ainsi que l'intégration des systèmes logiciels existants avec le système de gestion des processus métier.

La configuration d'un système de gestion des processus métier peut également impliquer des aspects transactionnels. Cela signifie que les transactions en relation avec les bases de données sont exécutées de manière à respecter les propriétés ACID (*atomicité, cohérence, isolation et durabilité*).

Une fois le système configuré, la mise en œuvre du processus métier doit être testée.

2.3.3 Phase Mise en œuvre

La phase de mise en œuvre du processus métier se concentre sur son exécution réelle. En effet, les instances du processus métier sont lancées pour atteindre les objectifs métier d'une entreprise. Le lancement d'une instance de processus suit généralement une séquence d'événements prédéfinie, par exemple, la réception d'une commande envoyée par un client.

Le système de gestion des processus métier contrôle activement l'exécution des instances de processus métier, telles que définies dans le modèle de processus métier. La mise en œuvre de processus doit répondre à une orchestration de processus correcte, garantissant que les activités de processus sont exécutées conformément aux contraintes d'exécution spécifiées dans le modèle de processus.

2.3.4 Phase d'évaluation et Supervision

La phase d'évaluation utilise les données générées au niveau du système de stockage utilisé par le BPMS. L'objectif est de d'analyser et d'évaluer les données existantes, en vue d'améliorer les modèles de processus métier et leurs implémentations. Un point clés d'analyse et l'exploration des journaux d'exécution (*event logs*). Ces derniers sont évalués à l'aide de techniques de surveillance de l'activité de l'entreprise et d'exploration des données générées lors de l'invocation des processus. Certaines techniques d'analyse de données (*Analyse en composante principale, analyse statistique, règles d'association* ...) peuvent êtres utilisées dans ce contexte. Ces techniques visent à identifier la qualité des modèles de processus métiers et l'adéquation de l'environnement d'exécution. A titre d'exemple, la surveillance de l'activité commerciale peut identifier qu'une certaine activité prend trop de temps en raison du manque de ressources nécessaires pour la mener.

2.4 Les modèles de représentation des PM

Plusieurs modèles de représentation ont été proposés dans la littérature de recherche afin de permettre de prendre en charge différentes abstractions et contraintes inhérentes aux processus métiers. Ces modèles offrent des spécifications, plus ou moins, formelles et d'autres sont plutôt graphiques. D'autres part, certains langages, tels que **BPEL** ou **BPMN** intègrent dans leurs spécifications des formalismes et des commandes pour représenter les processus métiers.

Dans cette section, nous allons exposer un panorama des modèles les plus utilisés pour prendre en charge l'aspect modélisation des PM.

2.4.1 Le langage BPMN

Le langage BPMN (Business Process Modeling and Notation) est une norme de notation pour la modélisation des processus métier. Il fournit aux utilisateurs une notation graphique et ce, indépendamment de l'outil utilisé. L'outil étant bien sûr censé supporter la norme [5].

La représentation des processus métier par des diagrammes BPMN nécessite un ensemble d'éléments qui rend la lecture de ces diagrammes compréhensible par tous les utilisateurs de ce langage. Ces éléments sont assemblés dans la figure 2.3 suivante qui représente : les activités, les flux, les relations, les données ainsi que leurs interactions. A noter que ces différents éléments sont la base des spécifications des diagrammes BPMN.

Les catégories d'éléments de base du BPMN sont :

— Objets de flux : Événements, activités, passerelles ;

— Les objets de connexions : Flux de séquence, flux de messages, associations ;

— Artefacts : Objet de données, groupe, annotation ;

L'exploitation des ces éléments de base permet de modéliser les processus métiers de manière adéquate. Dans ce qui suit, un exemple de diagramme BPMN est illustré dans la figure 2.4.

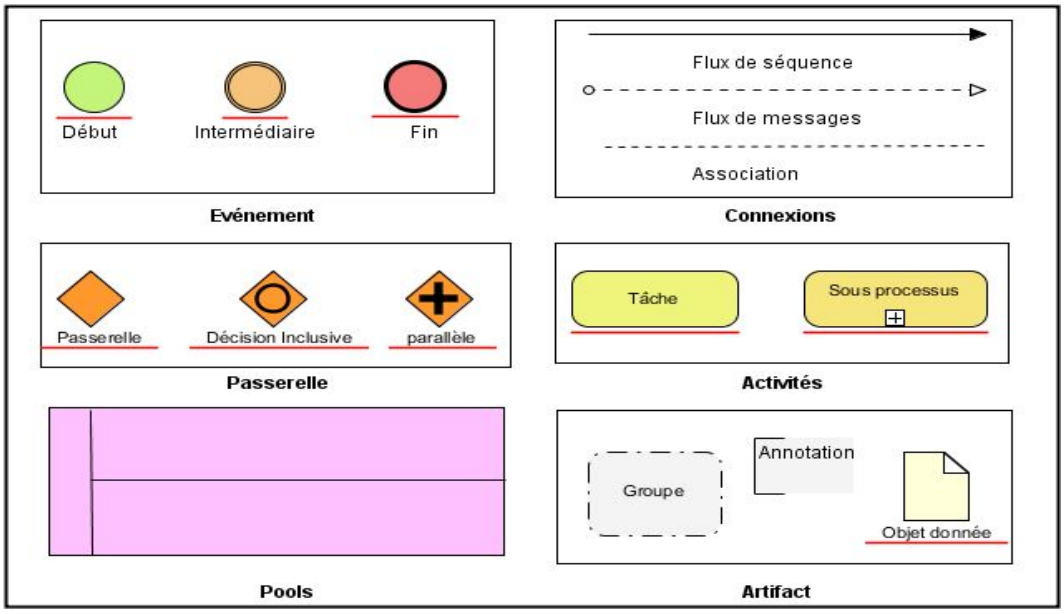

FIGURE 2.3 – Eléments pour modéliser les PMs par des diagrammes BPMN

Exemple 2.4 *Processus Achat modélisé par par un diagramme BPMN*
Dans cet exemple, un processus de commande d'un acheteur est modélisé par la notation de BPMN. Comme illustré dans la figure 2.4, le client commence par passer la commande, après il reçoit la facture, puis il règle le paiement de la facture et enfin, il reçoit les produits.

2.4.2 Business Process Execution Language (BPEL)

BPEL (**B**usiness **P**rocess **E**xecution **L**anguage) est un langage de programmation destiné à l'exécution de l'ensemble des tâches des processus métiers. Il se base principalement sur le standard XML.

BPEL permet de décrire les interactions entre les services Web implémentant les processus métiers. Cette spécification s'inspire des deux langages de processus qui sont XLANG de Microsoft et WSDL d'IBM. BPEL s'est imposé comme standard de base pour la composition des services Web.

C'est un langage qui supporte la spécification des schémas de composition et des protocoles de coordination. Le schéma de composition exprimé en BPEL est une véritable spécification d'un processus exécutable qui définit la logique d'implémentation d'un processus métier composite. Par ailleurs, le protocole de coordination est une perspective orientée service. Cette dualité descriptive de BPEL, permet de définir la séquence de messages échangés par un service (*messages envoyés et reçus*) et prend en compte les contraintes d'ordre entre les opérations constituant la logique métier.

En d'autres termes, BPEL peut être utilisé pour définir le comportement externe d'un service (*à travers un processus abstrait :* BPEL Abstract) aussi bien que pour spécifier l'implémentation interne (*à travers un processus exécutable*). Le processus BPEL spécifie l'ordre exact de l'invocation des services Web participant dans la composi-

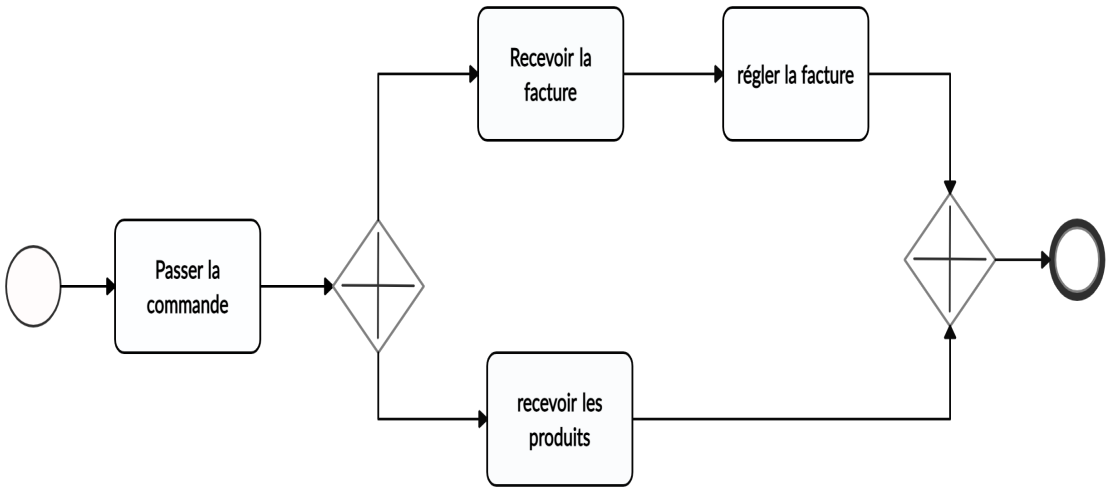

FIGURE 2.4 – Processus de commande modélisé par un diagramme BPMN

tion. L'invocation peut se faire soit en parallèle soit d'une manière séquentielle. Aussi, l'expression d'un comportement conditionnel est offerte, par exemple, lorsque l'invocation d'une activité peut être dépendante du résultat d'une invocation antérieure. La construction des boucles, la déclaration de variables et l'affectation de valeurs ..., sont aussi possibles. De plus, il est possible de combiner toutes ces constructions et de définir des processus métiers complexes d'une manière algorithmique.

D'une part, la partie Execution du standard BPEL est utilisée pour spécifier le déroulement du processus et pour exprimer les séquences d'évènements du processus métier. Le modèle de processus exécutable obtenu peut, à présent, être déployé et pris en charge par un moteur de processus métiers qui mettra en œuvre les règles métiers décrites. D'autre part, la partie Abstract d'un programme BPEL permet d'exprimer le comportement du PM. En effet, dans cette partie, la spécification de l'échange de messages publics entre les parties est décrite. Ainsi, la partie Abstract n'inclue pas les détails internes des flux de processus et elle n'est pas exécutable. Elle assure, simplement, le paradigme de la chorégraphie. Chaque processus abstrait est censé être une description complète du comportement externe pertinent pour le ou les partenaires avec lesquels il interagit.

Un processus BPEL spécifie l'ordre exact dans lequel les processus métiers participants doivent être invoqués, de manière séquentielle ou en parallèle. Avec BPEL, il est possible d'exprimer des comportements conditionnels. Par exemple, un appel d'un processus métier, implémenté en tant que service Web, peut dépendre de la valeur d'un appel précédent. Il est possible, également, de construire des boucles, déclarer des variables, copier et affecter des valeurs, définir des gestionnaires d'erreurs, etc. En combinant toutes ces constructions, les processus métier complexes sont construits de manière algorithmique.

Exemple 2.5 *Exemple de PM avec BPEL*
L'exemple suivant décrit un processus métier relatif à la soumission des articles de recherche pour les revues scientifiques.

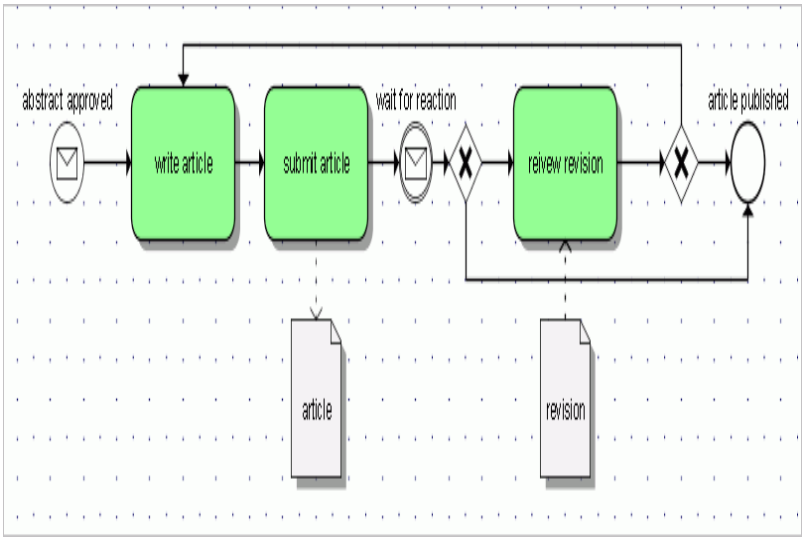

FIGURE 2.5 – Processus métier de suivi des soumissions d'articles avec BPEL

2.4.3 Automate d'états finis AFD

les **A**utomates d'états **F**inis **D**éterministes (AFD) sont des modèles mathématiques formels très utilisés pour la modélisation des systèmes dynamiques. De manière très simpliste, un AFD se compose d'un ensemble d'états reliés entre eux par des transitions (*les fonctions de transitions*) qui sont étiquetées par des symboles. Ce système a la possibilité de vérifier si un mot en entrée est accepté par l'automate ou refusé et cela après depuis l'état initial jusqu'à la fin, en parcourant les états et les transitions lettre par lettre. Schématiquement, les éléments de bases pour la représentation des processus métier en AFD sont assemblés par la figure 2.6.

Définition 2.4 ***Protocole métier*** *[6]*
Le protocole d'un processus métier est un tuple $\mathcal{P} = (Q, q_0, \mathcal{F}, \mathcal{M}, \mathcal{R})$*, tels que :*

- Q *est un ensemble fini d'états ;*
- $q_0 \in Q$ *est l'état initial du protocole ;*
- $\mathcal{F} \subseteq Q$ *est l'ensemble des états finaux du protocole (ou acceptant) ;*
- $\mathcal{M}$ *est un ensemble fini d'activités abstraites ;*
- $\mathcal{R} \subseteq Q \times Q \times \mathcal{M}$ *est une relation de transition. Chaque élément* $(q, q', m) \in \mathcal{R}$ *représente une transition d'un état source* q *vers un autre état cible* q'*, suite à l'exécution de l'activité* m*.*

La figure 2.7, ci-dessous illustre un processus métier modélisé par un AFD. Ce processus métier est relatif à la gestion des commandes clients.

Exemple 2.6 *La figure 2.7 expose un AFD pour la gestion du processus de commande des clients. Ce processus est composé d'un ensemble d'états qui commence par un état initial (Début), suite à la connexion d'un client et se termine par des états finaux (Commande Archivée), dans le cas où la commande est acceptée, ou bien l'état final*

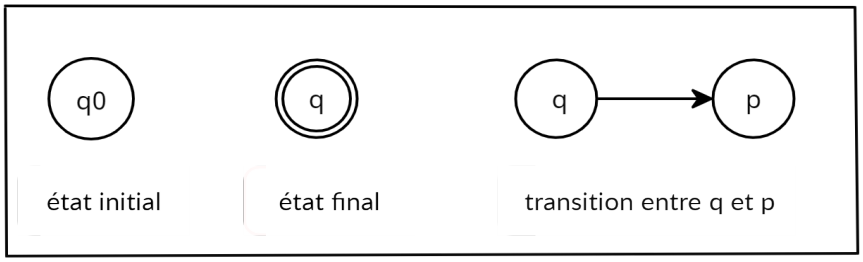

FIGURE 2.6 – Eléments de base d'un AFD

Commande rejetée si la commande est rejetée. Pour atteindre ces deux états finaux, une succession d'activités est réalisée, tout en respectant les règles métiers imposées par le fournisseur.

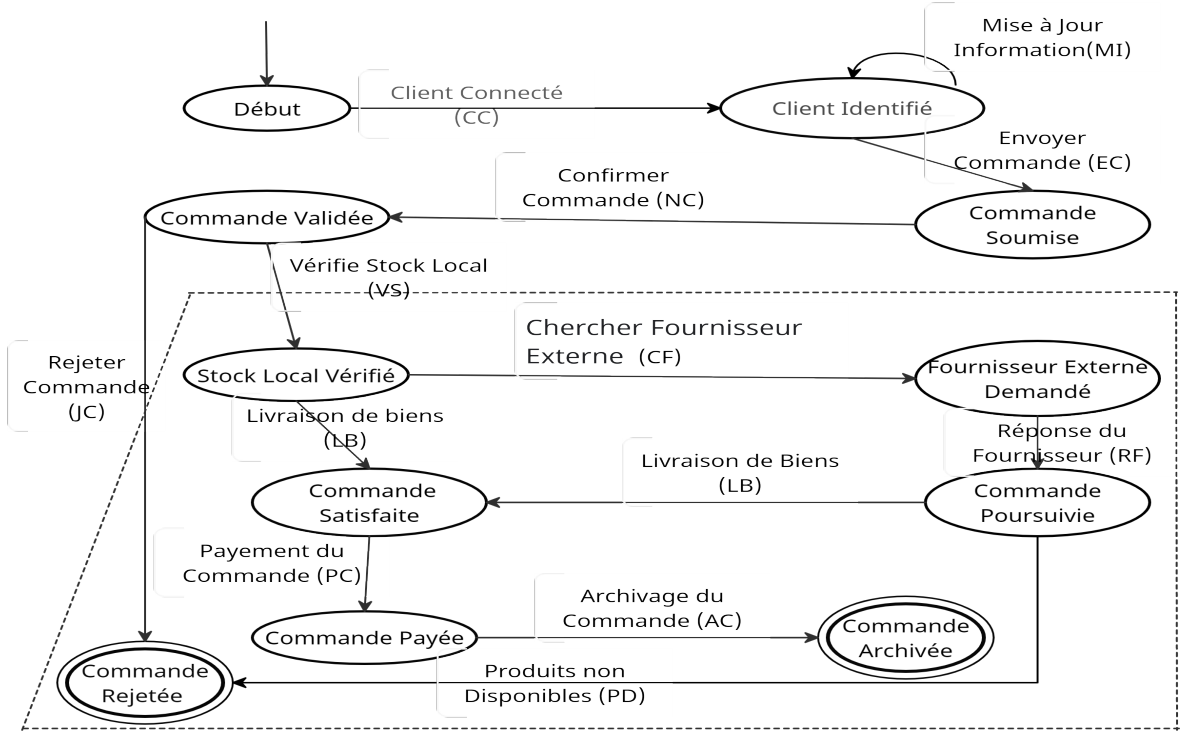

FIGURE 2.7 – Processus de gestion des commandes modélisé par un AFD

2.4.4 Réseaux de Pétri

Les éléments de bases pour la représentation des processus métiers par des réseau de Petri sont assemblés dans la figure 2.8.

Les réseaux de Petri modélisent des systèmes dynamiques qui évoluent d'un état à un autre lorsqu'un événement interne ou externe survient [7]. Le BP se présente sous

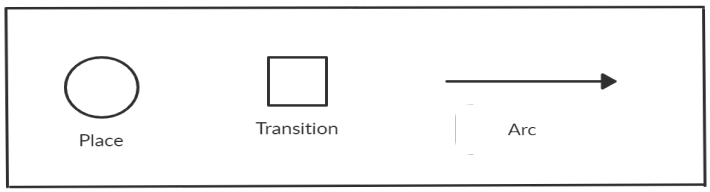

FIGURE 2.8 – Eléments de base pour modéliser un PM par un RDP

la forme d'un graphe biparti alterné, sur lequel les nœuds illustrent les transitions. Tout arc doit aboutir à un nœud à l'une de ses extrémités. Un arc relie une transition à une place ou bien inversement une place à une transition.

Plus formellement, on défini un réseau de Pétri comme suit :

Définition 2.5 *Un réseau est un triplet $N =< P, T; F >$ où :*

- *P est un ensemble fini de places*
- *T est un ensemble fini de transitions où P*
- *F relation de flot sur $N : \subseteq F\ (P \times T) \cup (T \times P)$*

ou $F \subset (P \times T \cup T \times P)$ est l'ensemble des arcs appelés la relation de flux.

Exemple 2.7 *Processus de commande réalisé par un réseau de Pétri*
La figure 2.9 montre la réalisation d'un processus de commande par un RDP, en partant initialement de la réception de la commande jusqu'à le livraison ou l'échec en cas de rupture de stock.

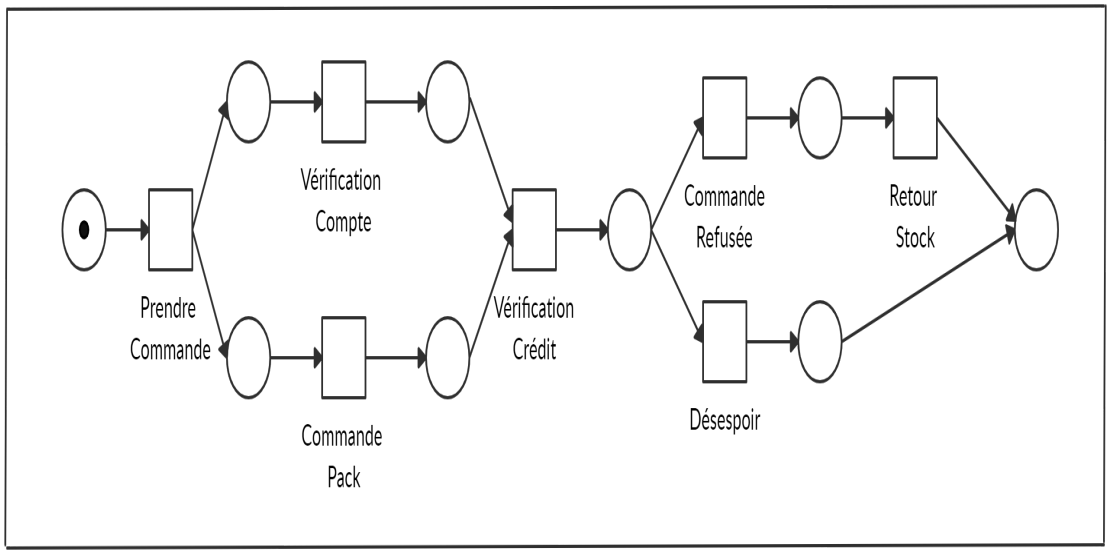

FIGURE 2.9 – Représentation d'un processus de commandes par un RDP

2.4.5 Diagramme UML

UML est un langage pour la modélisation des systèmes logiciels et processus métier qui offre plusieurs diagrammes pour les différents besoins associés au cycle de vie d'un logiciel.

Dans le contexte de notre étude relative aux processus métiers, on vas s'intéresser seulement aux diagrammes d'activités et diagrammes de séquences.

Diagramme d'activités

Les diagrammes d'activités sont utiles pour la modélisation des activités des entreprises. Ils sont utilisés pour détailler les processus impliqués dans des activités commerciales et administrative par l'expression des règles de gestion associées aux diffèents procédures de travail [8].

De manière simpliste, un diagramme d'activités permet de représenter l'ensemble des activités séquentielles d'un processus métier. Il commence par un point de départ, exprime les activités associées au processus et se termine par un point d'arrivé.

Exemple 2.8 ***Processus d'une inscription en ligne réalisé par un diagramme d'activités***
La figure 2.10 montre la succession des activités pour l'inscription en ligne où on commence par la demande d'une inscription et le remplissage d'un formulaire par la suite on achève cette inscription par le refus (inscription refusée) ou la terminaison avec sucés (inscription terminée).

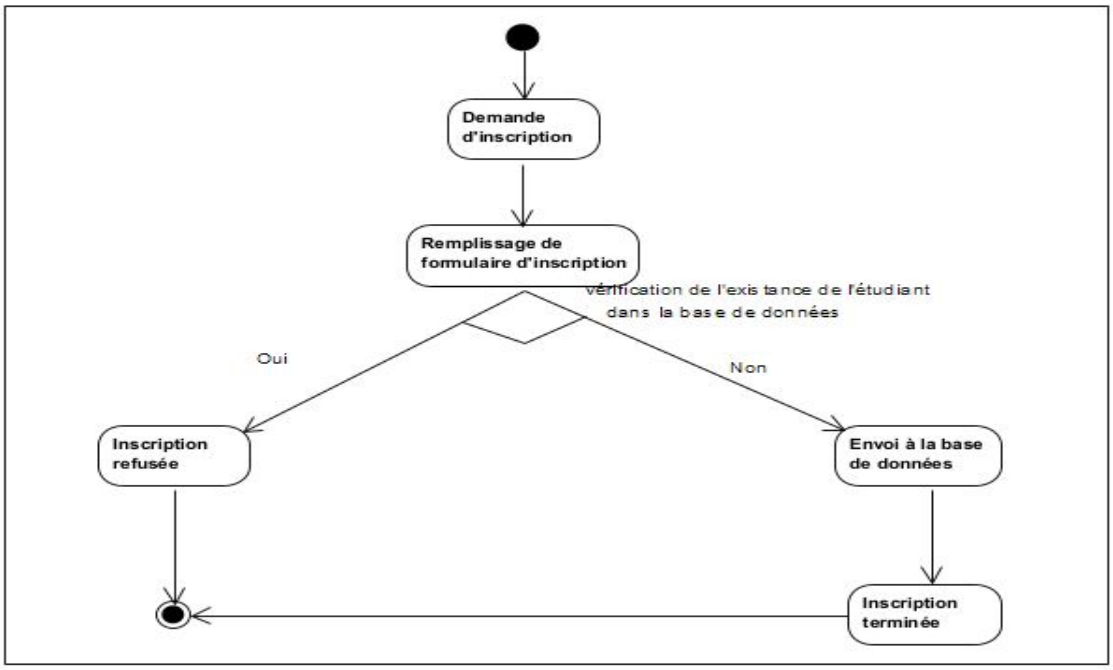

FIGURE 2.10 – Inscription en ligne modélisée par un diagramme d'activités

Diagramme de séquences

Un autre modèle dynamique proposé par UML pour prendre en charge la représentation des activités de l'entreprise est le diagramme de séquences. C'est un type de diagramme d'interaction qui décrit comment et dans quel ordre plusieurs objets inteagissent ensemble [9].

Exemple 2.9 ***Ajout d'un client modélisé par un diagramme de séquences*** *Ce diagramme représente le scénario de l'ajout d'un nouveau patient qui se présente la première fois au bureau des entrées d'un établissement de santé. comme illustré dans la figure ci-dessous, si le client est déjà inscrit, un message est lui envoyé (patient déjà existé),*

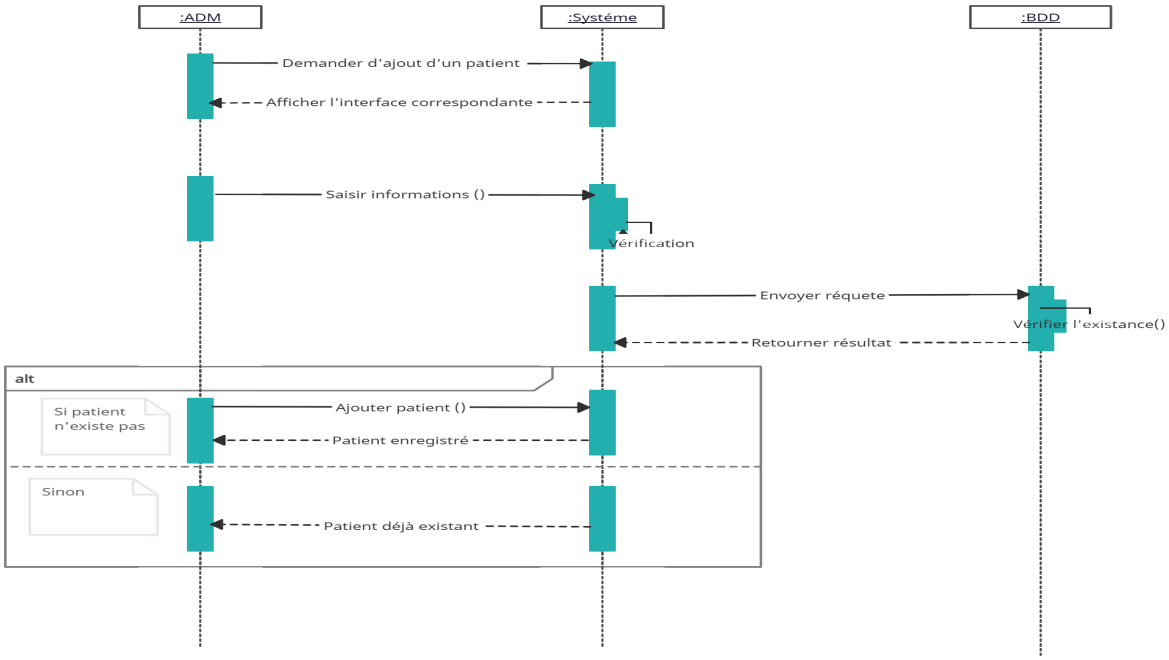

FIGURE 2.11 – Diagramme de séquences pour l'ajout d'un patient

2.5 Comparaison des modèle de représentation des PM

Beaucoup de modèles ont été proposés dans la littérature pour représenter les processus métier. Ces modèles sont dotés de niveaux d'expressivité variés et permettent de prendre en charge différents types de contraintes. Néanmoins, nous pouvons classer ces modèles en trois grandes familles qui sont les suivantes.

1. Famille des langages (BPEL, BPMN, ...),
2. Famille des modèles formels (AFD, Petri Nets,logiques descriptives, pi-calculus, ...),
3. Famille des modèles graphiques (Diagrammes de séquences et d'activités), diagrammes de flux de données (DFD), ...).

Chacun de ces modèles dispose de ses propres caractéristiques et de ses éléments de base pour représenter les processus métiers.

Pour évaluer l'expressivité et la pertinence de chaque modèle, nous avons dressé dans le tableau 2.2, une comparaison des différents modèles. Cette comparaison est basée sur un ensemble de critères pertinents que nous avons identifié et qui mettent en relief les forces et les faiblesses de chaque modèle.

Critères/ Modèles	BPMN	BPEL	AFD	RDP	UML	DFD
Représentation de granularité	Oui	Non	Oui	Oui	Oui	Oui
Description de l'enchainement	Oui	Non	Oui	Oui	Oui	Oui
Traitement de conditions	Oui	Non	Non	Oui	Oui	Oui
Gestion des événements	Oui	Non	Oui	Oui	Oui	Oui
Gestion des documentaire	Oui	Non	Non	Non	Oui	Non
Gestion de la performance	Non	Non	Non	Non	Non	Non
Description des ressources Oui	Non	Non	Non	Non	Oui	Non
Intégration des vues	Oui	Non	Non	Non	Oui	Non

TABLE 2.2 – Etude comparative des modèles de représentation des PM

UML et BPMN semblent être proches par rapport à nos critères. Cependant, Il faut signaler, que le formalisme BPMN a été construit spécifiquement pour la modélisation des processus métier (*contrairement aux diagrammes UML*). De plus, BPMN possède un lien direct avec le langage d'exécution BPEL. Ainsi, BPMN pourra permettre de réconcilier l'aspect modélisation des processus métiers avec les besoins de mise en oeuvre et d'exécution des processus métiers. Même si certains outils le font de manière semi-automatique, le passage d'une modélisation des processus métiers à une modélisation pour l'exécution de ces processus demande un travail d'adaptation manuelle. Ces adaptations peuvent entraîner des erreurs et même rendre difficile la compréhension, pour les concepteurs, des évolutions de leurs propres processus.

D'autre part, dans le cadre de notre étude, il est impératif de disposer d'un modèle **abstrait et formel** afin de pouvoir aborder la vérification de certains critères de conformité des processus métiers, tels que les états atteignables, les chemins redondants, la redondance des sous chemins, ... Les deux classes d'automates finis, les automates finis déterministes (**AFD**) et non déterministes, ont la même puissance d'expression : elles reconnaissent la même famille de langages, à savoir les langages rationnels, appelés aussi langages réguliers ou langages reconnaissables. En ce sens, les AFD s'avèrent les modèles les plus adéquats pour modéliser les processus métiers les raisons suivantes :

— simple, populaire, facile à manipuler,

— largement utilisé dans l'algèbre des processus et les systèmes réels dynamiques,

— existence d'une large gamme d'outils (logiciels) pour leur gestion et leur vérification.

Néanmoins, leur inconvénient majeur est la taille, mesurée en nombre d'états, qui peut dans certains cas être exponentielle par rapport à leur contre-part non déterministe.

2.6 Les données des PM

Une large parte des travaux de recherche sur la modélisation et la gestion du cycle de vie des processus métiers s'est focalisée, essentiellement, sur la gestion du flux d'activités. En effet, ces travaux ne considèrent que l'aspect activités et relèguent l'aspect données à un second plan. Cependant, il est évident que les données des processus métiers sont aussi importantes que les activités elles mêmes.

Pour remédier à cette insuffisance, plusieurs travaux ont tenté de prendre en charge l'aspect données des processus métiers par la proposition de modèles qui permettent de spécifier, et les données et les traitements en même temps. Ces modèles de processus métiers s'intéressent, aussi bien à la logique métier manipulée (*séquencement des activités*) qu'aux données gérées lors de l'exécution des différentes activités du processus. Dans cette perspective, certains spécialistes ont pensé à d'autres modèles qui peuvent gérer les PM en donnant une grande importance aux données.

Exemple 2.10 *Utilisation des DFD pour spécifier les processus métiers*

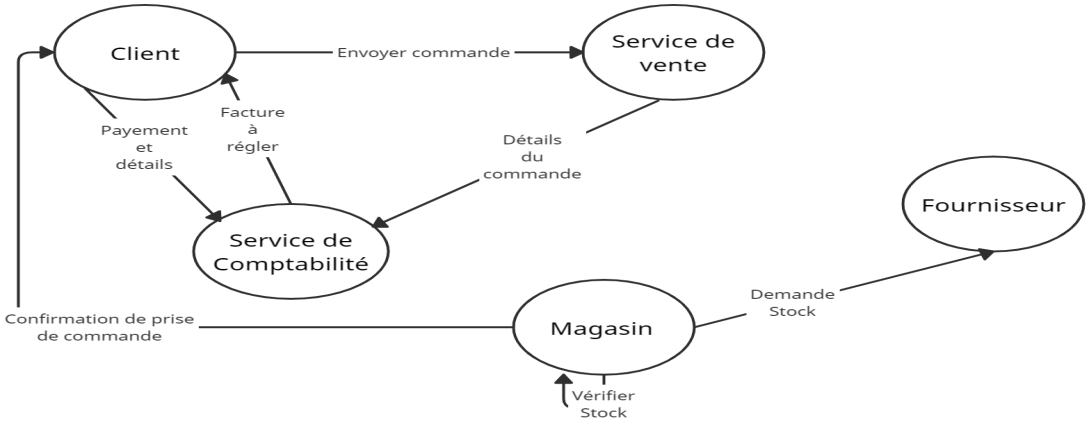

FIGURE 2.12 – Représentation d'un DFD exprimant les acteurs et les données

Pour illustrer cet aspect, considérons un modèle classique qui sont les diagrammes de flux de données (DFD). Ces diagrammes décrivent les processus métiers en montrant comment les différents acteurs d'un processus donné interagissent entre eux via les échanges de données, tout en accomplissant des procédures et des fonctions de l'organisation.

Les DFD sont l'épine dorsale de l'analyse structurée qui a été développée au début des années soixante par Yourdon [5].

Ainsi, un DFD peut être considéré comme une méthode d'organisation des données à partir de leur état brut.

A signaler que vue l'importance de la dimension données des processus métiers, elle sera abordée en détails dans les prochains chapitres.

Conclusion

Dans ce premier chapitre, nous avons introduit les processus métiers et nous avons exposé les concepts qui leur sont associés ainsi que les techniques de leur gestion. L'accent a été mis, particulièrement, sur leurs modèles de représentation et un panorama des différents modèles a été dressé, tout en comparant ces modèles sur la base de certains critères que nous avons spécifiés.

Nous avons terminé le chapitre, par un détour sur l'aspect données manipulées par les processus métiers et nous avons attiré l'attention sur cette dimension qui sera traitée avec plus de précision dans la suite du mémoire.

Le prochain chapitre sera consacré aux différentes techniques de gestion et manipulation des données des processus métiers. L'accent sera mis particulièrement sur les bases de données graphes.

Chapitre 3

Gestion des données des processus métiers

Introduction

La question fondamentale du stockage des données est l'une des problématiques les plus étudiées dans la discipline informatique. En effet, les structures de fichiers et de bases de données n'ont pas seulement une relation avec l'utilisation de l'ordinateur, mais elles sont indispensables dans tous les systèmes d'informations, aussi bien dans l'exercice des principales activités de la vie quotidienne (*gestion des commandes, consultation des comptes, ...*) que pour les activités relatives aux systèmes stratégiques (*systèmes de prise de décision, indicateurs de performances ...*).

Dans ce chapitre, nous allons nous focaliser sur cet aspect important des processus métiers et nous allons étudier les catégories existantes des bases de données ainsi que leur importance dans le domaine des processus métiers (*stockage et interrogation des données des processus métier*).

Nous allons aborder les points suivants relatifs aux bases de données, à savoir :

- Le stockage des données des PM.
- Les techniques de stockage et d'interrogation des données.
- Les différentes catégories de bases de données existantes (*Classiques, NoSQL, base de données graphes*)

Par la suite, nous effectuons une étude comparative des différents modèles de bases de données existants afin de faire ressortir les avantages et les inconvénients de chaque modèle.

Nous commençons le chapitre par une discussion sur l'aspect stockage et interrogation des données des processus métiers.

3.1 Techniques de stockage et d'interrogation des données

Dans le cadre de l'utilisation des données, il est clairement constaté que ces données constituent un point essentiel dans la bonne marche de toute l'entreprise. Néanmoins, les premiers systèmes de gestion de fichier (**SGF**) manifestaient des limites induites par les raisons suivantes [10].

- L'accès aux données était une tâche énorme et fastidieuse à réaliser.
- Besoin d'une programmation étendue dans un langage de troisième génération tel que COBOL, BASIC

- Séparation et isolation.
- Dédoublement des données – les mêmes données sont détenues par différents programmes, ce qui gaspille de l'espace et des ressources.
- Coûts de maintenance élevés tels que la cohérence des données et le contrôle de accès.
- Faible sécurité.

D'autre part, avec l'accroissement grandissant des moyens de traitements de l'information, la baisse des coûts des ordinateurs et la démocratisation de l'utilisation de l'informatique, actuellement les données générées par les systèmes existants ont vu leur importance et leur volume augmenter de manière spectaculaire. La conséquence immédiate de ce constat et que les bases de données relationnelles ont manifesté certaines limitations et elles sont devenues insuffisantes pour la manipulation de cette masse de données. Pour faire face à cette difficulté, les spécialistes se sont intéressés à d'autres approches pour améliorer et perfectionner le stockage, l'interrogation et la manipulation de données. Ainsi sont apparus divers autres approches et procédés pour la prise en charge du cycle de vie des données (*création, stockage, recherche, exploitation et archivage*). Parmi ces nouveaux modèles utilisés, nous distinguons en particulier les bases de données orientées graphes dont les caractérisations sont leur souplesse et leur flexibilité.

D'où est née la nécessité de trouver des solutions pour accéder à cette masse énorme et importante des données. Ainsi sont apparus les premiers systèmes de gestion des bases de données automatisées qui prennent en charge l'aspect description des données et manipulation ou interrogation des données.

Avant de présenter la panoplie des techniques de stockage des données existante dans la littérature, nous commençons par rappeler quelques notions fondamentales sur les bases de données.

Dans [11], l'auteur a défini une base de données comme suit :

Définition 3.1 *Une base de données est un ensemble de données qui ont été stockées sur un support informatique, organisées et structurées de manière à pouvoir facilement consulter et modifier leur contenu.*

Suivant cette définition, les avantages des BDD d'après [11] sont :

— L'indépendance entre données et traitements ;

— La duplication des données est réduite ;

— La base de données dote l'entreprise d'un contrôle centralisé de données opérationnelles qui représentent le capital important de l'entreprise ;

— L'ordre dans le stockage de données ;

— L'utilisation simultanée des données par différents utilisateurs.

Historiquement, deux grandes familles des bases de données sont apparues. La première famille est nommée les BDD classiques et contient les bases de données hiérarchiques, relationnelles et orientées Objets. La deuxième famille concerne les bases de données NoSQL avec ses différentes sous familles (*Documents, Clé-valeurs, XML, BDD orienté graphes*).

Nous pouvons remarquer que les bases de données NoSQL n'ont pas le même objectif que les bases de données SQL. Ces bases de données sont destinées à gérer un volume de données très conséquent. Elles doivent être plus performantes pour supporter une montée en charge plus élevée. Cela implique une simplification du stockage des données (*autorisation de la redondance*) et l'abandon de fonctionnalités et de principes propres aux bases de données traditionnelles.

Le schéma récapitulatif ci-dessous 3.1 montre l'évolution historique du domaine des bases de données.

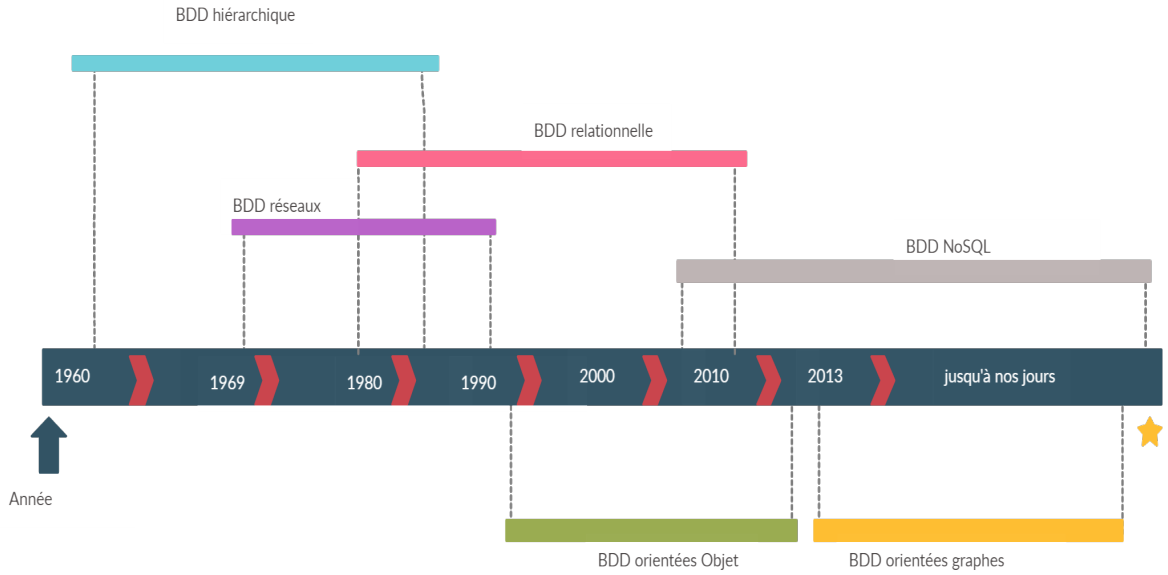

FIGURE 3.1 – évolution et historique des bases de données existantes

La suite de ce chapitre sera dédiée à l'exposé des différents modèles et techniques de stockage et d'interrogation des données des processus métiers.

3.2 Les bases de données classiques

Dans cette section, on exposera les différentes catégories de bases de données classiques, à savoir : hiérarchiques, relationnelles et orientées objets.

3.2.1 Les bases de données hiérarchiques

Au milieu des années 1960, Rockwell [1] collabore avec IBM pour créer le système de gestion de l'information (IMS), IMS [2] leader sur le marché des bases de données mainframe dans les années 70 et au début des années 80.

1. Entreprise de génie électrique et d'automatisation
2. IBM Information Management System

Définition 3.2 *[12] Une bases de données hiérarchique permet de structurer l'information de façon hiérarchique où chaque enregistrement dépendait d'un seul enregistrement. Elle se présente sous forme d'arborescence (arbre connexe et sans cycle).*

Dans ce modèle, les fichiers sont liés de manière parent/enfant, où chaque fichier enfant ayant au plus un fichier parent [10].

La figure 3.2 montre un exemple d'une BDD hiérarchique [12].

Les principales limites des BDD hiérarchiques sont :

— Un seul enregistrement parent ce qui rend la recherche difficile et coûteuse [12] ;
— Mise en œuvre complexe ;
— Difficile à gérer et manque de normes ;
— Difficulté de gérer plusieurs relations ;
— Manque d'indépendance structurelle [10] ;

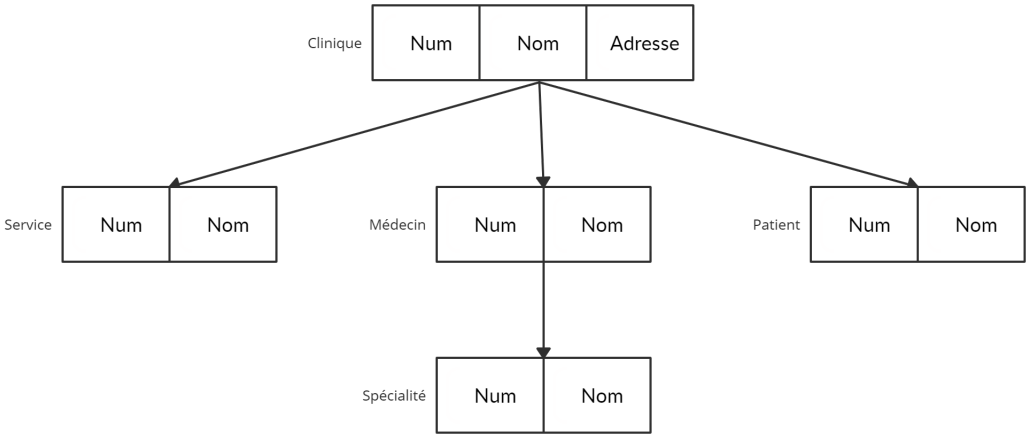

FIGURE 3.2 – Exemple d'une BDD hiérarchique dans domaine de la santé

Exemple 3.1 ***Exemple de BDD Hiérarchique***

```
    RECORD NAME IS VINS;
02 NV TYPE IS SIGNED PACKED DECIMAL 5;
02 CRU TYPE IS CHARACTER 10;
02 MILLESIME OCCURS 5 TIMES;
03 ANNEE TYPE IS SIGNED UNPACKED DECIMAL 4;
03 DEGRE TYPE IS BINARY 15.
    RECORD NAME IS PRODUCTEURS;
02 NP TYPE IS SIGNED PACKED DECIMAL 5;
02 NOM TYPE IS CHARACTER 10;
02 PRENOM TYPE IS CHARACTER 10;
02 ADRESSE
03 RUE TYPE IS CHARACTER 30;
03 CODE TYPE IS SIGNED PACKED DECIMAL 5;
03 VILLE TYPE IS CHARACTER 10;
```

3.2.2 Les bases de données relationnelles

Le modèle de base de données relationnelle a été conçu par E. F. Codd en 1970. Il peut être défini à l'aide des deux concepts suivants [13].

— Schéma : spécifie la structure (*nom de la relation, nom et type de chaque colonne*) ;

— Instance : une table avec des lignes et des colonnes.

Le modèle relationnel décrit les données d'une base de données comme étant stockées dans des tables [10]. Il définit de façon formelle, à partir de quelles tables les données sont accessibles et assemblées sans avoir à réorganiser les tables de la base de données. Cette façon de reprsenter les données est très familiére. Aussi, il est aisé d'interpréter la structure d'une base de données relationnelle au moyen de cette table [14]. Voir l'exemple ci-dessous 3.2.

Un système de gestion de bases de données relationnelle (SGBDR) est un système intégré permettant la gestion cohérente des bases de données relationnelles, comme le montre la figure 3.3.

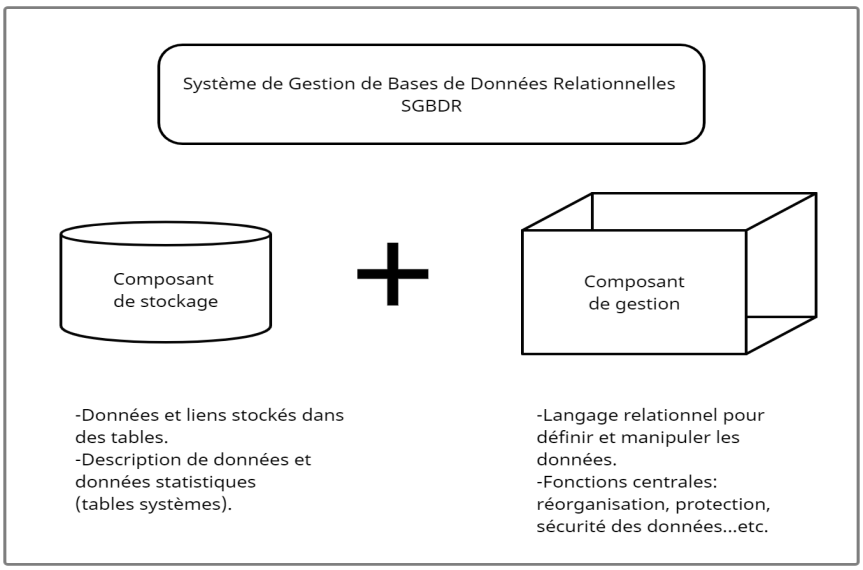

FIGURE 3.3 – Les deux composants d'un SGBDR

L'interface standard pour l'interrogation d'une base de données relationnelle est le langage SQL (**Structured Query Language**). Les commandes SQL sont utilisées pour interroger de façon interactive les informations contenues dans la base et pour rassembler des données dans le cadre de rapports [15].

Le principal avantage du modèle relationnel est qu'il assure la cohérence des données.

A titre d'illustration, considérons, la table Patient, qui peut inclure les champs suivants :

— NumP (numéro patient)

— NomP (nom patient)

— prenom (Prénom patient)

Les SGBD modernes, offrent plusieurs types prédéfini pour les champs d'une base de données (*par exemple, text, number, date*).

La figure 3.4 illustre un exemple d'un schéma d'une base de données relationnelle.

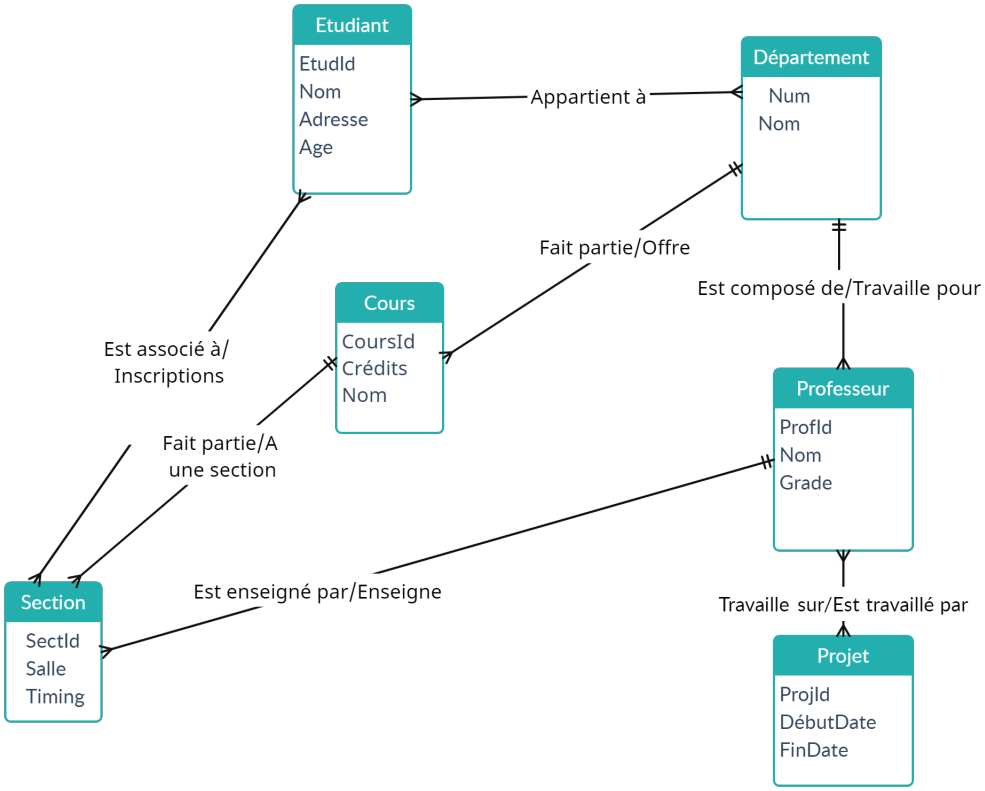

FIGURE 3.4 – Exemple d'une BDD relationnelle du domaine de la scolarité

Exemple 3.2 ***Code pour la création de tables d'une BDDR***

```
CREATE TABLE Patient
(NumP INTEGER PRIMARY KEY NOT NULL,
NomP VARCHAR (20),
NumM VARCHAR (20) );
CREATE TABLE Mutuelle
(NumMut INTEGER PRIMARY KEY NOT NULL,
NomMut VARCHAR (20) );
CREATE TABLE Médecin
(NumMed INTEGER PRIMARY KEY NOT NULL,
NomMed VARCHAR (20),
PrénomMed VARCHAR (20));
```

3.2.3 Les bases de données orientées objets

Définition 3.3 *[16]*
Une base de données orientée objet (BDOO) est une base de données dont le contenu est un ensemble d'objets ou instances des classes plutôt que des n-uplets des relations.

Une BDOO est caractérisée par des propriétés dérivées de l'approche orientée objet telles que :

- La représentation des structures de données complexes.
- Encapsulation de données et de traitements.
- L'héritage des classes.
- Le polymorphisme de méthodes.
- L'identificateur d'objet pour différencier entre objets mêmes s'ils ont la même valeur.
- La compatibilité avec les langages de programmation orienté objet tels que Java, C++.

Autrement, une BDOO est perçue comme un stock d'informations groupées sous formes de collections d'objets persistants. Les objets créés à l'aide de langage de programmation orientés objets sont généralement stockés dans des BDDR. Toutefois, en réalité, les bases de données orientées objets sont plus adaptées pour stocker ce type de contenu [17].

Les Langages d'interrogation des BDOO .
Dans ce contexte, il n'y a pas de solution largement répandue et on peut trouver des langages de programmation orienté-objet ou des langages spécifiques permettant l'interrogation directement la base de données de façon plus déclarative, comme SQL3 et ODMG. D'autre part, le langage **OQL** (Object Query Language) permet d'effectuer des requêtes sur les objets. Il est basé sur la syntaxe SQL [17].

L'exemple ci-dessous montre le code OQL pour la création d'une BDOO pour la gestion des prêts des ouvrages d'une bibliothèque.

Exemple 3.3 ***Exemple de BDOO***
```
class Livre(extent allLivre, key idLivre)
attribute long idLivre ;
attribute string titre ;
attribute string auteur ;
attribute date dateAcquisition ;
attribute date datePret ;
relationship Membre emprunteur
inverse Membre : :prets ;
relationship list<Reservations> reservations
inverse Reservation : :tupleLivre ;
void preter(in Membre emprunteur, in date datePret) ;
void retourner() ;
void reserver(in Reservation tupleReservation) ;
```

void annulerRes(in Reservation tupleReservation) ;
date dateRetour() ; class Membre(extent allMembres, key idMembre)
attribute long idMembre ;
attribute string nom ;
attribute long telephone ;
attribute short limitePret ;
relationship set<Livre> prets
inverse Livre : :emprunteur ;
relationship set<Reservation> reservations
inverse Reservation : :tupleMembre ;
. . .

Après ce survol du domaine des bases de données classiques, nous allons à présent exposer les bases de données NoSQL.

3.3 Les bases de données NoSQL

Le modèle relationnel manifeste certaines limites qui sont dues, essentiellement, à la gestion des données massives du Web qui a connu une révolution avec l'avènement des sites web à fort trafic tels que **Facebook, Twitter, Amazon et LinkedIn** (*Facebook doit gérer plus de 100 milliards de messages par mois et Twitter doit ingérer 7 téraoctets*[3] *de données par jour*). En effet, cette technologie de stockage s'avère inefficace pour la gestion des données massives. Le marché du Big Data regorge de solutions permettant de stocker et de manipuler de fortes volumétries de données, où les solutions dites NoSQL sont les plus populaires.

Les solutions qui couvrent ces problématiques font clairement le sacrifice d'une exploration aléatoire moindre au profit d'une optimisation des performances en termes de gestion et de stockage des données. Le choix se porte clairement sur des solutions de stockage aux **modèles les plus simples**, afin de minimiser les opérations de transformation des données avant l'écriture. Elles sont regroupées généralement sous le vocable **NoSQL**[4] pour "Not Only SQL".

NoSQL est une combinaison de deux mots : No et SQL. Cet acronyme pourrait être mal interprété car l'on pourrait penser que cela signifie la fin du langage SQL et qu'on ne devrait donc plus l'utiliser. En fait, le "*No*" est un acronyme qui signifie "*non seulement*", c'est donc une manière de dire qu'il y a autre chose que les bases de données relationnelles.

Aujourd'hui, le terme NoSQL englobe tous les SGBD **qui ne suivent pas la tendance de type relationnel**. Cela signifie que le NoSQL n'est pas un seul produit ou même une technologie unique. En effet, il représente de nombreux produits ainsi que plusieurs concepts de stockage et de manipulation de données. Leurs principaux avantages sont leurs **performances** et leur **capacité à traiter de très grands volumes de données**.

Nous allons, dans cette section, nous focalisé sur les trois types de bases NoSQL que sont les bases XML, orientées documents et les bases de données clés valeurs. La

3. 1 téraoctets = 1000 gigaoctets
4. Not only SQL

prochaine section sera consacrée aux bases de données graphes vue leur importance.

3.3.1 Les bases de données XML

Le langage de balisage extensible XML (eXtensible Markup Language) [18], est un sous-ensemble du Standard Generalized Markup Language (SGML).

L'objectif initial de XML etait de faciliter l'échange automatisé de contenus complexes (*arbres, texte enrichi, différents format de texte* ...) entre systèmes d'informations hétérogènes et résoudre, ainsi, les problèmes d'interopérabilité, en offrant des outils et des langages associés. En effet, XML constitue un langage de gestion de données semi structurées [19]. Aussi, il permet aux développeurs de créer leurs propres formats de stockage et de partage d'informations. En utilisant cette possibilité, les développeurs ont créé des documents représentant un nombre important de types différents d'informations. Un point essentiel de ce processus est la déclaration et la documentation formelle de ces formats qui sont nécessaires pour que les développeurs puissent réaliser leurs applications sur des bases solides [18].

Il est à noter que le langage XML est considéré comme le nouveau standard d'échange d'internet et des réseaux intranet/extranet. Aussi, actuellement, il prend une place de plus en plus importante dans l'informatique [20].

Les deux exemples, ci-dessous illustrent deux fichiers représentant des bases de données au format XML.

Exemple 3.4 ***Exemple simple de données en XML***

```
< ?xml version="1.0" ?>
<annuaire>
<personne id="p17">
<nom>Bochra</nom>
</personne>
< !– Ceci est un commentaire –>
</annuaire>
```

ou bien

Exemple 3.5 ***Base de données Livres en XML***

```
   <BIBLIOTHEQUE>
<LIVRE>
<AUTEUR>
<PRENOM> Mathias </PRENOM>
<NOM> Weske </NOM>

   </AUTEUR>
<TITRE> Business Process Management </TITRE>
<EDITEUR>
<NOM> Second Edition </NOM>
<ANNEE> 2012 </ANNEE>
</EDITEUR>
</LIVRE>
</BIBLIOTHEQUE>
```

Les langages d'interrogation des données XML

Xpath et XQuery sont connus comme standards pour l'interrogation des fichiers XML.

XPath est le langage de requêtes élémentaire dans XSLT (*eXtensible Stylesheet Language Transformations*). Il permet, notamment, de transformer un document XML dans un autre format, tel que PDF ou encore HTML pour être affiché comme une page web) [21]. Ses premières versions (**Xpath et XSLT**) ont été définies de manière à pouvoir fonctionner indépendamment de toutes compréhension explicite de la structure des documents manipulés. Mais cela à imposer des limites en termes de performances et de fonctionnalités. La connaissance de la structure des documents peut améliorer l'efficacité des programmes d'optimisation. La deuxième version de Xpath est XSLT ainsi que la première version de XQuery. Une spécification à venir qui définit un langage de réquêtes en XML- s'appuiras sur la disponibilité de schéma XML pour assurer les différentes fonctionnalités d'interrogation [18].

D'autre part, XQuery est un langage de requête informatique permettant non seulement d'extraire des informations d'un document XML, ou d'une collection de documents XML, mais également d'effectuer des calculs complexes à partir des informations extraites et de reconstruire de nouveaux documents ou fragments XML [22].

3.3.2 Les bases de données orientées documents

Une base de données de documents est un type de base de données non relationnelle conçu pour stocker et interroger des données sous forme de documents de type JSON (**JavaScript Object Notation**). JSON est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée) [23].

L'avantage des bases de données orientées documents est l'unité d'information et l'adaptation à la distribution. Aussi, la flexibilité, la richesse de la structure, l'autonomie est la sérialisation sont garanties avec les bases de données documents. En effet, cela évite de faire des jointures pour reconstituer l'information, car elle n'est plus dispersée dans plusieurs tables. Il n'y a plus besoin de transaction car l'écriture est suffisante pour créer des données sur un document ou pour modifier un objet. Une seule lecture est suffisante pour reconstituer un document. Aussi, les documents étant autonomes, on peut les déplacer facilement, du fait qu'ils sont indépendants les uns des autres. Néanmoins, la base de données documents présent plusieurs inconvénients, à savoir : la hiérarchisation d'accès, l'absence de perspective dans la base de données et la perte d'autonomie des entités.[24]

Exemple 3.6 ***Exemple de BDD orientée documents***
db.test.save(

"cours" : " NoSQL",
"chapitres" : ["familles", "CAP", "sharding", "choix"]
"auteur" :
"nom" :"Travers",

"prenom" :"Nicolas"
)

3.3.3 Les bases de données orientées clé-valeurs

Comme toutes les bases **NoSQL**, ce type se base sur le principe de stockage d'une valeur associée à une clé unique. Mais, contrairement aux autres bases NoSQL, la valeur associée à une clé peut être une simple chaine de caractère comme un document, ou encore un objet beaucoup plus complexe pouvant contenir une multitude d'informations.

Ces bases NoSQL sont parmi les moins répandues, mais aidée par leur simplicité de fonctionnement, elles commencent à prendre en notoriété.

Exemple 3.7 ***BDD livres avec le modèle clé valeurs***
"nom-ecrivain2" : "GRR Martin"
"pays-ecrivain2" : "USA"
"dateNaiss-ecrivain2" : 20/09/48
"titre-livre1-ecrivain2" : "Le trone de fer"
"prix-livre1-ecrivain2" : 15
"datePubli-livre1-ecrivain2" : "01/08/96"

3.4 Les bases de données orientées graphes

Les graphes étaient et restent un outil de modélisation d'un large éventail de phénomènes informatiques, tels que les réseaux d'ordinateurs, la recherche opérationnelle, l'intelligence artificielle et le génie logiciel, pour ne citer que ceux là.

Aujourd'hui, il y a un regain d'intérêt pour les graphes qui permettent de formaliser de nouveaux domaines émergents liés à l'informatique. Nous citons, essentiellement, les systèmes complexes, les systèmes bio-inspirés, la bio-informatique, réseaux de capteurs et réseaux sociaux.

En percevant, une base de données comme étant une collection inter-connectée d'un ensemble de données, les graphes s'avèrent très utiles pour modéliser ces données.

3.4.1 Rappel sur les graphes

Les graphes orienté

Définition 3.4 *Un graphe G= (X ;A). est défini par deux ensembles :*
un ensemble X= x1 ; x2 ; . . . xn dont les éléments sont appelés sommets , et un ensemble A= a1 ;a2 ; . . . am , dont les éléments sont appelés arêtes.

Définition 3.5 • *V un ensemble (fini ou infini)*

• *E une partie de V × V (i.e., une relation sur V).*
Le graphe G = (V ,E) est la donnée du couple (V ,E).
Les éléments de V sont les sommets de G .
Les éléments de E sont les arcs de G .
Si V est fini, on parlera de graphe fini.

Vocabulaire Soient V = v_i | i ∈ I et a = (v_i, v_j), i, j ∈ I
l'origine v_i et la destination v_j de l'arc a.

v_i et v_j sont les extrémités de l'arc a
a relie v_i à v_i.

Si b = (v_i, v_i) : b est une boucle.

Deux arcs adjacents ont au moins une extrémité en commun.

Les graphes Connexes

Définition 3.6 *Un graphe G = (X, U) est connexe si i, j ∈ X, il existe une chaîne entre i et j ; autrement dit, pour n'importe quels 2 somments, il existe un chemin.*

Arbres

Définition 3.7 *Un arbre est un graphe non orienté, acyclique et connexe.Et contient une racine [25].*

3.4.2 Fondement des bases de données graphes

Une base de données graphe (*BDDG*) est une **base de données spécifiquement dédiée au stockage de structures de données de type graphe**. Il s'agit donc de stocker exclusivement les données dans des nœuds et des arcs.

Par définition, « une BDDG correspond à **tout système de stockage fournissant une adjacence entre éléments voisins sans indexation** : tout voisin d'une entité est accessible directement par un pointeur physique.

Un nœud représente **une entité** (*personne, entreprise, film, activité ... etc*) et chaque arête (*arc*) est une connexion ou **une relation** entre deux nœuds (*amis de, parent de ... etc*) [26].

Chaque nœud d'une BDDG est aussi défini par un **identifiant unique**, **un ensemble d'arêtes sortantes et/ou entrantes**, ainsi qu'un ensemble de **propriétés** exprimées sous la forme de paires clé/valeur. Chaque arête se définit à son tour par un **identifiant unique**, un **nœud de départ** et/ou un **nœud d'arrivée**, ainsi qu'un ensemble de **propriétés** [27].

Définition 3.8 *Une base de données orientée graphe, ou graphe, est un type de base de données* ***NoSQL*** *qui utilise la théorie des graphes pour stocker, et effectuer des requêtes sur les relations entre les données.*

Les bases de données graphe sont constituées de nœuds et d'arcs. Chaque nœud représente une entité, et chaque arc représente une connexion entre les nœuds. Les bases de données graphes gagnent en popularité dans le domaine des analyses d'interconnexions. Par exemple, les entreprises peuvent utiliser une BDD graphe pour explorer les données sur ses clients à partir des réseaux sociaux.

Exemple 3.8 *Illustration d'une Base de donnéesGrpahes*

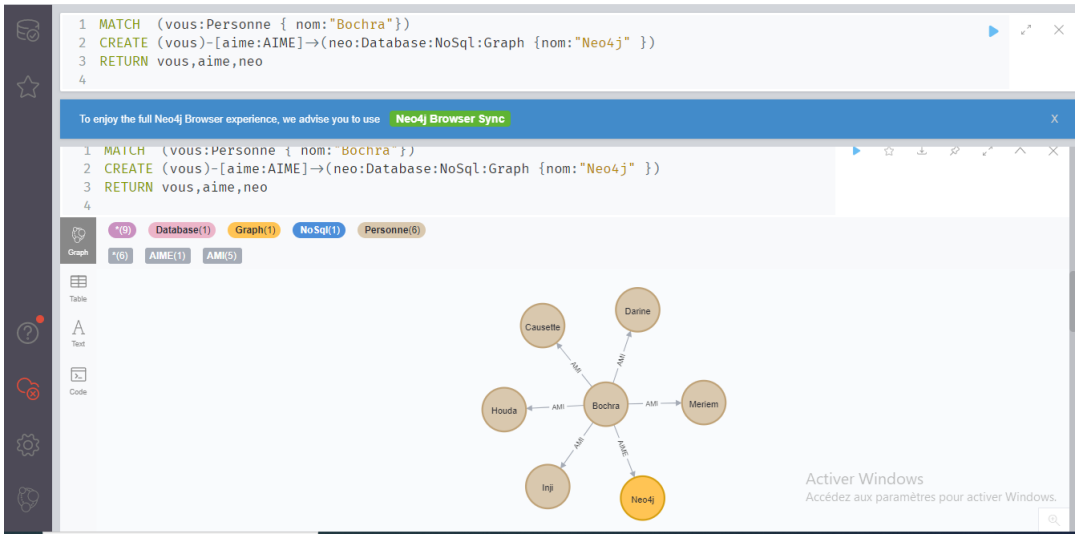

FIGURE 3.5 – Une BDD graphe

De plus en plus souvent, des bases de données jadis séparées sont combinées électroniquement sous forme de collections plus larges que l'on appelle les Data Warehouses. Les entreprises et les gouvernements utilisent ensuite des logiciels de Data Mining pour analyser les différents aspects des données. Par exemple, une agence gouvernementale peut procéder ainsi pour enquêter sur une entreprise ou une personne qui ont acheté une grande quantité d'équipement, même si les achats sont disséminés dans tout le pays ou répartis entre plusieurs subsidiaires.[28]

3.4.3 Avantages

Ces bases ont pour objectif de stocker les données en se basant sur la théorie des graphes. Elles s'appuient sur les notions de : (i) nœuds qui ont chacun leur propre structure, (ii) les relations entre les nœuds et (iii) les propriétés (de nœuds ou de relations).

Ce modèle de stockage facilite la représentation du monde réel, ce qui le rend particulièrement bien adapté au traitement des données des réseaux sociaux et géographiques par exemple, et de toutes les données fortement connectées de manière générale. Elles sont bien adaptées aux objets complexes organisés en réseaux, aux données présentant des dépendances fortes. Par ailleurs, elles permettent d'appliquer les algorithmes de théorie des graphes et la mise en place de visualisation de graphes nativement beaucoup plus rapides que les autres systèmes de stockage pour manipuler les données fortement connectées.

3.5 Discussion sur les modèles de stockage des données

Généralement, pour relier deux objets (ajout d'une relation) on doit créer une table n-tiers. Dans ce cas la solution est d'utiliser une base de données relationnelle, mais est ce que cela est le cas lorsqu'on veut stocker des données connectées ?

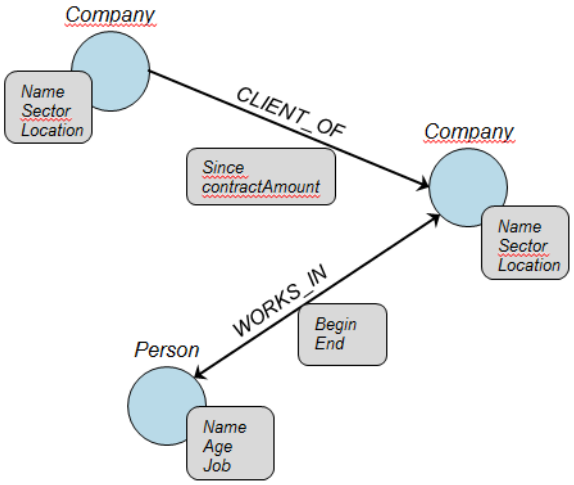

FIGURE 3.6 – EXEMPLE 2 BDD graphe

lorsqu'on crée une nouvelle relation, on applique quelques modifications sur le schéma de la base. Alors cette non adaptabilité (*flexibilité*) allonge les temps de développement. Par conséquent, les inconvénients des BDDR est que l'ajout d'une relation va impliquer un ajout de jointure ce qui rend les requêtes complexes et donc difficilement maintenables. Par conséquent, plus on a de données dans une table et moins bonnes sont les performances. Les moteurs SQL font le produit cartésien de chacune des tables pour calculer le résultat.

- les BDDR sont inappropriées, surtout si l'on veut faire du temps réel.
- leur schéma sont non flexibles.
- de mauvaises performances (en cas de jointure multiples).
- code complexes et difficile à comprendre.

Les développeurs ont, alors pensé à un nouvel axe pour les bases de données, qui sont les bases de données NoSQL(not only SQL) qui ne suivent pas tous le même schéma "schéma less". Le stockage de données est fait d'une manière souple contrairement aux données SQL qui fait avec précaution selon un schéma fixé avec contraintes. Ainsi, on résume les propriétés de données NoSQL dans les points suivants :

consistency : tous les nœuds voient la même version.

Availability : chaque requête donne une réponse.

Partition tolerence : la perte de messages n'empêche pas le système de continuer à fonctionner.

Gestion des informations incomplètes.

Traitement parallèle de données.

On précise que les bases de données orientées documents qui sont faites pour les bases de données actuels qui sont volumineuses et qui ne suivent pas le même modelé de données (XML, BDR, etc), la gestion de ces grosses données (Big data) est impossible via les SGBD existants. L'approche NoSQL est une solution permettant de traiter le

Big data. le NoSQL ne sont pas transactionnelles (ACID), alors, elles ne sont pas faites pour y stocker des données connectées.

Les spécialistes des données ne cessent de penser à une autre approche pour les bases de données, qui est une base de données orientées graphes. A cet effet, on a besoin de cette dernière lorsqu'on peut répondre affirmativement, au moins à deux questions parmi les suivantes :

- Vos données sont-elles dynamiques ?
- Vos données sont-elles connectées ?
- Avez-vous besoin d'un schéma flexible ?
- Devez-vous faire du temps réel ?

3.6 Conclusion

Vue l'importance des données dans tout les SI, dans ce chapitre nous nous sommes focalisés sur cet aspect en exposant les différentes technologies de gestion des données des processus métiers.

L'évolution des diverses technologies des BDD a été présentée, et un intérêt particulier a été accordé aux **bases de données graphes**. En effet, leur fonctionnement a été exposé et leurs avantages par rapport aux BDD classiques ont été mis en relief.

Dans le prochain chapitre, nous allons aborder les travaux connexes ayant traité le problème de la gestion des données des processus métiers. Ainsi, dans ce chapitre nous avons examiné les techniques de stockage et d'interrogation des données des processus métiers. Pour chaque technique, nous avons dressé une comparaison entre les différentes approches de stockage de ces données. Néanmoins, des nouvelles technologies, essentiellement, les BDDG qui peuvent offrir beaucoup d'avantages pour la gestion du cycle de vie des PM n'ont pas été exploitées, dans ce domaine.

Le prochain chapitre sera dédié à une étude de l'état de l'art des travaux qui ont abordé le problème de la gestion des données des PM.

Chapitre 4

Problématique et Travaux connexes

Introduction

Les deux premiers chapitres de ce mémoire ont été consacrés à la présentation des processus métiers et à l'exposé des technologies supportant la gestion des données. Ce chapitres est consacré à la présentation de notre problématique ainsi qu'à l'étude et l'exploration des travaux connexes ayant aborde la question de la gestion des données des processus métiers. Après l'exposé de notre problématique et les motivations qui ont conduit à ce travail, on abordera de manière détaillée les différents types de données des processus métiers. On termine le chapitre par une étude comparative des travaux qui ont traité la question de la gestion des données des processus métiers.

4.1 Problématique

Les processus métiers sont au cœur des systèmes d'information contemporains et les organisations investissent des sommes colossales pour leur gestion et leur mise à jours. En effet, la gestion du cycle de vie des processus métiers fournit aux entreprises une grande opportunité pour répondre aux besoins évolutifs de ses clients et aux exigences du marché.

A cet effet, les entreprise attachent un intérêt particulier à leur processus métiers et elles utilisent différents modèles pour prendre en charge leur cycle de vie (*modélisation , déploiement, exécution et supervision*). Dans cette perspective, la majorité des modèles utilisés fait une nette séparation entre les données et les traitements. Néanmoins, en réalité ces deux aspects ne sont que les deux faces d'une même réalité qui est la procédure de gestion spécifiant le processus métier en question. en définitif, le processus métiers englobe, à la fois, et les opérations (*flux d'activités*) et les données associés.

D'autre part, l'exécution des processus métiers déployés génère des instances de ce processus qui seront dotées de leur propres données. Avec la versatilité du Web et l'interconnexion accrue des réseaux d'ordinateurs, les processus métiers peuvent être invoqués par des milliers, voir des millions d'utilisateurs à la fois. En effet, dans les applications grand-public, telles que les applications de e-commerce, e-learning, e-gouvernment ou encore les bibliothèques électroniques, le volume des instances générées devient de plus en plus consistant. Ce phénomène engendre une masse de données importante dont la prise en charge devient problématique pour les organisations. Par ailleurs, les besoins de collaboration et de coopération inter-organisations induisent des besoins grandissant pour les échanges des données massives produites lors des invocations des processus métiers.

En général, les exécutions des processus métiers génèrent des données qui ont les caractéristiques suivantes :

— Un volume de données important qui croit de manière exponentielle avec le nombre d'instances déclenchées. En effet, chaque instance est dotée de ses propres données qui sont relatives à son exécution, aux ressources affectées et aux propriétés intrinsèques à l'instance elle même (Id, Time-stamp,...)

— Les processus métiers sont directement affectés par les changements qui surgissent dans l'environnement des entreprises, tels que les changements des lois et règlementations. Cela engendre le besoin de schémas (modèles) qui soient **évolutifs et flexibles** pour prendre en charge ces évolutions.

— Les données des processus métiers sont fortement connectées et inter-liées. Ce constat est dû fondamentalement à l'intégration des systèmes d'information, à l'interaction permanente entre sous-systèmes de gestion et aussi pour des raisons de coopération et d'échange entre entités organisationnelles. La mondialisation de l'économie et le phénomène de globalisation accélérés par les échanges à l'échelle planétaire ont fortement accéléré ces échanges.

De ce qui précèdent, se pose alors les questions inhérentes à la gestion de cette catégorie spécifique de données.

— Comment stocker les données des processus métiers ?

— Quels est/sont le/les modèles les plus adéquats permettant la description et l'interrogation des données des processus métiers ?

— Quels sont les technologies offrant le plus d'avantages en terme de performances (*espace et temps*) pour la gestion des données des processus métiers ?

— Quels mécanismes et quels outils pour gérer les données ?

Comme il a été expliqué dans le chapitre précédent, les bases de données de type relationnelles sont limitées et ne permettent pas de prendre de manière adéquate ce type de données. En effet, les exigences imposées par les données des processus métiers mènent vers une réflexion autour de modèles de données qui soient plus flexibles et dynamiques. Dans ce sens, il s'avère que les bases de données graphes apportent des réponses aux questions précédentes. En partant des avantages qu'offrent les bases de données graphes, tels que expliqué dans le chapitre précédent, ils permettent, potentiellement, à ces nouvelles exigences et offrent un atout majeur pour la gestion du cycle de vie des données des processus métiers (*création, stockage, interrogation, mise à jour, diffusion et archivage*).

A notre sens, l'utilisation des BDDG nous fournit des avantages évidents pour la gestion des données des processus métiers. Elle nous permet de :

— Gérer les données inter-connectées ;

— Surmonter le problème de l'explosion des jointures du modèle relationnel ;

— Offrent une réponse aux besoins de flexibilité et d'évolution des schémas ;

— Offrir de meilleurs performances.

Si nous considérons que le modèle adopté pour la représentation des processus métiers est celui des AFD, nous pouvons illustrer notre problématique par la figure 4.1 ci-dessous.

Dans cette figure on remarque qu'après la représentation des processus métiers par des systèmes de transition étiquetés (Labelled Transition Systems : LTS), les systèmes

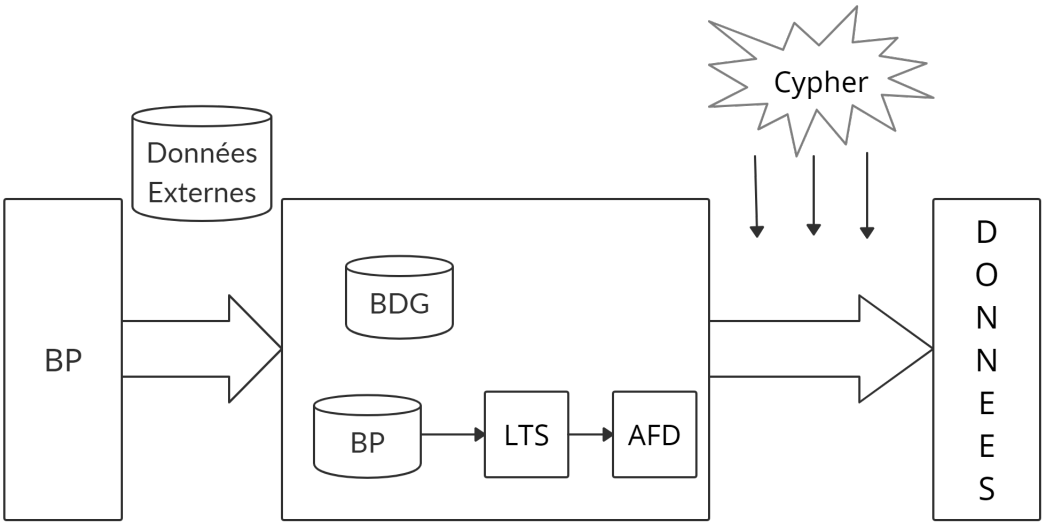

FIGURE 4.1 – Problématique de la gestion des données des PM

de bases de données graphes peuvent êtres déployés pour gérer les données afférents et la langage **Cypher** peut être utilisé pour interroger les données.

4.2 Motivations

Pour faire face aux défis inhérents aux données des processus métiers, nous proposons dans ce mémoire de faire la jonction entre deux domaines qui sont (i) le domaine des processus métiers et (ii) celui des bases de données graphes.

Cette jonction apporte une contribution importante au deux domaines précédents. En effet, d'une part, elle permettra de prendre en charge les données spécifiques à un domaine particulier par l'utilisation de la technologie des bases de données graphes. D'autre part, les bases de données graphes seront exploitées dans une perspective de généralisation à un domaine aussi particulier qui est celui des processus métiers. Cette dualité permettra de cerner plusieurs dimensions induites par les raisons suivante.

— **Gestion des données inter-connectées :** Le phénomène de la mondialisation de l'économie a entrainé des mécanismes d'échange qui sont de plus en plus liés. Cela a causé une interconnexion des S.I et, par conséquent des données gérées par ces systèmes. Il est fort constaté que les processus métiers actuels sont trans-frontaliers et traversent les frontières de l'entreprise. Ainsi, les partenaires, concurrents et clients sont de nature universelle. D'où le besoin de gérer des données qui sont fortement connectées.

— **Explosion du volume de données :** Le volume considérable des données générées lors des invocations des PM, surtout dans les applications grand public (e-gouvernement, e-santé, e-commerce, e-learning) devient de plus en plus volumineux. Dans ce type de processus métiers, des millions d'instances peuvent être en cours d'exécution à un instant. Par conséquents, les données gérées de-

viennent de type de données massives ou big data, exigeant des technologie et des mécanismes dédiés pour leur prise en charge.

— **Gérer des schémas flexibles des PM :** Comme les changements et les évolutions sont des caractéristiques intrinsèques à tout système qui doit prendre en charge les changements qui surgissent dans l'environnement, il s'avère impératif de disposer d'outils et de mécanismes permettant de prendre en considération cet aspect important des processus métiers. Dans cette perspective, une modélisation qui tient compte des facteurs de l'environnement (*lois et réglementations*) apportera une souplesse et une flexibilité des modèles des PMs. L'utilisation des bases de données graphes contribue de manière directe à cet aspect, du fait que ce type de SGBD tolère les modifications permanentes des modèles et schémas sans contraintes majeures.

— **Rehausser les données à leur niveau d'intérêt :** Contrairement aux autres approches de gestion du cycle de vie des processus métiers qui considèrent les données comme un aspect secondaire, dans notre approche nous mettons les données des PMs au même pied d'égalité que les flux d'activités. Dans ce sens, l'utilisation des bases de données graphe permettra de gérer, à la fois, les schémas sémantiques des activités sous formes de graphes (LTS), ainsi que leurs données descriptives.

Les motivations citées ci-dessus, renforcent notre conviction d'exploiter les bases de données graphes pour la prise en charge des données des processus métiers. A notre modeste connaissance, peu de travaux se sont orientés vers cette direction et nous estimons que notre contribution apportera une plus value, aussi bien pour le domaine des PMs que pour celui des BDDG.

Avant d'aborder les travaux connexes ayant traité, de manière plus ou moins rigoureuse, la question de la gestion des données des processus métiers, nous allons tout d'abord nous focaliser sur l'analyse des données des processus métiers en examinant leurs topologies et les caractéristiques.

4.3 Types de données des PM

En tant qu'évolution naturelle des systèmes de gestion des workflows classiques, la technologie de gestion des processus métier (BPM) s'est concentrée, initialement, à la conception et la documentation des processus de l'entreprise. En effet, les modèles des processus métiers proposés s'intéressaient principalement au séquencement des activités pour atteindre les objectifs métier, tout en s'occupant du flux de contrôle des activités. Par la suite, la technologie BPM a reçu beaucoup d'attention en tant qu'approche systématique et structurée pour analyser, améliorer, contrôler et gérer les processus métier. Cependant, la modélisation du flux d'activités devient insuffisante, et d'autres approches se sont orientées vers des modèles plus riches qui gèrent les processus métiers, tout en donnant une importance particulière aux données.

Dans cette perspective, il est constaté que les approches les plus récentes pour la modélisation des processus métiers consistent à la représentation de activités et des données en même temps par des modèles plus riches. Ces dernières peuvent gérer les données des processus métier d'une manière souple et flexible. Dans ce qui suit,

nous allons nous focaliser sur les différents types des données des processus métier qui peuvent être gérées par des modèles appropriés.

La gestion des données est essentielle dans les environnements BPM, d'une part pour permettre la description des BPs et d'autre part pour assurer l'exécution des instances. Dans [29] quatre catégories de données persistantes sont identifiées.

— Données descriptive (*spécification du schéma de PM*) ;

— Données historiques (*fichier logs ou log events*) ;

— Données d'exécution (*données métiers et état des ressources*) ;

— Données de supervision (*monitoring*) ;

Dans la suite de cette section, nous allons exposer une description de chaque catégorie des données précédentes.

4.3.1 Données descriptive (*schéma*)

Les données de schéma ce sont des données qui spécifient la description du comportement du processus métiers lui même. A titre d'illustration, ce type peut englober un tableau qui exprime les différentes transitions entre les états d'un automate définissant un processus métier, ou encore un graphe (états, transitions) qui exprime le fonctionnement d'un réseau de Petri relatif à un processus métier. Comme dernier exemple, ce type de données peut contenir un fichier au format XML qui décrit les états et les transitions d'un automate qui spécifie un processus métier.

4.3.2 Données historiques

Les données des historiques des exécutions d'un processus métier enregistrent fidèlement la progression des exécutions des opérations réalisées par chaque instance, du début de l'invocation du processus jusqu'à son état actuel, autrement dit une journalisation des exécutions.

Il faut rappeler que certains processus peuvent avoir à un instant données un nombre important d'instances qui sont soit achevées ou encore en cours d'exécution. Chaque instance ayant atteint un état d'exécution qui lui est spécifique. Avec le temps, le volume des données relatives à la mémorisation des exécutions peut être important et exige, alors, des moyens de stockage adéquat (sauvegarde, systèmes redondants, ...).

Ainsi, les informations captées par les systèmes d'information, lors de l'exécution des processus métiers, produisent une mine à valeur ajoutée qui requiert une importance capitale. Ces données de processus sont particulièrement pertinentes dans le contexte des processus métier automatisés, du contrôle des processus et de la représentation des actifs de base des organisations [30].

4.3.3 Données d'exécution

Comprend quatre sous catégories, à savoir.

— **Les données métiers :** Se sont des données qui sont essentielles pour la logique métier (*par exemple adresse de livraison pour un achat*, ou quantité disponible en stock).

— **Le statut d'exécution du processus :** permettent de mémoriser certaines information utile à l'exécution du processus (*par exemple la commande du client a été satisfaite*). Cela comprend à la fois des données conceptuelles (*significatives pour l'utilisateur*) et des données spécifiques à l'implémentation (*nécessaires pour effectuer les activités de base correctement (contraintes, règles, conditions, ...*).
— **Données sur les ressources** : du fait que l'exécution de chaque opération d'un processus métier nécessite la disponibilité de certaines ressources utiles à l'accomplissement des tâches, donc il est impératif de connaitre à tout moment les ressources disponibles dans le système (organisation). Ainsi, cette catégorie de données permet de suivre et de garder la trace des différentes ressources existantes, telles que les ressources matérielles (*e.g. ; disponibilité d'une imprimante ou un moyen de transport*), les ressources humaines (*un chauffeur ou un opérateur*) ou encore des ressources financières (*disponibilité des crédits pour l'approvisionnement du stock*).
— **Données de corrélation entre instances d'exécution**. Certaines données peuvent prendre en charge des contraintes liées aux interactions entre plusieurs processus de la même organisation ou inter-organisationnels. Par exemple, l'instance de processus métier commande du client Numéro 100 a généré trois instances du processus métier **livraison**.

4.3.4 Données de supervision

Tout système de gestion de processus métiers (**BPMS**) doit assurer les fonctions de supervision (*monitoring*) et d'audit des processus pris en charge. Afin d'assurer convenablement cette tâche, le système doit mémoriser les données afférentes et doit pouvoir les analyser, vérifier leur conformité et d'en extraire les informations utiles au bon fonctionnement du système. A titre d'exemple, le BPMS doit pouvoir détecter les processus qui sont inactifs depuis une certaines durée, les processus qui sont en situation de conflit sur une ressource, ou encore les processus dont certains chemins d'exécution (sous procédures) sont surchargés.

Après l'exposé des types des données manipulées par les processus métiers, la prochaine section est consacrée à l'étude des travaux ayant abordé cette question.

4.4 Étude comparative des travaux connexes

Dans cette section, nous analyserons les travaux connexes relatifs à l'utilisation des données des processus métiers. Nous commençons notre étude par les travaux de recherche du domaine académique, puis nous explorerons les suites logicielles du domaine industrielle, et enfin, nous terminerons notre étude par une analyse comparative des travaux existants.

4.4.1 Dans le domaine de la recherche académique

Beaucoup de travaux de la recherche académique ont abordé la question de la gestion et de la manipulation des données des processus métiers. On peut répertorier ces travaux en trois grandes catégories qui sont :

- La catégorie des travaux qui s'intéressent aux données du niveau des modèles (schémas).
- La catégorie des travaux qui traitent des données d'exécutions.
- Une troisième catégorie qui combine les deux précédentes (*analyse mixtes : schéma + exécutions*).

Ces catégories sont profondément examinées dans ce qui suit.

a) Gestion des données au niveau modèle (schéma)

Compte tenu de l'importance des modèles de processus métiers dans la gestion de leur cycle de vie, une panoplie de travaux ont porté sur les techniques de modélisation, d'analyse formelle et d'interrogation des processus métier [31] [32].

Dans [31], les auteurs ont proposé un cadre conceptuel pour l'interrogation et la réutilisation des modèles des PM dont les modèles sont basés sur des graphes. Dans cette approche un nouveau langage pour l'interrogation visuelle et graphique des processus métier appelé **BPMN-Q** a été proposé. Ainsi, ce nouveau langage **BPMN-Q** est utilisé pour filtrer et sélectionner les modèles des processus contenus dans une base de processus, en opérant un mapping (*correspondance*) entre les modèles stockées sous forme de graphes avec un graphe d'interrogation introduit en entrée. De plus, le travail élaboré est amélioré par un composant permettant l'expansion des requêtes par des attributs sémantiques.

Ainsi, **BPMN-Q** permet d'exprimer les requêtes structurelles et de spécifier des procédures pour déterminer si un modèle de processus donné est structurellement similaire à une requête. Cette dernière est considérée comme un graphe qui sera mis en correspondance avec un ou plusieurs graphes de processus.

Les points forts de ce système sont :

(i) Le cadre est basé sur un nouveau langage intuitif et visuel pour l'interrogation des modèles des PM avec un langage graphique.

(ii) Le cadre conceptuel supportant la langage BPMN-Q est amélioré par l'ajout de l'aspect sémantique qui utilise une dimension ontologique lors de l'analyse de correspondance.

(iii) En vue d'améliorer les performances du modèles proposé, le langage de requêtes SQLbackend utilise l'infrastructure d'indexation robuste disponible dans les SGBDR.

(iiii) L'architecture du cadre conceptuel proposé est conçue de manière très flexible et peut être facilement adaptée à d'autres notations de modélisation.

Le travail réalisé dans [32], les auteurs présentent un langage formelle pour l'analyse et la modélisation des PM. Ce langage est fondée sur une base sémantique et mathématique solide articulée sur le langage AMBER qui est basé sur les graphes. Ce langage a pour but d'analyser et de spécifier les caractéristiques des PM, d'identifier les goulots d'étranglement et de comparer les alternatifs en vue d'opter pour la solution la plus appropriée dans un contexte d'exécution donné.

Un autre travail intéressant [33], propose le système Testbed qui est basé sur un framework (*cadre conceptuel*) pour les systèmes distribués. Un autre système a été

développé à l'université de Twente [34] et porte le nom d'AMBER[1]. Ces deux travaux ont pour objectif l'analyse fonctionnelle et quantitative des modèles de PM modélisés par des graphes [32]. En effet, ils proposent de déterminer le chemin critique, l'analyse du temps de réalisation et l'analyse des files d'attentes des instances des PM. Par la suite, le langage de modélisation AMBERet les méthodes qu'il spécifie ont été implémentés dans un outil professionnel d'ingénierie des PM. Ce framewrok s'est avéré très utile et bénéfique dans de nombreux scénario et cas pratiques [33].

Par ailleurs, dan une perspective d'analyse de performances, le système proposé présente un algorithme de réduction des graphes afin de permettre de calculer les caractéristiques globales des PMs (*distribution du temps d'achèvement d'une instance, temps moyens de réalisation d'une activité, caractéristiques du chemin, ldots*). Les algorithmes décrits dans ce travail peuvent facilement de s'adapter à d'autres types d'analyse, telles que :

— L'analyse des coûts ;

— L'analyse des risques ;

— L'analyse d'achèvement ;

Pour faire face aux défis inhérents aux évolutions des processus métiers induites par les changements qui surgissent dans l'environnement, les auteurs dans [35] formalisent un langage déclaratif pour la gestion des évolutions dynamique des protocoles des services Web implémentant les processus métiers. Ce langage a l'avantage de donner aux fournisseurs de services la possibilité de définir d'une manière déclarative les contraintes qui gouvernent le processus de migration des instances actives et de spécifier, par là même, leurs propres stratégies de migration spécifiques à leur besoins.

b) Gestion et Analyse des données au niveau exécution

Les études portant sur l'analyse des processus métier de point du vue données se sont focalisées sur les données générées à partir des instances en cours d'exécution. A cet effet le principe de stockage des données des PM s'appuie sur les bases de données qui sont par la suite utilisées selon les méthodes existantes (*classiques, NoSQL*).

Contrairement à l'analyse précédente qui se concentre sur les contraintes imposées sur la spécification du processus métier (*c'est à dire son modèle*), les travaux d'analyse des processus métiers par rapport au niveau données s'intéresse uniquement aux données générées lors de l'exécution des processus métiers. Ces données sont souvent stockées dans des fichiers logs ou dans des bases de données adéquates.

Plusieurs domaines ont traité le stockage et l'analyse des données des processus métiers. Dans ce qui suit, une étude des domaines les plus significatifs est exposée.

b.1 L'entrepôt de données des processus métiers

Améliorer les procédures de travail est vital à toutes les entreprises. L'amélioration des processus métiers nécessite une analyse réelle des données collectées, comme première étape de base.

Dans ce contexte les auteurs dans [36] présentent les défis de l'analyse de processus qui sont les suivants :

1. AMBER :Architectural Modelling Box for Entreprise Redesign

— la conception de solutions ad-hoc pour chaque processus métier exécuté dans l'entreprise est trop coûteuse. Par conséquent, des approches génériques doivent être recherchées ;

— le niveau d'abstraction auquel les processus doivent être analysés est beaucoup plus élevé par rapport à l'information disponible dans l'environnement d'exécution du processus ;

— le besoin croissant de co-développement de l'analyse des processus et de la solution d'automatisation des processus ainsi que l'ampleur du problème rendent difficile la gestion des changements fréquents dans les sources des données de processus.

Pour relever ces défis, les auteurs ont proposé un modèle conceptuel pour l'entreposage de données de processus. Plus précisément, l'article apporte les contributions suivantes :

- Analyser et classer les exigences d'analyse pour l'entreposage de données de processus.
- Fournir un modèle d'entrepôt configurable qui peut répondre aux besoins des rapports complexes pour n'importe quel processus, en tenant également compte des contraintes de performance. Le modèle aborde les problèmes récurrents des clés, tels que le compromis entre le besoin de modéliser l'hétérogénéité (*chaque processus est différent*) et celui de définir une représentation uniforme pour tous les processus (*pour soutenir la réutilisabilité et l'analyse croisée des processus*).
- Montrer comment extraire des données de bas niveau sur les processus exécutés vers des vues de plus haut niveau du même processus qui soient adaptées à la création de rapports. L'approche est basée sur la définition de processus abstraits et ensuite la cartographie de la progression du processus vers des événements se produisant dans les systèmes sources.
- Décrire comment extraire, transformer et charger des données de processus, et en particulier comment maintenir de façon semi-automatique les procédures ETL [2] à la suite de changements dans les applications source.
- Montrez comment la solution peut être rapidement prototypée en utilisant un environnement d'émulation pour obtenir rapidement les réactions des utilisateurs.

Dans [37], les auteurs introduisent un nouveau formalisme grammatical qui est les grammaires S-graph pour le calcul des représentations sémantiques basés sur les graphes et qui est en cohérence avec les vues plus classiques de la construction sémantique.

L'article illustre aussi un certain nombre de grammaires jouets écrites à la main et esquissent (schématisent) l'utilisation des grammaires S-graph pour la construction sémantique pilotée par les données et en donne les clarifications nécessaires.

D'aprés ce document l'idée de l'analyse sémantique est de faire en sorte qu'un système apprenne automatiquement le mappage de la chaine de caractères de la représentation sémantique avec ou sans l'utilisation d'une représentation syntaxique ex-

2. ETL : Extract, Transform and Load

plicite comme étape intermédiaire.L'accent est donc mis sur les grammaires induites automatiquement.

L'entrainement de tel modèle basé sur les données nécessite des corpus sémantiquement annotés.

Des spécialistes ont montré comment adapter les analyses des dépendances pilotées par les données des arbres aux graphes.

Aussi, nous retenons que le document participe à combler le fossé entre la construction sémantique motivée linguistiquement et l'analyse sémantique par les données.

b.2 Approches basée sur Big Data (données volumineuses)

Les processus métier sont omniprésents et interviennent dans plusieurs secteurs : marketing, santé, gestion financière et bien sûr métier, et génèrent une quantité importante de données dites big data. Ces dernières années, la gestion des modèles et des données des processus métiers est très difficile. D'une part, les processus métiers doivent être puissants en termes de modélisation. D'une autre part, le support d'analyse de Big Data permet de trouver des connaissances appropriées pour adopter des modèles de processus métier.

La contribution de [38] est de proposer une framework pour faciliter l'amélioration des processus métiers basée sur l'analyse de Big Data. Cette framework décrit la surveillance des processus métiers depuis la phase de modélisation, le déploiement jusqu'à l'analyse des données pertinentes au moyen d'outils d'analyse des données volumineuses.

La première section de l'article est dédiée au BPM actuel et explique le cycle de vie des processus métiers. Après l'introduction des concepts du big data, les auteurs exposent le concept de 3V (**volume, variété et vitesse**) présenté par Laney en 2001, et expliquent le cycle de vie des Big Data. Ensuite, ils définissent l'analyse de Big Data(Big Data Analytics BDA), puis ils présentent les principales techniques adoptées pour analyser le Big Data. les auteurs mettent l'accent sur l'analyse de Big Data et son domaine d'activité applicable, et montrent la relation entre l'analyse des Big Data et l'analyse des processus métiers. Ils ont montré aussi l'insuffisance des systèmes de gestion de bases de données relationnelles (SGBDR), et favorisent les bases de données NoSQL [3], comme XLM, et citent ses points forts et ses avantages. En effet, les auteurs stipulent que les systèmes relationnels accordent un peu d'attention aux données semi-structurées et non structurées. Ils ne prennent en charge que les données structurées. De plus, les SGBDR évoluent avec du matériel coûteux, ce qui ne permet pas de gérer le volume croissant de données. L'architecture du framework proposé est expliquée en détails. Pour la conception de leur travail, ils ont choisi d'adopter le standard MongoDB pour stocker les données.

b.3 Les techniques On-line Analytical Processing (OLAP)

Après la phase de collecte et de stockage des données, différents types d'analyses s'imposent. Analyser les données des systèmes transactionnels consiste à les collecter dans un entrepôt de données (*à l'aide d'outils d'extraction, de transformation et de*

3. Not Only SQL

chargement ETL), puis à utiliser un outil adéquat pour découper les données selon différentes dimensions. On-line Analytical Processing (OLAP) fait référence à l'activité générale d'interrogation et de présentation de données textuelles et numériques à partir d'entrepôts de données et/ou de data-marts à des fins d'analyse.

Le terme OLAP a suivi le développement du concept de base de données standard Online Transactional Processing OLTP. OLTP fait référence à l'activité générale de mise à jour, d'interrogation et de présentation des données textuelles et numériques des bases de données à des fins opérationnelles. En d'autres termes, OLTP englobe toutes les transactions quotidiennes effectuées sur les systèmes de base de données opérationnels.

Les trois fonctionnalités OLAP de base utilisées régulièrement par les analystes sont communément appelées :

- Slice et Dice
- Pivot (Rotation)
- Drill Down et Drill Up

Les outils OLAP sont désormais une partie essentielle du processus de prise de décision pour chaque organisation qui collecte de grandes quantités de données. Plus les données accumulées dans ses opérations sont importantes, plus les capacités OLAP essentielles deviennent importantes pour une organisation. L'attrait universel de OLAP réside dans la simplicité de la structure et de la conceptualisation, et par conséquent, dans sa simplicité d'utilisation.

A titre d'illustration de l'importance des travaux ayant abordé l'analyse des données des processus métiers au niveau données, dans [39], les auteurs donnent un aperçu des techniques OLAP et expliquent comment elles sont utilisées pour l'aide à la décision.

Dans ce travail, les auteurs présentent les fonctions spécifiques d'un système OLAP et ces plates formes. Ils spécifient aussi, la connexion entre les systèmes OLAP et les référentiels de données analytiques. Ensuite, un aperçu des fonctionnalités communes à tous les outils OLAP est présenté.

Dans cette section on a montré comment les techniques OLAP sont liées à des données référentiels analytiques, tels que les entrepôts de données et les data marts, et qu'elles offrent à l'utilisateur un moyen efficace d'accéder aux données des processus métiers pour faciliter l'analyse de données et permettre une prise de décision de gestion efficace.

A partir de cette étude on peut conclure que, naturellement, l'approche OLAP a ses limites. Une nouvelle approche pour le stockage, l'interrogation et l'analyse des données est inévitable. Cependant, dans la plupart des scénarios liés aux activités, ces méthodes complexes ne doivent être appliquées qu'après l'analyse basée sur OLAP. Ainsi, Une analyse plus approfondie révèle que le modèle dimensionnel est essentiel pour OLAP. Si les données sous-jacentes n'étaient pas organisées de manière dimensionnelle, avec une table de faits au centre connectée à un certain nombre de tables de dimensions, les trois opérations OLAP de base ne pouvaient pas être exécutées efficacement. De ce qui précède, d'autres techniques de stockage et par conséquent d'analyse doivent entrer en force pour prendre en charge les besoins des gestionnaires de processus métiers.

Dans [40] les auteurs ont montré que l'examen des données relatives aux historiques

d'invocation des PMs par les techniques d'analyse formelle peuvent contribuer à la refonte des PM. Ces technique permettent d'obtenir le choix de la solution la plus efficace et de savoir si les objectifs de re-engineering et de maintenance, telles que fixées au préalable sont toujours réalistes.

La refonte d'un PM est une activité complexe et implique des spécialistes de différentes discipline et une nouvelle technologie est introduite (changement radicaux, manque argent et temps...).

c) Analyse des travaux mixtes (schéma et données)

Dans cette catégorie de travaux de recherche, certains travaux se sont intéressés à l'extraction des modèles de processus métiers à partir des traces d'exécution. L'objectif est d'arriver à formaliser des modèles abstraits, en explorant les journaux d'exécution.

- Dans [41], l'auteur présente une approche pour la découverte d'informations sur les PM à partir de leur journaux d'événement relatifs à leur exécution. Il définit un algorithme pour calculer un graphe nommé Directly Follow Graph (DFG) à partir de la base de données graphe. cette représentation, nous offre de meilleures performances par rapport aux techniques habituelles.

En effet, la quantité croissante de journaux d'événements pose des problèmes aux techniques actuelles d'exploration des processus qui tendent à changer les données de la mémoire d'un ordinateur et introduisent des risques par manque de capacité et de faisabilité de gestion et de stockage des données, et cela quand on les applique à un grand volume de données dans un seul ordinateur d'où la nécessité de résoudre cette difficulté et formaliser aussi une nouvelle approche pour le stockage et la récupération des journaux d'événements dans/depuis de base de données de graphe.

Ainsi, est défini un algorithme pour calculer le graphe à suivi direct (Directly Follow Graph(DFG)) à l'intérieur de la base de données de graphe déplaçant ainsi les parties de calculs lourdes de l'exploration des processus dans la base de données graphe.

En outre, le calcul de DFG supprime la nécessité de déplacer les données dans les ordinateurs des analystes permettant ainsi d'utiliser la capacité de gestion des données dans les bases de données de graphe.

Aussi, cet algorithme (DFG) permet de :

i) supprimer l'obligation de déplacer les données dans l'ordinateur des analystes,

ii) Appliquer le contrôle d'accès à grain fin (*la partition*) des bases de données graphiques sur les journaux d'événements et de préserver la confidentialité tout en appliquant l'exploration de processus ;

iii) Mettre à l'échelle le calcul de la DFG selon une méthode déterminée (*verticalement et horizontalement*).

Il est à noter que cette approche est implémenté dans Neo4j et ses performances sont évaluées par rapport aux techniques actuelles en utilisant un fichier journal réel. Il est à constater, de même, que le grand volume de journaux permet de découvrir davantage d'informations sur les PM, mais soulève également certains défis, tels que la faisabilité, les performances et la gestion des données d'où la nécessite de palier et de résoudre ces difficultés par les développeurs concernés.

- Dans [42], les auteurs présentent une approche articulée sur un algorithme efficace pour la vérification visuel du graphe de flux de travail (*Workflows*) et pour le réduire

progressivement. L'algorithme utilise un ensemble de règles de réduction de graphe pour identifier les conflits structurels dans les modèles de processus pour le langage de modélisation de flux de travail donné ainsi que des indications sur la correction et la complexité des processus de réduction.

Il est à noter que dans ce travail, il découle qu'un modèle de processus intègre différents scénarios possibles d'exécution du processus par le biais de la modélisation structurelle. Un sous-graphe d'instance d'un modèle de processus représente l'un des scénarios d'exécution possible d'un modèle sous-jacent avec des sous-graphes correspondants.

Il faut signaler dans ce contexte que le calcul de nombre de sous-graphe d'instances est un problème complexe et que le nombre de sous-graphes d'instances possibles peut croitre de manière exponentielle en fonction du nombre et de l'emplacement des structures de choix, de fusion, d'éclatement et de synchronisation.

- D'après [43], l'un des points essentiels des entreprises est la conception organisationnelle basée sur les processus en vue davantages concurrentiels. Elles doivent être en mesure de gérer leurs modèles de PM et de faire face aux problèmes pour exister, en particulier, lorsque les modèles de PM sont exploités par les rétro-ingénierie (à partir des SI [4] par exemple), cela en modifiant la structure interne des modèles des PM sans en modifier le comportement externe. Les représentations standards, telles que BPMN fournissent aux entreprises un moyen pour gérer leurs PM, c'est à dire d'analyser, d'exécuter et d'adapter leurs PMs d'une manière efficace.

- L'une des techniques les plus appliquées et les plus éprouvées est le refactoring des modèles de PM qui améliore leur compréhensibilité et leur maintenance. Dans cette perspective, l'approche proposée dans le cadre du projet **IBUPROFEN** (Improvement and BUsiness Process Refactoring OF Embedded Noise) est une approche de refactoring de PM basée sur les graphes qui a été conçu pour les modèles de PM.

IBUPROFEN définit un ensemble d'algorithmes regroupés en trois catégories en fonction du défi de l'assurance qualité qu'il aborde. Les critères de qualité sont les suivants.

- La maximisation des éléments pertinents.
- La réduction des éléments à grain fin.
- La réduction de la taille des échantillons et l'amélioration de la qualité.

Ce document décrit, enfin, comment les modèles de PM sont gérés sous forme de graphes et comment ils sont remaniés selon l'ensemble des algorithmes basés sur les graphes dont IBUPROFEN.

- Dans le cadre des travaux utilisant les techniques du modèle checking, dans [44], les auteurs nous présentent le model checking explicite qui est un parcours par force brute (*par l'essai*) de tous les états possibles du modèle qui permet d'affirmer si une propriété est satisfaite ou non. Ce travail présente l'avantage d'interroger les traces par le langage KriQL qui est un langage de requêtes travaillant sur les traces et les systèmes de transition étiquetés sous-jacents.

Dans ce travail, une propriété est satisfaite, si l'algorithme de modèle cheking proposé produit une trace réelle comme contre exemple.Ainsi, l'interprétation des traces

4. SI : Système d'Information

aidera à l'interprétation du problème, étant donné qu'elle supporte les outils de visualisation et de diagnostique.

La contribution la plus significative dans ce travail est qu'il évalue différentes implémentations de KriQL, principalement, l'utilisation des **BDDR** et des **BDDG** pour la gestion des systèmes de transition.

De même, le langage Cypher est présenté comme un langage prometteur pour l'interrogation des bases de données graphes.

- Dans [45], les auteurs ont évalué les performances de plusieurs projets de BDDG natives, évolutives et ont montré que le SGBDG **Neo4j** est l'un des SGBD orientés graphes les plus efficaces. En effet, suite à une implémentation des différents systèmes au moyen de SGBD Progres et une autre au moyen du SBDG Neo4j, d'aprés [44], aucune d'entre elles isolément n'a pu aboutir complètement aux résultats concrets attendus, lorsqu'elle est prise isolément. Ce constat a imposé aux concepteurs de réaliser une implémentation mixte avec les deux SGBD pour atteindre les résultats escomptés. Néanmoins, cette implémentation mixte comporte aussi des inconvénients résiduels, dont on peut citer, par exemple, le manque de synchronisation entre les deux systèmes.

- Dans [46], il est constaté des difficultés dans la modélisation des PM (suite à l'ajout d'autres tâches, cycle de vie des données,...) et la détection des erreurs avec un état de chemin qui peut devenir incontrôlable.

Ainsi, les méthodes formelles permettent de détecter les erreurs des la première phase de modélisation et les erreurs détectées sont expliquées par un chemin allant de l'état initial à l'état d'erreur qui peut s'allonger et devenir ingérable en terme de performances (consommation de beaucoup de temps).

L'étude [46] propose une nouvelle explication des erreurs et donne une nouvelle approche qui, au lieu de s'intéresser à énumérer les actions sur le chemin de l'erreur, s'intéresse seulement aux décions qui y conduisent qui sont signalées et mises en évidence dans le modèle original.

Réf.	**Principe**	**Modèle Proposé**	**Gestion des données**	**Modèle de gestion des données**	**Bilan**
[31]	Technique basait sur le nouveau langage BPMN-Q pour l'interrogation des modèles de PM.	BPMN, BPEL, UML	Oui, Schéma, Exécution	BDDR	Cette technique peut être amélioré et offre une flexibilité d'utilisation.
[40]	Une approche qui s'interèssent aux techniques d'analyse formelle des PM	RdP	Oui, Données	BDDR	Constat que Le processus de refonte dans la conception des PM peut permettre la réduction des couts et des risques
[32]	La modélisation d'un langage AMBER sert particuliérement à identifier les goulots de tranglement	AMBER	Oui, Schéma	BDDR	Peut être facilement transformé en une représentation graphique permet d'apporter les solutions appropriées.
[41]	La conception d'un algorithme Directly Follows Graph (DFG) qui permet de déplacer les parties de calculs lourdes.	BPMN	Oui, Schéma, Données, Exécution	BDG	Nous offre de meilleures performances par rapport aux techniques actuelles.
[42]	Présentation d'une approche et d'un algorithme de vérification visuelle utilisant un langage générique pour éliminer les conflits de synchronisation.	RdP	Oui, Schéma	BDDR	Technique qui permet de corriger les conflits dans les modèles de processus.
[37]	Introduction d'un nouveau formalisme qui est les grammaires S-graph.	S-graph	Oui, Données	BDDR	Utilise les grammaires S-graph pour induire l'apprentissage automatique des systèmes.

[43]	Constat de la capacité des entreprises à gérer leurs modèles de PM et leurs problèmes en relation avec la qualité.	BPMN	Oui, Données, schéma	BDR	La schématisation des modèles des PM est un moyen trés efficece de gérer les données
[47] [44]	Vérification de modèle explicite qui est une traversée par force brute pour savoir si une propriété est satisfaite ou non.	Model checking, LTS	Non	BDDR, BDG	objectif d'interroger les LTS dans un outil frontal au moyen de modèle-cheker.
[46]	Constat de difficultés dans la modélisation des PM (ajout d'autres tâches :cycle de vie des données ...). Détection des erreurs avec un état de chemin qui peut devenir incontrôlable.	RdP	oui, schéma	BDDG	

TABLE 4.1 – Tableau d'évaluation des travaux existants

4.4.2 Dans le domaine industriel (Industrie du logiciel)

Aujourd'hui, les systèmes de gestion de processus métiers (BPMS) ont émergé en tant que technologie de pointe permettant la prise en charge des différentes phases du cycle de vie des processus d'entreprise. Ainsi, les fournisseurs de logiciels investissent des sommes colossales pour offrir aux organisations des solutions adéquates garantissant l'automatisation de leur processus métiers.

En exploitant des moyens techniques appropriés, les BPMS permettent, d'une part, de définir les spécifications relatives aux modèles (*schéma*) et d'assurer leur évolution, et d'autre part de gérer les données générées lors des interactions des clients. En fin, ils assurent aussi certaines fonctions de supervision et d'administration. Ainsi pour assurer une place prépondérante, divers concepteurs ont largement investi ce marché. On peut citer les solutions les plus utilisées : ARIS (Software AG), IBM Websphère, Bonitasoft, Oracle BPM, SAP Netweaver BPM, JBoss et jBPM).

La figure ci-dessous 4.2 illustre les répartitions principales des outils BPM les plus utilisés par les entreprises concernées.

L'utilisation adéquate des outils BPMS doit, en particulier, satisfaire les attentes des clients et améliorer les performances de l'entreprise avec une réduction des coûts.

Il est à signaler que l'approche BPM est plus utilisée dans l'industrie manufacturière avec la production de produits standardisés que dans le secteur produisant un résultat immatériel [50].

Nous pouvons classer les logiciels utilisés par les entreprises en deux types : les logiciels propriétaires dont on peux citer :(Oracle BPM, IBM Business Process Ma-

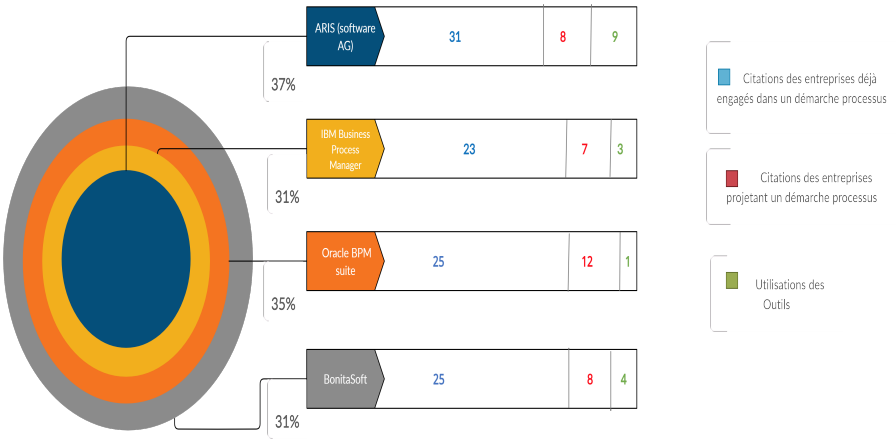

FIGURE 4.2 – Répartition des principaux outils BPMS

nagement, SAP Netveaver BPM) et les logiciels Open Source (Bonita Soft, JBoss jBPM).

Dans ce qui suit on va présenter ces deux grandes familles et les outils qu'elles englobent.

1) Les logiciels propriétaires Ce sont les logiciels dont l'utilisation exige une licence d'utilisation. Cette famille regroupe les systèmes suivants.

a) Oracle BPM Suite Oracle BPM Studio est un composant d'Oracle BPM Suite où les analystes de processus peuvent concevoir des modèles de processus à partir de son environnement convivial qui prend en charge Business Process Management Notation (BPMN)2.0.

Il nous fournit aussi un ensemble complet d'outils qui nous permettent de créer, d'exécuter et d'optimiser les PM. Cela a pour but d'automatiser et d'optimiser les PM qui peuvent être utilisés lors de l'exécution d'oracle BPM.

Parmi les fonctionnalités principales d'oracle BPM Suite on peut citer.

- Éditeur de processus Drag et Drop.
- Gestion des priorités.
- Gestion des problèmes.
- Gestion des tâches et alertes.
- Indicateurs de performance.
- Processus de validation.

Aussi cet outil nous permet de gérer les données des processus métier sous forme de bases de données relationnelles.

L'exemple de la figure 4.3 illustre l'utilisation de cet outil.

Exemple 4.1 *Utilisation d'Oracle BPM*

b) IBM Business Process Management IBM-BPM est un logiciel de gestion des PM pour les entreprises et les professionnels.

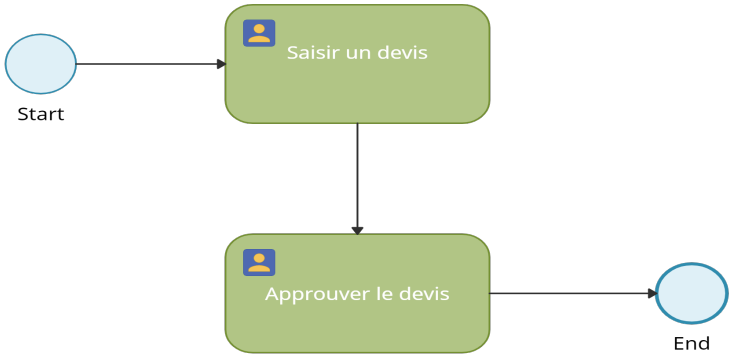

FIGURE 4.3 – Exemple d'un processus métier en utilisant Oracle BPM

Cet environnement, compatible avec tout les systèmes d'exploitation en utilisation, est une plateforme complète de gestion des PM qui fournit un ensemble robuste d'outils pour créer, tester, et déployer des processus métier. Elle offre une visibilité totale et permet de comprendre la gestion de ces PM.

Ce logiciel est très populaire chez les utilisateurs, en comparaison avec les autres logiciels qui offrent les même services. Aussi c'est un logiciel simple à installer des points de vue : paramétrage, configuration, personnalisation.

Il nous offre une interface API qui permet, par exemple, de se connecter à une base de données, d'échanger des données ou même de synchroniser les fichier entre plusieurs programmes informatiques via une extension. Les données de ce progiciel d'entreprise sont hébergées sur un serveur informatique (*stockées dans un data center*). Ce logiciel utilise diverses méthodes pour découvrir, modéliser, analyser, améliorer et organiser les PM.

Il présente, en particulier, les principaux avantages suivants :

- Concevoir et gérer les PM.
- Modéliser et mettre en œuvre les règles métiers.
- Cartographie organisationnelle avec graphiques et diagrammes.
- Créer et exécuter les applications basées sur des processus.
- Analyse en temps réel et surveillance des performances.
- Gestion de contenu et stockage de documents commerciaux.

La figure 4.4 ci-dessous illustre l'utilisation de cet outil.

Exemple 4.2 *Exemple d'utilisation de IBM BPM*

c) SAP Netweaver BPM L'outil SAP Netweaver BPM permet aux entreprises de concevoir leur processus métiers, de les déployer, d'identifier les goulots d'étranglement et enfin d'optimiser les processus en supprimant ces goulots d'étranglement et en augmentant ainsi la transparence, l'efficacité et l'évolution.

SAP Netweaver BPM offre les fonctionnalités suivantes :

- Conception/déploiement des processus.

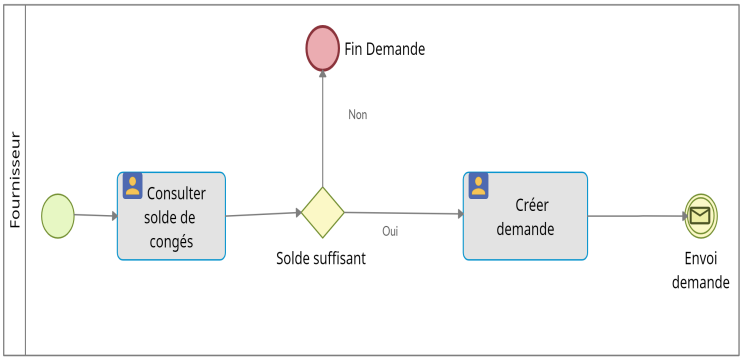

FIGURE 4.4 – Exemple d'un processus métier en utilisant IBM BPM

- Intégration étroite avec SAP-BRM (Business Rules Management).
- Interface utilisateur dynamique.
- Délégation/répartition des tâches.
- Interaction humaine (*faciliter les activités humaines*).
- Surveillance des processus (*signalement des anomalies...*).

Les avantages constatés de SAP sont :

- Transparence.
- Perfectionnement des processus.
- Centralisation des données.
- La responsabilité associée aux tâches réalisées.

SAP NetWevear Composition Environnement (CE) est conçu et mis en oeuvre comme un type d'utilisation de la pile JAVA de SAP NetWevear. Il vise deux domaines distincts :

- SAP NetWevear CE permet le développement guidé par le modèles de pratiques propres, également appelées applications composites.
- Les clients sont en mesure de concevoir, de déployer et d'exécuter les application JAVA avec SAP NetWevear CE en respectant les norme JEE[5][51].

Exemple 4.3 *Exemple de Processus métiers avec SAPNet weaver*

2) Les logiciels Open Source

a) Bonita Soft

Le projet de Bonita Software (BonitaSoft) a été créé afin de proposer une alternative Open Source aux solutions commerciales existantes dans le domaine BPM (Business Process Managment). Il est l'un des logiciels les plus utilisés. En effet, il est devenu l'un des principaux logiciel BPM Open Source et connait une ascension internationale (France, Espagne, États-Unis...).

5. Java Enterprise Edition

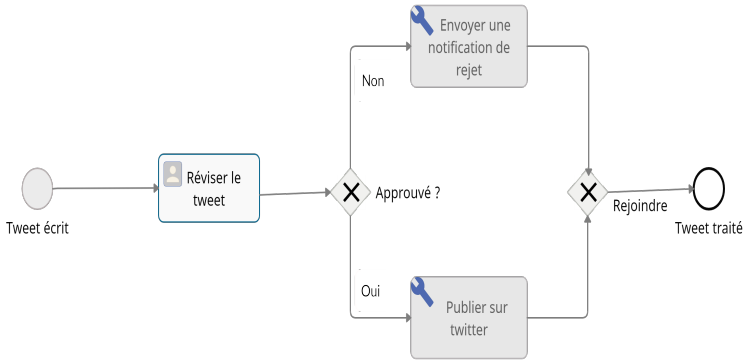

FIGURE 4.5 – Exemple d'un processus métier en utilisant SAP NetWevear

Bonita est une solution Open Source dédiée à l'automatisation des PM. Cette plateforme permet la création des applications basées sur des processus qui peuvent être partiellement ou entièrement automatisés.
Cette solution s'articule autour de trois composants :

- Un studio de modélisation de processus "Bonita Studio", qui fait concurrence avec le standard Business Process Management Notation (BPMN) et qui offre une solution graphique simple et intuitive. Cet éditeur de processus s'accompagne également de nombreux connecteurs vers des bases de données relationnelles (PostgreSQL, MySQL, Sybase...), des annuaires, des services web (Twitter, Facebook, Google...) ainsi que vers d'autres solutions logicielles (Jboss Drools,...).
- Un moteur d'exécution des processus qui est assez flexible et adaptable à diverses architectures de systèmes d'information.
- Une interface utilisateur de contrôle qui permet d'exécuter et de vérifier le résultat des processus.

Il est à remarquer que Bonita dispose de principes forts qui permettent en particulier de :

- Donner la possibilité aux utilisateurs d'avoir une certaines autonomie pour aboutir à la spécification des instances d'utilisation (*cas*).
- Offrir la possibilité d'arrêter, de reprendre ou de modifier l'action à tout moment sans avoir à prédéfinir le chemin.
- S'adapter au contexte de travail.
- Centrer sur les données : l'exécution des instances n'est pas centrée sur un processus strict, mais sur les données de l'instance.

Il est à remarquer que l'outil Bonita nous permet la création d'une application Bonita en suivant les étapes ci-dessous :

(i) Concevoir graphiquement un ou plusieurs processus en utilisant la notation BPMN.

(ii) Définir le modèle de données en utilisant la fonctionnalité de gestion des données métier de Bonita (*on peut utiliser notre propre base de données en cas de besoins*)

(iii) Créer des interfaces utilisateur web en utilisant l'éditeur d'interface utilisateur de Bonita.

(iiii) Définir les utilisateurs impliqués dans le processus.

(iiiii) configurer des connecteurs pour intégrer Bonita au système d'information (*ex : appeler un web service,...*)

Aussi, Bonita Studio nous fournit les fonctionnalités pour modéliser :

- Les processus BPMN (Business Process Modeling Notation)
- Le BDM (Business Data Model)
- Les interfaces utilisateur (UI Designer)
- Les applications

Par ailleurs, l'outil Bonita nous fournit aussi un serveur Bonita intégré. Il est didié principalement au développement.

La figure 4.6, ci-dessous, montre l'architecture et les composants de Bonita soft BPM

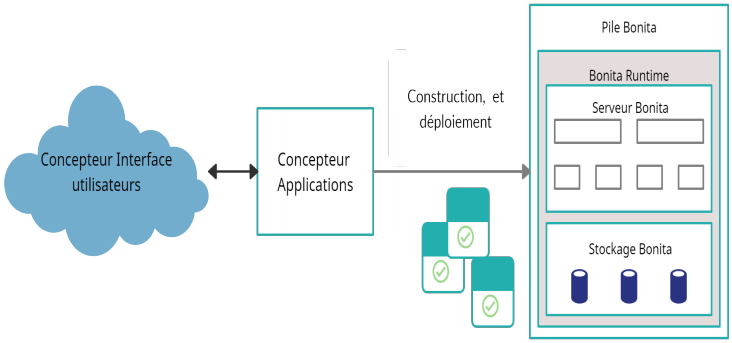

FIGURE 4.6 – Architecture de l'outil Bonita soft BPM

Exemple 4.4 *Exemple de Processus métier avec Bonita*
La figure 4.7 montre l'utilisation de Bonita pour la gestion des demandes d'achats.

b) JBoss jBPM

JBoss jBPM a été défini et décrit comme étant la réponse à la grande déception de l'industrie à l'égard de la technologie des workflow et BPM. C'est un outil permettant de rendre la technologie BPM plus accessible et plus facile à appliquer. Il constitue, ainsi, une technologie émergente et prometteuse.

jBPM (Java Business Process Management) est une plateforme de flux de travail (workflow), d'automatisation des processus qui permet la coordination entre les applications et des services divers, ce qui entraine le déploiement de nouveaux processus

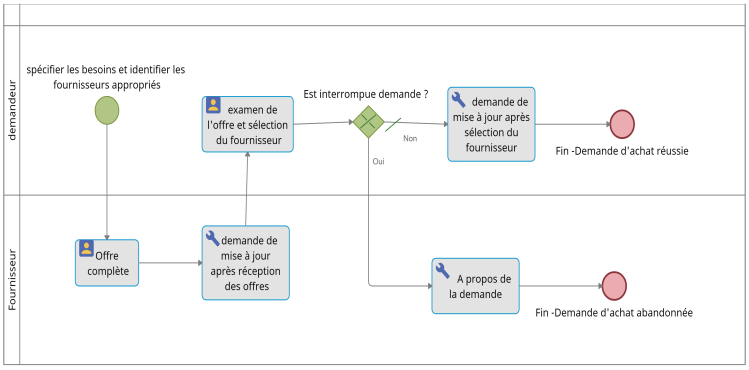

FIGURE 4.7 – Exemple d'un processus métier en utilisant Bonita soft

métiers générateurs de revenus. jBPM est utilisé en conjonction avec JBoss Application Serveur 4.2 et Eclipse EE 3.4.1. Aussi, ma boîte "jBPM Process Engine" est le noyau central qui gère automatiquement les processus définis et leurs interactions avec les événements externes [52].

De plus, JBoss jBPM est devenu le système de gestion de flux de travail le plus facilement utilisé par les applications commercialles dans le domaine de flux de travail open source. Le mécanisme de JBoss jBPM est élaboré à partir de quatre aspects : définitions de processus, mecanisme de planification de processus, mécanisme d'exécution de processus et gestion des instances de processus. En plus, d'autres fonctionnalités existantes, telles que créer, déployer, exécuter et surveiller les processus... sont offertes par le systèmes.

Dans la figure 4.9, une interface de JBoss est exposée.

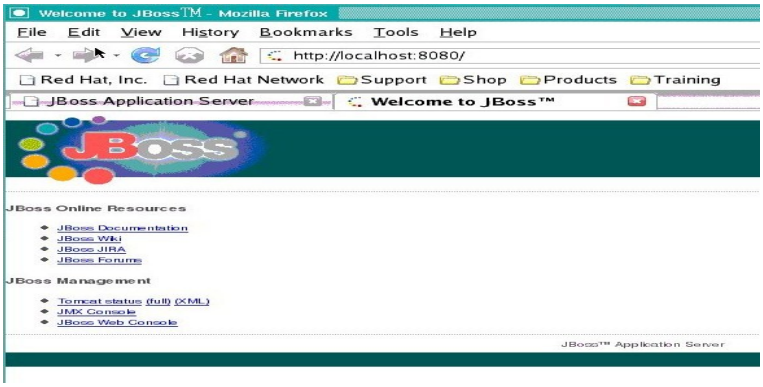

FIGURE 4.8 – interface de l'outil JBoss jbpm

L'exemple 4.5 montre une utilisation du système jBPM pour la gestion des ressources humaines.

Exemple 4.5 *Exemple d'utilisation de JBoss jBPM*

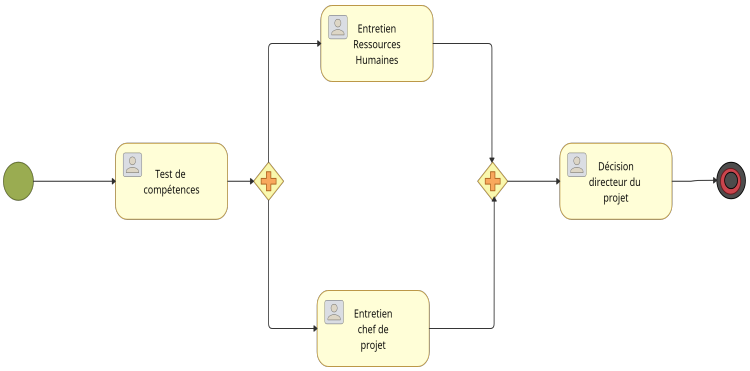

FIGURE 4.9 – Exemple d'un processus métier en utilisant JBoss jbpm

Les bases de données manipulées

Après avoir passé en revue les systèmes de gestion des PM les plus connus dans le domaine industriel, nous estimons qu'il est impératif de se pencher sur la question de la gestion des données gérées par ces systèmes. Dans ce qui suit un résumé, sous forme de tableau est exposé. Il permet de mettre en relief comment les SGBD sont exploités afin de prendre en charge les données des PM.

BDD \ Outils	**Oracle**	**MySQL**	**server SQL**	**Postgre SQL**	**IBM DB2**	**Informix**
1. Oracle BPM		×		×		
2. IBM	×		×		×	
3. Bonita Soft	×	×	×	×	×	×
4. SAP NetWeaver BPM		×			×	
5. JBoss jBPM	×	×		×	×	×

TABLE 4.2 – Tableau des bases de données manipulées

Conclusion

La gestion des données des processus métiers est un problème crucial pour lequel beaucoup de travaux de recherche ont été réalisés. Ces données sont caractérisées par :

— La dynamicité.

— Leurs schémas sont flexibles.

— Elles sont fortement connectées.

Néanmoins, la plupart des travaux de recherche existants se focalise essentiellement sur la modélisation des flux d'activités et ne donnent pas beaucoup d'importance à la gestion des données. D'un autre côté, dans le domaine industriel la panoplie des outils BPM disponibles sur le marché est capable de répondre, dans une large mesure, aux diverses préoccupations opérationnelles des utilisateurs. En outre, ces outils proposent des interfaces graphiques (e.g BPMN) pour la modélisation des processus métiers et pour le suivi de l'exécution des instances. Du point de vue gestion des données, les outils existants stockent les données persistantes afférentes aux processus métiers dans des bases de données de type relationnel, et l'utilisateur pourra choisir le type de SGBD qu'il va utiliser. En Effet, les données des processus métiers sont gérées par des SGBDR, avec toutes les difficultés induites.

L'analyse de l'état de l'art fait ressortir un besoin impératif pour une nouvelle perception des données des PM à la lumière des avancées technologiques et une nouvelle vision de cet aspect s'impose de manière forte. Cette vision doit prendre en compte les acquis réalisés par les SGBDG et leurs performances.

Nous estimons que l'obsolescence des infrastructures technologiques actuelles doit laisser la place à des solutions nouvelles qui dépassent les limites du modèle relationnel. Pour atteindre cet objectif et surmonter les insuffisances constatées, nous proposons dans le prochain chapitre une nouvelle approche, axée sur les avancées réalisées par les bases de données orientées graphes, pour la gestion des données des processus métiers.

Deuxième partie

Contribution et Implémentation de l'approche

Chapitre 5

Conception de l'approche

Introduction

Dans ce chapitre nous nous intéressons à la modélisation de notre approche pour la prise en charge des données des processus métiers. Nous y introduisons les modèles formels utilisés pour la représentations des processus métiers en tant que AFD, puis nous proposons un enrichissement de ce modèle par l'intégration des données associées et enfin nous illustrons l'exploitation des BDDG pour la gestion des données des processus métiers.

5.1 Modèle de représentation des PM

L'approche de résolution du problème de la gestion des données doit passer, inévitablement, par un choix du modèle qui sera utilisé pour représenter les PM. Du fait que nous visons à exploiter les BDDG, le modèle à choisir doit être le plus proche possible de celui manipulé par les BDDG. En effet, nous estimons qu'un modèle basé sur le principe des systèmes de transitions étiquetés (STE) ou **LTS** [1] offre une représentation qui est très adéquate avec le modèle des graphes sur lesquels sont basés les BDDG.

D'une manière formelle, un système de transition étiqueté est un triplet (S,Λ ,$\rightarrow$), avec :

- S est l'ensemble des états,
- Λ un ensemble d'étiquettes,
- $\rightarrow \subseteq S \times \Lambda \times$ S la relation de transition. S'il existe une transition étiquetée par $\lambda \in \Lambda$ entre deux états p et q, on écrit alors p $\xrightarrow{\lambda}$ q.

Il est constaté que différents modèles mathématiques abstraits peuvent être utilisés pour représenter un STE. On distingue essentiellement les trois modèles les plus utilisé.

— Les automates d'états finis déterministes (AFD).

— Les réseaux de Pétri.

— Les modèles de graphes.

Dans notre approche nous avons choisi d'utiliser les AFD pour représenter les PM. Dans ce qui suit, nous donnons la spécification formelle associée à la représentation des PM par des AFD.

1. Labelled Transition Systems

5.1.1 Utilisation des AFD pour représenter les protocoles des PM

Les AFD ont été largement utilisés pour représenter les processus métiers. Un AFD est un modèle mathématique de calcul, très utilisé dans le domaine de modélisation informatique. Son formalisme se compose d'un ensemble d'états reliés entre eux par des transitions (*les fonctions de transitions*) qui sont désignés par des symboles. Cette machine abstraite à la capacité de vérifier si un mot, introduit en entrée, est accepté ou refusé par l'automate. Cette fonction de vérification est réalisée par la reconnaissance des caractères introduits du début jusqu'à la fin, en parcourant le mot en entrée lettre par lettre. On parle d'**acceptabilité** ou de **reconnaissance du mot**.

Plus formellement, un protocole d'un processus métier est formalisé comme suit.

Définition 5.1 ***Protocole métier d'un PM*** *[35, 6]*
Le protocole d'un processus métier est un tuple $\mathcal{P} = (Q, q_0, \mathcal{F}, \mathcal{M}, \mathcal{R})$*, tels que :*

- Q *est un ensemble fini d'états ;*
- $q_0 \in Q$ *est l'état initial du protocole ;*
- $\mathcal{F} \subseteq Q$ *est l'ensemble des états finaux du protocole (ou acceptant) ;*
- $\mathcal{M}$ *est un ensemble fini d'activités abstraites ;*
- $\mathcal{R} \subseteq Q \times Q \times \mathcal{M}$ *est une relation de transition. Chaque élément* $(q, q', m) \in \mathcal{R}$ *représente une transition d'un état source* q *vers un autre état cible* q'*, suite à l'exécution de l'activité* m*.*

Selon cette spécification, les états de l'automate correspondent aux différentes phases par lesquelles passe le processus et les transitions entre les états expriment les opérations à réaliser pour passer d'une phase à une autre, lors de la progression du processus.

5.1.2 Quelques exemples de PM modélisés par des AFD

Pour montrer la force d'expressivité de notre modèle de représentation des PM, et sa capacité à pouvoir prendre en charge la modélisation d'une large variété de PM, nous exposons dans cette section trois exemples de PM qui sont inspirés du monde réels et leur spécifications respectives sous forme d'AFD. Pour chaque exemple, nous illustrons l'AFD qui représente le protocole du processus métier, puis nous donnons une brève description du fonctionnement de processus.

Exemple 5.1 ***Gestion des Commandes clients***
La figure 5.1 nous montre un AFD modélisant le PM de la gestion des commandes client. Comme il observé dans la figure, cet AFD est composé d'un ensemble d'états commençant par l'état initial début*. Suite à l'exécution des différentes activités spécifiés par la description de l'AFD, une instance donnée du PM peut atteindre différents état jusqu'à atteindre l'un des états finaux* Commande Archivée *ou* Commande rejetée *.*

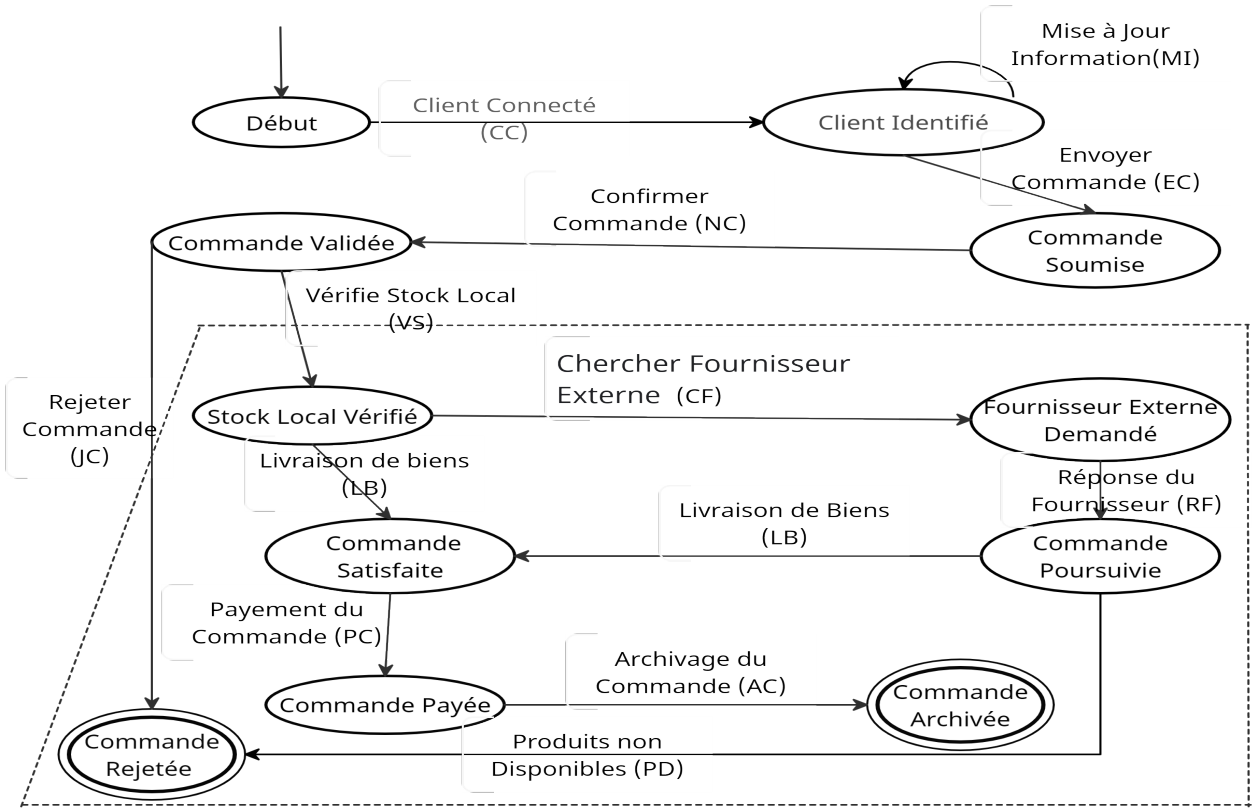

FIGURE 5.1 – AFD du PM de gestion des Commandes Client

Fonctionnement du Processus gestion des commandes Clients : Les spécifications décrites par l'automate de la figure 5.1 véhiculent le comportement suivant.

- Le processus commence à l'état initial q_0 (début), quand un client invoque le processus en lançant une nouvelle commande. A cette instant, une instance (*ou un cas*) du processus est générée.
- Suite à l'exécution de l'activité (Client Connecté)ou en abrégé (**CC**)), l'instance créée passe de l'état (début) vers l'état suivant Client Identifié.
- A partir de l'état Client Identifié, l'instance a deux possibilités. Soit exécuter l'opération Envoyer Commande (EC) et passer à l'état Commande soumise ou bien exécuter l'opération Mise à jour Informations (MI) et rester dans le même état Client Identifié.
- Conformément à ce mécanisme, l'exécution de chaque opération (*activité*) du PM, permet de réaliser une transition pour se déplacer vers le prochain état (**en utilisant l'état actuel et l'activité qui vient d'être exécutée**). Le processus de gestion des commandes client continue son exécution conformément à la logique métier décrite par l'automate modélisé.
- Une expression (**mot**) est reconnu par l'AFD, si et seulement si le dernier état (*i.e., l'état correspondant à l'exécution de la dernière activité*) est un état final de l'automate.

Exemple 5.2 ***Gestion des opérations électorales***
*La figure 5.2, nous montre un AFD pour la gestion d'une application e-gouvernement (**opérations électorales**). L'automate est composé par un ensemble d'états qui commencent par l'état initial* début*. Par la suite, une inscription d'un électeur est obligatoire pour qu'il puisse effectuer le vote. Aussi il peut consulter les profils des candidats ou visualiser les statistiques ou même abandonner le vote.*

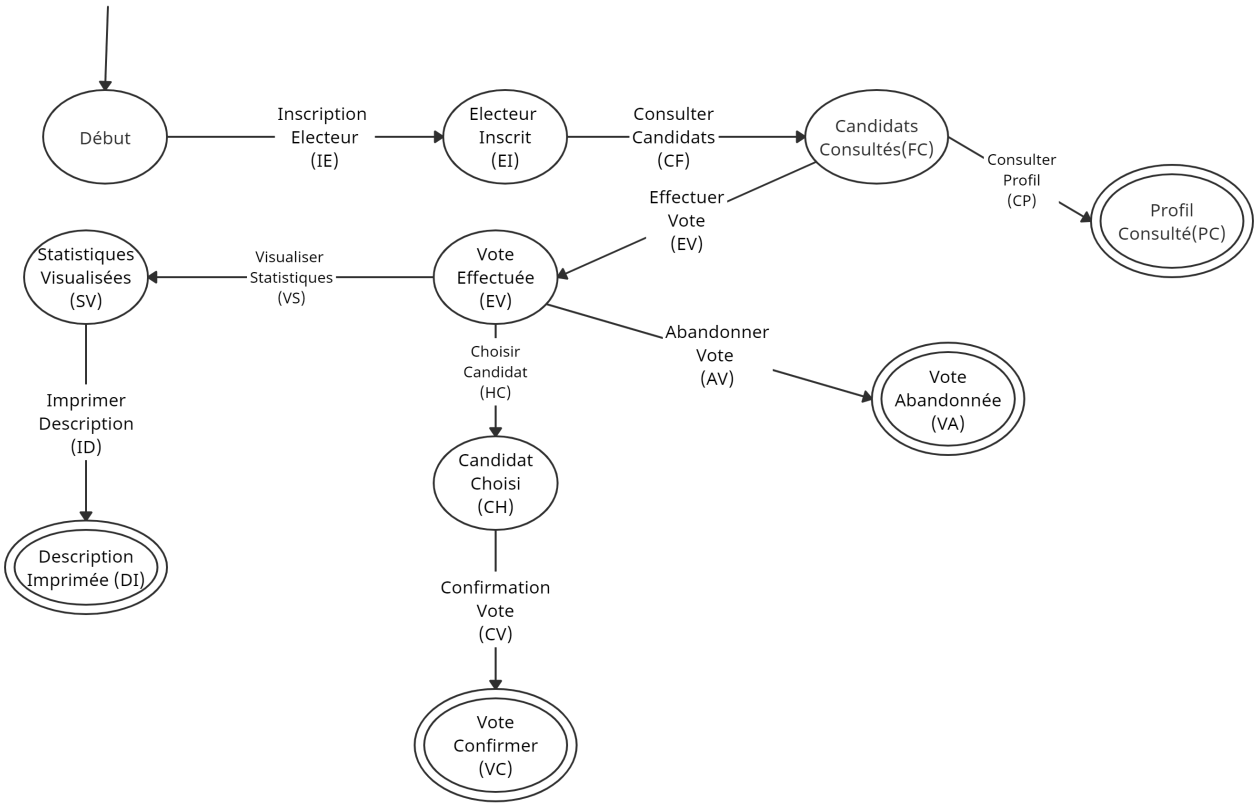

FIGURE 5.2 – Protocole Opérations électorales modélisé par un AFD

Fonctionnement du PM Gestion des opérations électorales : Considérons la figure 5.2

- Le processus commence à l'état initial q_0 (début), quand un électeur lance l'exécution du processus en invoquant la première opération offerte par la spécification du protocole, en l'occurrence l'opération Inscription électeur ou (IE). A cette instant, une instance (*ou un cas*) du processus est créée est ses données sont instanciées.
- Suite à l'exécution de l'activité (Inscription Électeur) ou en abrégé (**IE**)), l'instance générée passe de l'état (début) vers l'état successeur Électeur Inscrit.
- A partir de l'état Électeur Inscrit, l'instance peut invoquer l'opération Consulter Candidats et basculer par la suite vers l'état Candidats Consultés.

- Une fois au niveau de l'état Candidats Consultés, l'instance a deux possibilités. Soit exécuter l'opération Effectuer Vote (EV) et passer à l'état Vote Effectué et ensuite elle continue son exécution conformément au modèle offert. La deuxième possibilité consiste à exécuter l'opération Consulter Profil (CP). Dans ce dernier cas, l'instance passe à l'état *final* Profil Consulté.
- Conformément à ce mécanisme, l'exécution de chaque opération (*activité*) du PM, permet de réaliser une transition pour se déplacer vers le prochain état (*en utilisant l'état actuel et l'activité qui vient d'être exécutée*). Le processus métiers **Opérations Électorales** peut progresser en continuant son exécution conformément à la logique métier décrite par l'automate modélisé.

Exemple 5.3 ***PM Apprentissage à distance (e-learning)***
La figure 5.3 nous montre un AFD pour le suivi des formations à distance (e-learning). Le modèle de protocole du PM explicite l'inscription d'un apprenant pour accéder à une formation, choisir un cours et par la suite suivre son état d'avancement jusqu'à atteindre l'achèvement des formations choisies, tout en terminant l'exécution par des succès ou des échecs.

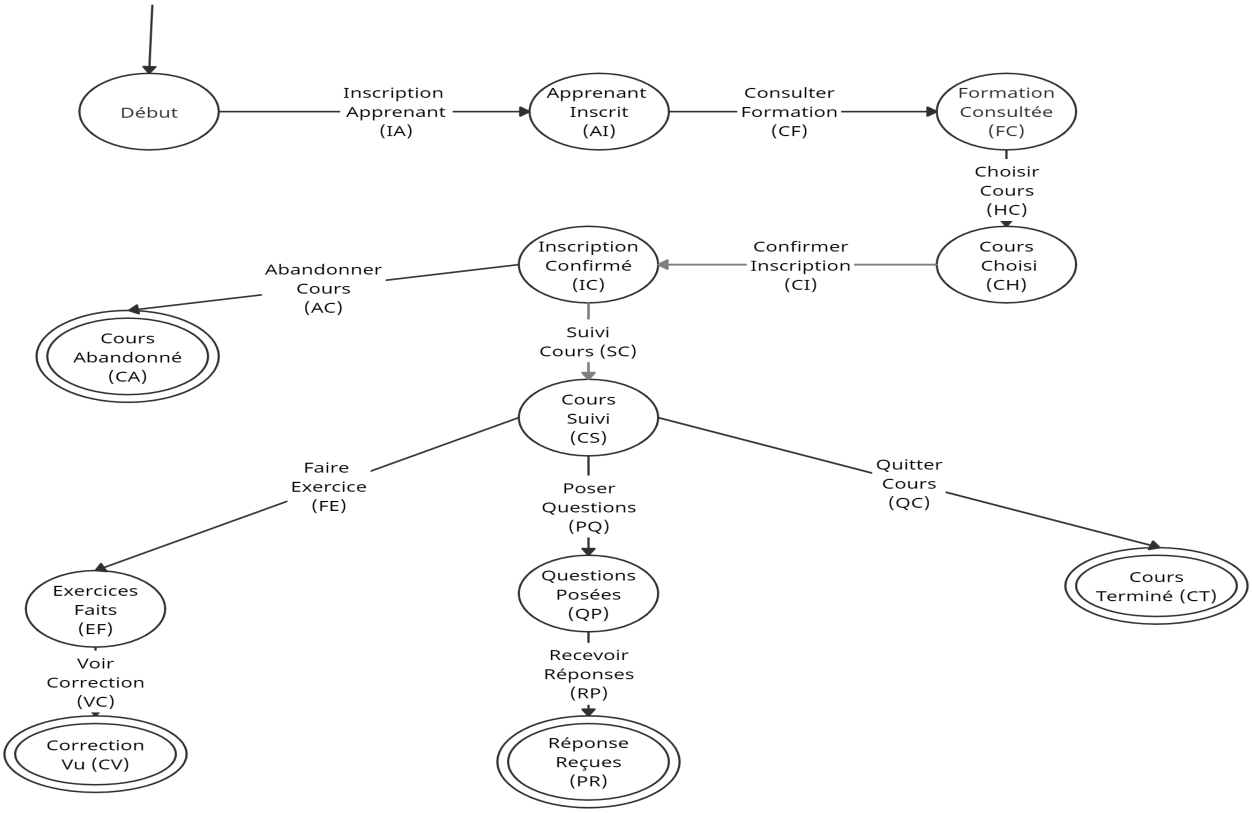

FIGURE 5.3 – AFD modélisant le PM e-learning

Fonctionnement du PM apprentissage à distance : Le comportement formalisé par l'AFD de la figure 5.3 est le suivant.

- Le processus commence à l'état initial q_0 (début), quand un apprenant invoque le processus d'apprentissage en ligne. A cette instant, une instance (*ou un cas*) du processus est générée.
- Suite à l'exécution de l'activité (Inscription Apprenant) ou en abrégé (**IA**), l'instance créée passe de l'état (début) vers l'état suivant Apprenant Inscrit.
- A partir de l'état Apprenant Inscrit, l'instance peut continuer son exécution en invoquant l'opération Consulter Formation (CF) et passer à l'état Formation Consultée. A partir de cet état, l'instance lance l'opération suivante Choisir Cours (HC) qui la fait basculervers l'état Cours Choisi, puis l'opération Confirmer Inscription(CI) qui conduit à son tour vers l'état Inscription Confirmée.
- A partir de l'état Inscription Confirmée, l'instance en cours a deux possibilités. Soit exécuter l'opération Abandonner Cours (AC) et passer à l'état final Cours Abandonné, ou bien lancer l'opération Suivi Cours(SC) et progresser suivant la logique décrite par le protocole jusqu'à atteindre l'un des trois états finaux possibles Correction Vu (CV), Réponses Reçues (PR) et Cours Terminée (CT).
- Conformément à ce mécanisme, l'exécution de chaque opération (*activité*) du PM permet de réaliser une transition pour se déplacer vers le prochain état (*en utilisant l'état actuel et l'activité qui vient d'être exécutée*).

5.1.3 Motivations pour le choix du modèle de PM

Différents modèles formels, dotés de niveaux d'expressivité variés, ont été proposés dans la littérature de recherche pour représenter les protocoles des PM. Néanmoins, dans notre approche nous avons choisi d'utiliser les AFD. Ce choix est justifié par les raisons suivantes.

— Les AFD ont été largement utilisés pour la modélisation des systèmes dynamiques et dans la formalisation de l'algèbre des processus.

— Ils sont dotés d'une assise théorique très solide ce qui facilite leur vérification.

— Existence d'une panoplie d'outils pour leur modélisation et leur vérification.

— Ils sont simples et très populaires.

Il est à noter que bien que nous utilisons les AFD pour représenter les protocoles des PM, les principes utilisés dans notre approche peuvent être étendus et exploités pour d'autres modèles, tels que les réseaux de pétri.

5.2 Concepts associés au modèle utilisé

Cette section est dédiée à l'exposé de l'ensemble des notions et concepts associés au modèle de PM choisi (*les AFD*) ainsi qu'à l'énoncé de certaines définitions. Ces ingrédients de base sont très importants pour la spécification de notre approche et pour l'illustration de nos développements futures.

Nous notons par $\Sigma = \{a, b, c, d, e, f, g, \ldots\}$ l'alphabet qui représente l'ensemble des symboles correspondant aux noms des *activités* fournies par un PM. Ces symboles sont associés aux opérations (*les activités abstraites*) et expriment les différentes transitions du processus d'un état vers un autre.
Par exemple, dans le PM Gestion des commandes de la figure 5.1, les activités sont exprimées par l'ensemble de symboles :
$\Sigma = \{CC, MI, EC, NC, VS, CF, RF, LB, JC, PC, AC, PD\}$.

5.2.1 Instances de processus

Un concept fondamental dans le domaine de la gestion des PM est la notion d'instance de PM que nous définissons ci-dessous.

Définition 5.2 *Une instance d'exécution d'un processus métier représente un cas réel d'un processus opérationnel d'une compagnie. Elle consiste en des exécutions structurées d'activités pour le processus en question [3].*

Il en résulte que chaque modèle du processus métier agit comme un plan ou moule pour l'ensemble des instances du processus métier et que chaque activité agit comme un plan pour l'ensemble des instances des activités.

Lors de l'invocation d'un PM par les clients, les instances associées empruntent divers chemins d'exécution en invoquant les différentes activités. Au fil du temps, l'historique d'exécution se constitue. En effet, chaque exécution d'une instance particulière génère une trace d'exécution.

A noter que différentes instances de processus peuvent être exprimées par la même trace ; *i.e. ; elle empruntent le même chemin d'exécution.* Cependant, chaque instance est caractérisée par ses propres attributs descriptifs qui lui sont spécifiques, par exemple : son identifiant, l'utilisateur qui l'a invoquée, les données manipulées lors de l'exécution des activités et le temps de démarrage, ...

Nous donnons, ci-dessous la notion de chemin d'exécution très utile pour spécifier le concept de trace d'exécution.

5.2.2 Exécution et chemin d'exécution

Soit $\mathcal{P} = (Q, q_0, \mathcal{F}, \mathcal{M}, \mathcal{R})$ le protocole d'un PM et soit $\mathcal{I}$ une instance du protocole $\mathcal{P}$. Nous introduisons, à présent, le concept d'exécution d'un protocole.

Définition 5.3 ***Exécution d'un PM***
Étant donné un protocole $\mathcal{P} = (Q, q_0, \mathcal{F}, \mathcal{M}, \mathcal{R})$, une ***exécution*** *de $\mathcal{P}$, notée* e*, est une séquence alternée d'états et d'activités de $\mathcal{P}$, telle que :*
e$=q_0.m_0.q_1 \ldots q_n.m_n.q_{n+1}$*, qui vérifie les trois conditions suivantes : (i) débute par l'état initial q_0 de P, (ii) se termine par un état q_{n+1} de $\mathcal{P}$, et (iii) respecte la relation de transition de $\mathcal{P}$, i.e., $(q_i, q_{i+1}, m_i) \in \mathcal{R}, \forall i \in [0, n]$.*

Une exécution e est dite **complète**, si elle commence par un état initial (*i.e., q_0*) et se termine par un état final (*i.e., si $q_{n+1} \in \mathcal{F}$*).

A toute exécution e $= q_0.m_0.q_1.m_1 \ldots q_l.m_l \ldots q_n.m_n.q_{n+1}$ de $\mathcal{P}$ est associée un **chemin d'exécution** de $\mathcal{P}$, noté c, tel que c $= m_0.m_1 \ldots m_l \ldots m_n$, constitué seulement

de la séquence finie d'activités obtenue suite à la suppression des noms des états de l'exécution e. Un chemin d'exécution est dit **complet** s'il est issu d'une exécution complète.

Le concept de chemin d'exécution est très utile pour distinguer les types d'instances de processus. En effet, les chemins d'exécution complets correspondent à des instances qui sont déjà terminées, alors que les chemins non complets reflètent des instances qui sont encore en cours d'exécution, ou **instances actives**.

Exemple 5.4 ***Exécutions et chemins d'exécution d'un protocole***
A titre d'exemples, les trois expressions suivantes sont des exécutions du protocole Gestion des commandes de la figure 5.1.

- e_1 =*debut.CC.Client Identifié.EC.Commande soumise.NC.Commande Validée.JC. Commande rejetée,*
- e_2 =*debut.CC.Client Identifié.EC.Commande soumise.NC.Commande Validée.VS. Stock Local Vérifié, et*
- e_3 =*debut.CC.Client Identifié.EC.Commande soumise.NC.Commande Validée.VS. Stock Local Vérifié.CF.Fournisseur Externe Demandé.*

On observe que e_1 *est la seule exécution complète, car le chemin d'exécution* $c_1 = CC.EC.NC.JC$ *associé à* e_1 *est complet et permet d'atteindre l'état final Commande rejetée* $\in \mathcal{F}$.

5.2.3 Trace d'exécution d'une instance de PM

Soit $\mathcal{P} = (Q, q_0, \mathcal{F}, \mathcal{M}, \mathcal{R})$ le protocole d'un PM, $\mathcal{I}$ une instance de $\mathcal{P}$, et soit $e = q_0.m_0.q_1 \ldots q_n.m_n.q_{n+1}$ l'exécution de $\mathcal{I}$ dans $\mathcal{P}$.

Définition 5.4 ***Trace d'exécution d'une instance***
La ***trace d'exécution*** *ou* ***historique d'exécution*** δ *d'une instance* $\mathcal{I}$ *du protocole* $\mathcal{P}$ *est exprimée par son chemin d'exécution* $c = m_0.m_1 \ldots m_n$, *extrait de l'exécution* e. *Elle représente la séquence finie des activités historiques qui ont été exécutées par l'instance* $\mathcal{I}$, *depuis le début de l'invocation du service jusqu'à l'état courant de* $\mathcal{I}$. *Une trace d'exécution est dite* ***complète*** *si elle est issue d'un chemin d'exécution complet.*

A titre d'illustration, nous donnons ci-après trois traces d'exécution δ_1, δ_2 et δ_3 associées à des instances d'exécutions différentes du protocole Gestion des commandes de la figure 5.1. Ces traces sont issues, respectivement, des chemins d'exécutions e_1, e_2 et e_3 de l'exemple 5.4.

δ_1 = CC.EC.NC.JC
δ_2 = CC.EC.NC.VS
δ_3 = CC.EC.NC.VS.CF

Parmi ces traces, seule δ_1 est une trace d'exécution complète, car son chemin d'exécution e_1 est complet, alors que δ_2 et δ_3 sont des traces incomplètes et représentent des instances en cours d'exécutions.

Formellement, l'ensemble des traces d'exécution complètes possibles d'un automate $\mathcal{P}$ est donné par le langage $L(\mathcal{P})$ reconnu par $\mathcal{P}$ (*i.e., l'ensemble des mots acceptés par* $\mathcal{P}$).

5.3 Enrichissement du modèle de PM par les données

5.3.1 Objectifs

Le modèle de représentation des PM proposé dans la section précédente est simple et basique. En effet, il ne prend pas en charge toute la sémantique véhiculée lors de la spécification et la manipulation des règles de gestion. Ainsi, il ne traite que les états et les transition sans prise en compte ni des contraintes temporelles ni des contraintes transactionnelles. Pire encore, il ne gère aucune donnée relative à l'exécution et à l'affectation des ressources. Les seules données manipulées par le modèle basique sont celles correspondantes aux schémas décrivant le PM.

Pour surmonter cette insuffisance, nous proposons dans cette section une extension du modèle de base par la prise en considération des données d'exécution afférentes aux processus métiers. Dans cette perspective, nous décrivons un enrichissement qui tient compte de l'axe des données et cela en ajoutant un ensemble de données (*attributs*) propres soit aux états et/ou aux transitions (*messages*).

5.3.2 Formalisation de l'enrichissement

D'après la définition 5.1 mentionnée dans la sous section précédente, nous avons défini un modèle d'AFD comme suit :

$$\mathcal{P} = (Q, q_0, \mathcal{F}, \mathcal{M}, \mathcal{R}), \tag{1}$$

— Q *est un ensemble fini d'états ;*

— $q_0 \in Q$ *est l'état initial du protocole ;*

— $\mathcal{F} \subseteq Q$ *est l'ensemble des états finaux du protocole (ou acceptant) ;*

— $\mathcal{M}$ *est un ensemble fini d'activités abstraites (messages ou transitions) ;*

— $\mathcal{R} \subseteq Q \times Q \times \mathcal{M}$ *est une relation de transition. Chaque élément* $(q, q', m) \in \mathcal{R}$ *représente une transition d'un état source* q *vers un autre état cible* q'*, suite à l'exécution de l'activité* m*.*

L'enrichissement que nous proposons est articulé autour de l'insertion de nouveaux paramètres additionnels qui permettront de décrire et de spécifier les données d'exécutions des PM. Ces paramètres sont : D_s et D_m, associées respectivement aux données des états et celles des messages. On obtient ainsi le nouveau modèle d'AFD décrivant les PM. Plus formellement, on aura la spécification suivante :
$\mathcal{BP} = (D_s, q_0, F, D_m, R)$, où :

- D_s : représente les données associées aux états Q, D_s Ce paramètre explicite 2 sous ensembles de données.
  - A_s : Attributs spécifiant la description des états du BP.i.e ; les attributs qui décrivent un état. Exemple : si on prend l'exemple 5.3 de e-learning les données de l'état **Formation Consultée** sont les suivants : l'identificateur de l'état *(ID)*, le type de l'état (initial, intermédiaire,final) *(Type)*,ressources utilisée pour chaque état (imprémente,...) *(Ressource)*, le temps maximum pour exécuter chaque état*(TempsMax)*, la condition ou bien la contrainte pour qu'on puisse exécuter cet état *(Condition d'atteinte)*.

- R_s : Les réalisations ou les occurrences des données des états du BP. Chaque instance R_s exprime une affectation d'une valeur à chaque attribut A_s.

  Si on prend par exemple , l'état précédent **Formation Consultée** avec ses attributs (exemple 5.3), on peut avoir à un instant donné l'instantiation suivante :

  Formation Consultée(145, intermédiaire, "Null", 10mn, la connexion). Il est à noter que les attributs associées aux états sont automatiquement les mémes attributs pour les activités.

- D_m : Données associées aux messages M (Transition ou activité).

  - A_m : Attributs spécifiant la description des messages (activités) du BP. Par exemple si on prend l'exemple 5.3 de e-learning les données de l'activité **ConsulterFormation** sont les suivants : l'identificateur de la formation *(IdFormation)*, le titre de la formation*(TitreFormation)*, la description de la formation *(DescriptionFormation)*, la date de la formation *(DateFormation)* , l'heure de la formation *(HeureFormation)*, la durée de la formation *(DuréeFormation)*,l'identificateur de l'instance de l'activité **(IdInst)**, **(TimeStamp)**, la situation de l'activité,elle peut étre : EnCours, EnAttente, Terminée **(Situation)**, nom de l'activité en cours d'exécution(ou son abréviation) dans le cas où cette activité est EnCours ou EnAttente sinon sa valeur est *"NULL"***(ActEnCours)**.
  - R_m : Les réalisations ou les occurrences des données des messages (activités) du BP. Chaque instance R_m exprime une affectation d'une valeur à chaque attribut A_m . Si on prend par exemple , l'activité précédente **ConsulterFormation** avec ses attributs (exemple 5.3), on peut avoir à un instant donné l'instantiation suivante :

    ConsulterFormation(126, SIQ, Systèmes informatiques,08/08/2021, 8h00, 36h, 1245, 25/08/2021,Terminée, Null).

    Il est à remarquer que l'ensemble de ces attributs : (IdInst, TimeStamp, Situation, ActEnCours) sont fixés pour toutes les activités avec des instances différentes.

Pour bien clarifier l'utilité des données des PM, en prenant toujours l'exemple d'e-learning (5.3), le tableau suivant nous montre l'ensemble des activités qui compose ce protocole avec ses données et la réalisation (l'instantiation) de ces dernières ;

5.3.3 Illustration de l'enrichissement des BP

le tableau suivant montre un exemple complet de la prise en compte des des données des activités du PM e-learning. Cet enrichissement est basé sur le modèle étendu.

Nom activité	[2] Abr	Données propres aux activités	Instantiation des	Attributs communs pour les activités
Inscription Apprenant	IA	IdApprenant NomApprenant PrénomApprenant Adresse Mail Niveau	16 "Bouziane" "Bochra" "Lotissement AinDefla" "bouzianebochra1@ gmail.com" "Master2"	IdInst : 5478, TimeStamp : "2021/05/10""04h00" , Situation : T , ActEnCours :NULL
Consulter Formation	CF	IdFormation TitreFormation [3] DescFormation DateFormation HeureFormation DuréeFormation	126 "SIQ" "Système Informatique" "2021/06/01" "08h00" "80heures"	IdInst : 4862, TimeStamp : "2021/05/12""02h12" , Situation : A ActEnCours :SC
Choisir Cours	HC	[4]AbrCours CodeCours	"BDDA" 1243	IdInst :3654 , TimeStamp : "2021/03/15""15h02", Situation :T ActEnCours : NULL
Confirmer Inscription	CI	Pseudo Password	"Bochra.Bouziane" "bochrabouziane"	IdInst :1563 , TimeStamp : "2021/04/13""16h00" , Situation :A ActEnCours :PQ
Abandonner Cours	AC			
Suivi Cours	SC	Pseudo Password CodeCours	"Bochra.Bouziane" "bochrabouziane" 1243	IdInst : 9241, TimeStamp : "2021/06/25""18h15", Situation : T ActEnCours :NULL
Envoyer Forum	EF	Pseudo Password	"Bochra.Bouziane" "bochrabouziane"	IdInst :6254 , TimeStamp : "2021/04/16""13h25" , Situation : C ActEnCours : FE

5.4 Jonction entre le modèle enrichi et les BDD graphes

A présent, nous disposons de 02 types de modèles distincts.
-Le premier est le nouveau modèle des PM enrichi par les données.
-Le deuxième est celui des BDD graphes.
Au début, il est constaté que les concepts manipulés par chacun sont différents.
Néanmoins, par une fonction de mapping qui permet de faire une correspondance

Poser Questions	PQ	Pseudo Password	"Bochra.Bouziane" "bochrabouziane"	IdInst : 4125 , TimeStamp : "2021/08/20""03h30" , Situation : A ActEnCours : VC
Faire Exercices	FE	Pseudo Password [5]DescExercice	"Bochra.Bouziane" "bochrabouziane" "Exercice N°10"	IdInst :4652 , TimeStamp :"2021/05/06""14h15" , Situation : C ActEnCours : FE
Voir Correction	VC	VoirCorrection	"OUI"	(IdInst : 4268, TimeStamp :"2021/08/15""13h00" , Situation :T ActEnCours :NULL)

TABLE 5.1 – Instantiation des données des activités du protocle de elearning

Concepts AFDe	Concepts BDDG
Etat	Nœud
Message	Lien
Etat initial	Nœud primaire
Etats finaux	Feuilles
Données états	Propriétés des noeuds
Données messages	Propriétés des association

TABLE 5.2 – Tableau de correspondances des concepts d'AFDe et des concepts des BDDG

de chaque concept du modèle du PM enrichi avec un autre concept qui lui soit adéquat dans le modèle cible qui est celui des BDDG

5.4.1 Spécification de la fonction de mapping

Plus formellement, soit C un concept, la fonction de mapping est la suivante : $C_{AFDe} \rightarrow C_{BDDG}$

Le tableau ci-dessous 5.2 résume les correspondances des concepts

5.4.2 Mise en œuvre du modèle enrichi par les données

Pour illustrer l'efficacité et la pertinence de notre approche pour la prise en compte des données des PM, nous exposons dans ce qui suit 3 exemples réels de PM enrichis par les données et ce conformément à la modélisation proposée.

Exemple 5.5 *Gestion des commande en ligne (e-commerce) pour le PM*
Prise en compte des données des messages (transitions) La figure montre bien la succession des activités.

Exemple 5.6 *Gestion de protocole de l'élection en ligne (e-gouvernement) pour le PM*
Prise en compte des données des états.

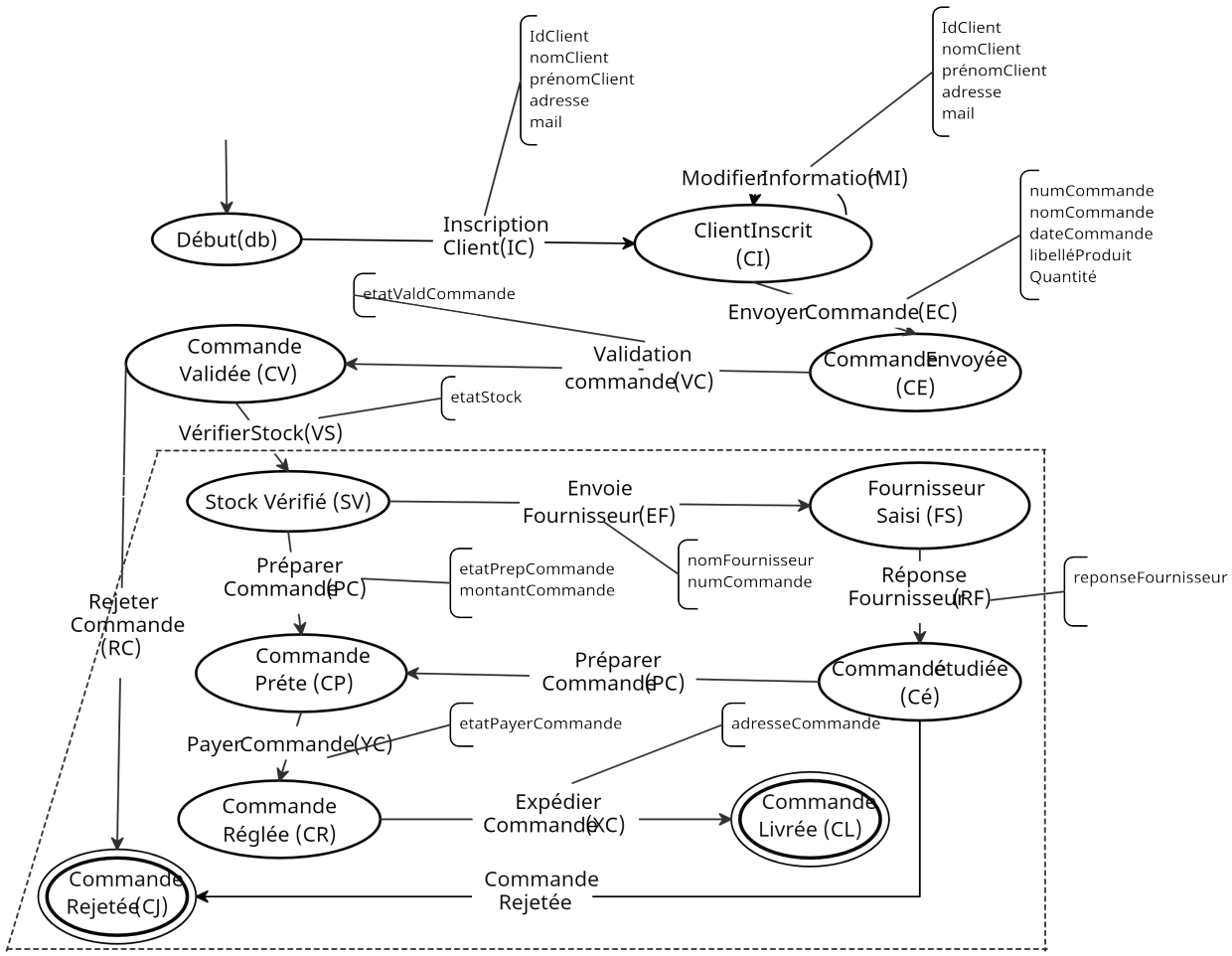

FIGURE 5.4 – réalisation d'un processus de commande par un AFD avec les données

Exemple 5.7 *Gestion de protocole de l'apprentissage en ligne (e-learning) pour le PM Prise en compte des données des états et des messages (transitions) , les deux à la fois.*

5.5 Conclusion

Dans ce chapitre nous avons exposé notre contribution et nous avons formalisé notre approche pour la gestion des données des PM.

Nous avons choisi la représentation des PMs par les AFD grâce à ses multiples avantages , puis nous l'avons enrichie par la prise en compte des données des PM.

Et enfin nous l'avons confronté avec le modèle utilisé *(AFD)* par les BDDG.

L'adéquation des deux modèles a été mise en évidence par l'application d'une fonction de correspondance des concepts.

Cette façon de faire ouvre la voie à la mise en œuvre de l'approche proposée.

C'est ce que nous allons voir dans le prochain chapitre.

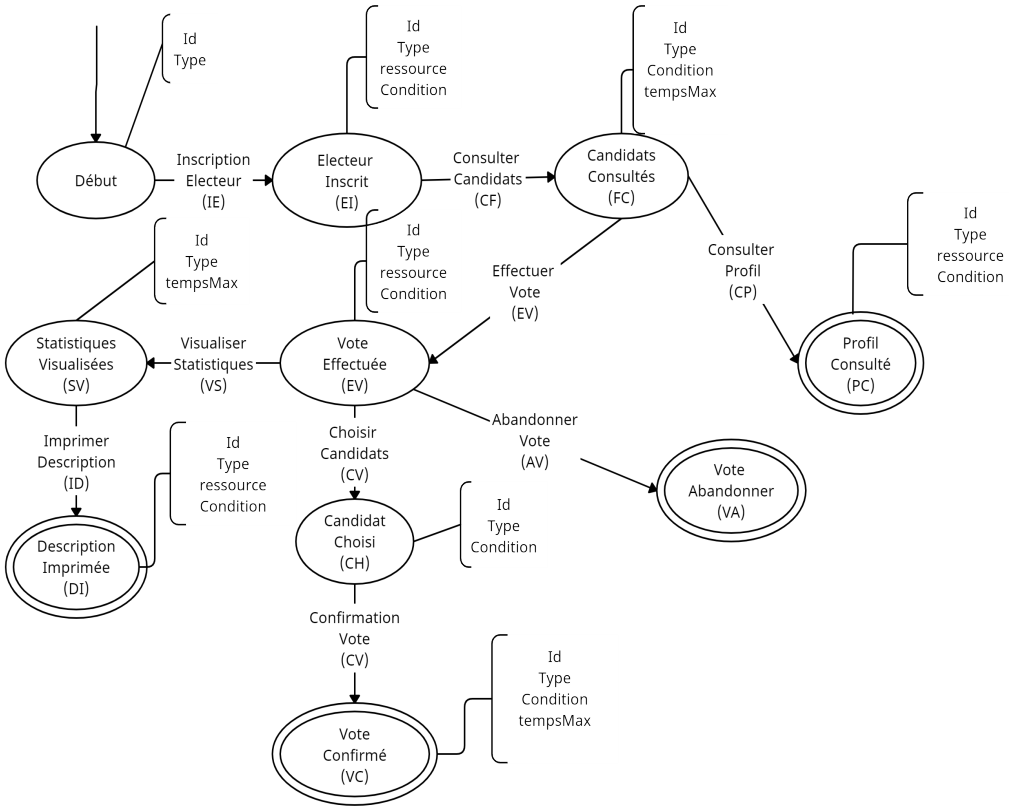

FIGURE 5.5 – Exemple de modèle comportemental d'un processus e-gouvernement par un AFD avec les données

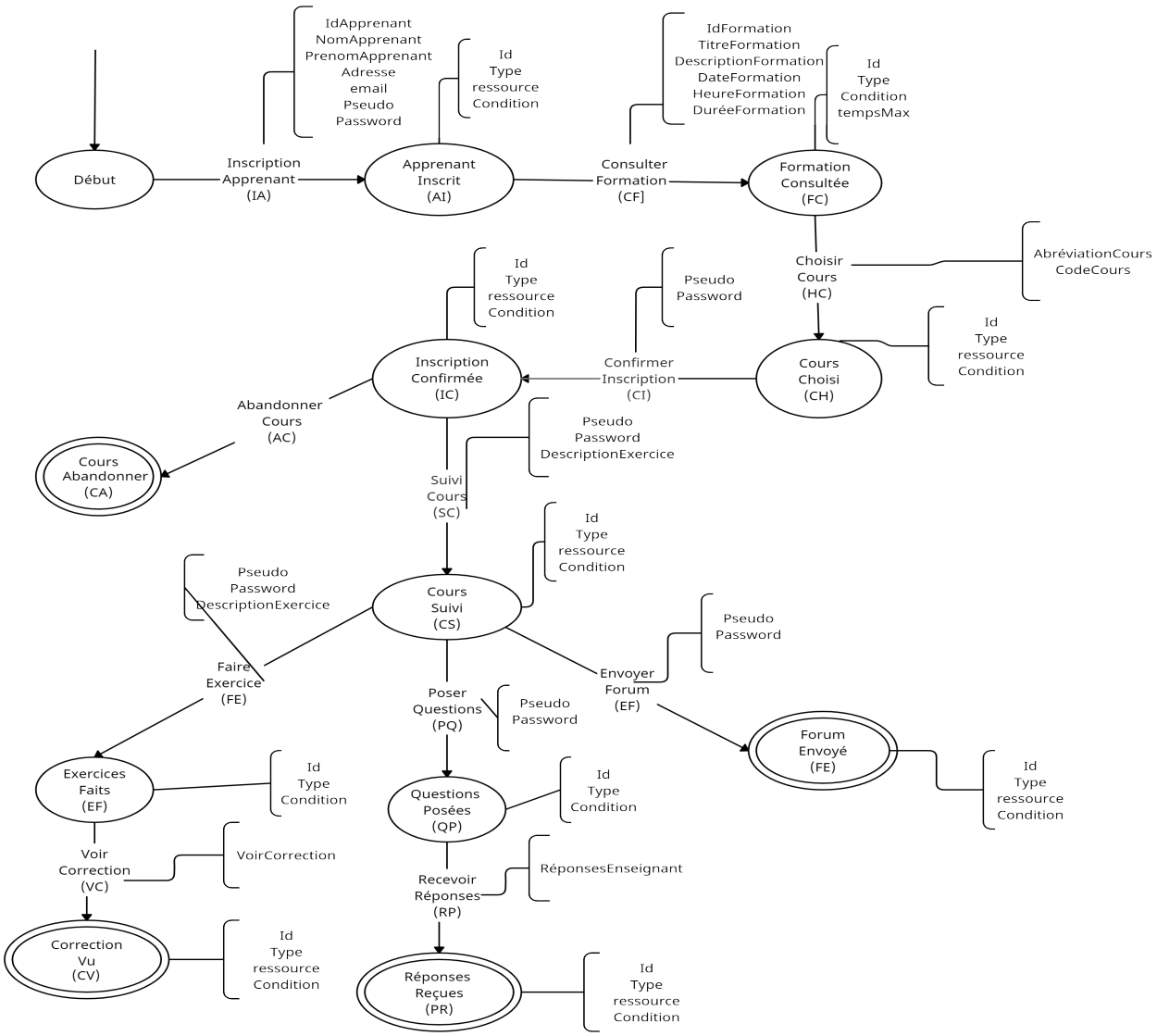

FIGURE 5.6 – Exemple de modèle comportemental d'un processus e-learning par un AFD avec les données

Chapitre 6

Implémentation et Expérimentation de l'approche

Introduction

Ce chapitre est dédié à l'implémentation de notre approche.Il a pour but de décrire la mise en œuvre pratique de la solution proposée.

Nous y introduisons une brève présentation de l'environnement de travail utilisés, puis nous exposons l'architecture générale du système réalisé ainsi que ses fonctionnalités et enfin un panorama d'exemples de PM concrétisés sous l'environnement de travail Neo4j est illustré.

6.1 Présentation de l'environnement de travail

Pour l'implémentation de notre approche nous avons utilisé un ensemble d'outils qui nous ont permis de réaliser notre objectif.Ces outils sont :

- Neo4j : un système de gestion de base de données orientée graphes.
- Cypher : le langage déclaratif pour l'interrogation de ces BDDG ;
- Eclipse : comme environnement de développement avec la version 4.20.0
- Java comme langage de programmation.

Dans ce qui suit ces outils seront présentés brièvement.

6.1.1 Les APIs Java

le langage de programmation java est très riche en bibliothèque open source parmi lesquelles nous avons utilisé windowbuilder [53], aussi nous avons utiliser d'autre API ; tels que JDBC Driver, API Java pour accéder aux données d'une base de données relationnelle et établir la connectivité entre cette base de données et le langage de programmation Java [54].

6.1.2 Neo4j

Neo4j est une base orientée graphes libre (sous licence GPLv3) appartient aux SGBD No-SQL, a été développé par Neo Technology (une société suédoise dont le siège est aux US) et initialement déployé en 2007. C'est la base de données orienté graphe la plus utilisée. C'est une base de donnée orienté graphe de haute performance, qui offre une structure orientée objet. Elle utilise les propriétés de modèle de graphe notamment les nœuds, arêtes avec leurs propriétés.

complètement compatible avec les transactions ACID, écrite en Java. Neo4j possède une architecture conçue pour la gestion, stockage et parcours de nœuds et leurs relations et il est spécialement conçu pour des données pouvant être modélisées sous forme de graphe, où les données sont stockées sur disque sous la forme d'une structure de données optimisée pour les réseaux de graphes. C'est une utilisation alternative du système de gestion de base de données classique. Elle permet de mettre l'accent sur les liens entre les données. Elle est hautement disponible et scalable.

Neo4j est utilisée pour les applications nécessitant de complexes relations comme les réseaux sociaux ou les moteurs de recommandations [55].

Neo4j possède un langage de requête puissant, Cypher, qui permet d'interroger le graphe pour obtenir toutes sortes d'informations sur les nœuds, leurs liens et le contenu de ces derniers[56].

Ses principales caractéristiques sont les suivantes : — transaction : c'est une base de données transactionnelle, respectueuse des principes ACID ; — Haute disponibilité : via la mise en place d'un cluster ; — Volume : stocker et requêter des milliards de nœuds et de relations ; — Cypher : un langage de requête graphe déclaratif, simple et efficace ; — schemaless : pas de schéma préétabli.

6.1.3 Cypher

Cypher est un langage de requête de base de données graphique expressif aussi le langage déclaratif permettant de requêter et mettre à jour le graphe, spécifique à neo4j.

Inspiré du SQL, on y retrouve beaucoup de concepts familiers, comme les clauses WHERE, ORDER BY, SKYP, LIMIT . . .

Cypher permet de naviguer dans les graphes en découvrant sa structure et les données. [57]

Comme la plupart des langages de requête, Cypher est un compositeur de clauses. Les requêtes les plus simples consistent en un ensemble de clauses MATCH suivi d'une clause RETURN.

RETURN Cette clause spécifie quels nœuds, relations et propriétés dans les données correspondantes doivent être renvoyés au client.

Autres clauses Cyper Les autres clauses que nous pouvons utiliser dans une requête Cypher incluent : **WHERE** fournit createria pour filtrer les résultats de correspondance de modèle.

CREATE and CREATE UNIQUE créer des nœuds et des relations.

MERGE Garantit que le modèle fourni existe dans le graphe, soit en réutilisant les nœuds et les relations existantes qui correspondent aux prédicats fournis, soit en créant de nouveaux nœuds et relations.

DELETE Supprimez les nœuds, les relations et les nœuds.

SET Définit les valeurs de propriété.

FOREACH Effectue une action de mise à jour pour chaque élément d'une liste.

UNION Fusionne les résultats de deux ou plusieurs requêtes.

WITH Enchaîne les parties de requête suivantes et transmet les résultats de l'une à l'autre. Commandes de tuyauterie similaires sous Unix.

STRAT Spécifie un ou plusieurs points de départ explicites (nœuds ou relations) dans le graphique[58].

Le navigateur Neo4j Les requêtes s'écrivent dans la partie haute. Les résultats se visualisent dans la partie basse.

6.2 Architecture générale et les fonctionnalités du système

Dans cette section nous allons aborder l'architecture générale de notre système qui est le stockage et l'interrogation des données des PM en exploitant les BDDG. Ainsi que les principales fonctionnalités que ce système nous offre.

Ainsi, comme il est constaté dans la figure 6.1 notre système utilisé s'articule sur deux partie principale :

- La gestion des schémas des PMs : cette partie commencera par la création des schéma des PMs, la modification (MAJ) des PMs et l'évolution des PMs, après cela interviendra la représentation des PM par les AFD, XML. Ces fichiers XML seront interrogés au moyen du langage XPATH, XQuery.

  A la fin de cette partie, sera la représentation des PM par des AFD étendus (graphe). en contre partie,
- La gestion des données des PMs : cette partie englobera plusieurs types de données des PMs (schémas, historiques,...) qui peuvent être stockées dans des BDR qui seront interrogées au moyen du langage SQL.

A la fin de ces étapes, on stockera toutes ces données des PMs dans des BDG à l'aide de l'outil Neo4j et on les interrogera au moyen du langage Cypher qui conclura le travail du notre système.

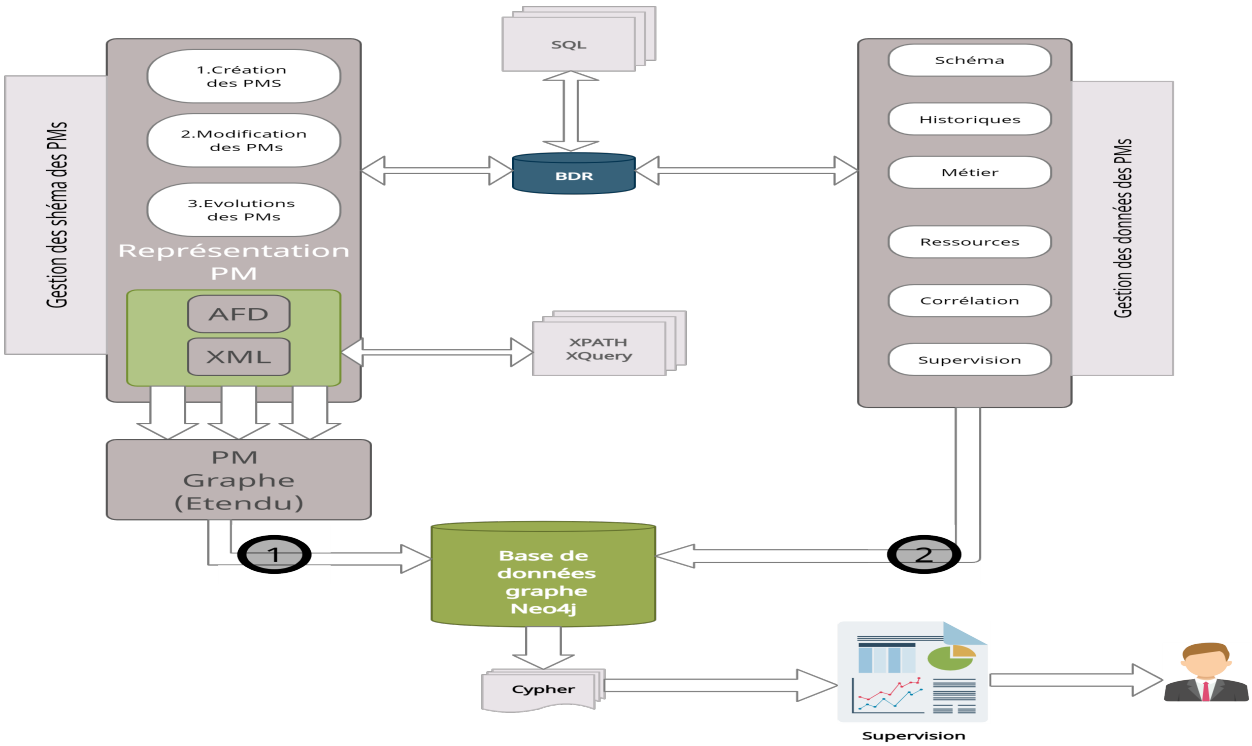

FIGURE 6.1 – Architecture générale de notre système

6.3 Déploiement de Neo4j pour la gestion des données des PM

Dans cette section, nous allons montrer l'utilisation de Neo4j qui nous permet de visualiser les graphes des PM et les interroger avec des requêtes. captures d'écran illustrant des exemples sur les graphes des PM et les requêtes sur les graphes des PM.

6.3.1 Graphe des PM

Dans cette partie, nous allons illustrer notre démarche au moyen de quelques exemples comme suit :

La figure 6.2 nous montre l'exemple de PM d'e-learning :

La figure 6.3 nous montre l'exemple de PM d'e-commerce :

La figure 6.4 nous montre l'exemple de PM d'e-election :

6.3.2 requêtes sur les graphes des PM

Dans cette partie, nous allons montrer quelques exemples de requêtes sur les graphes des processus métiers.

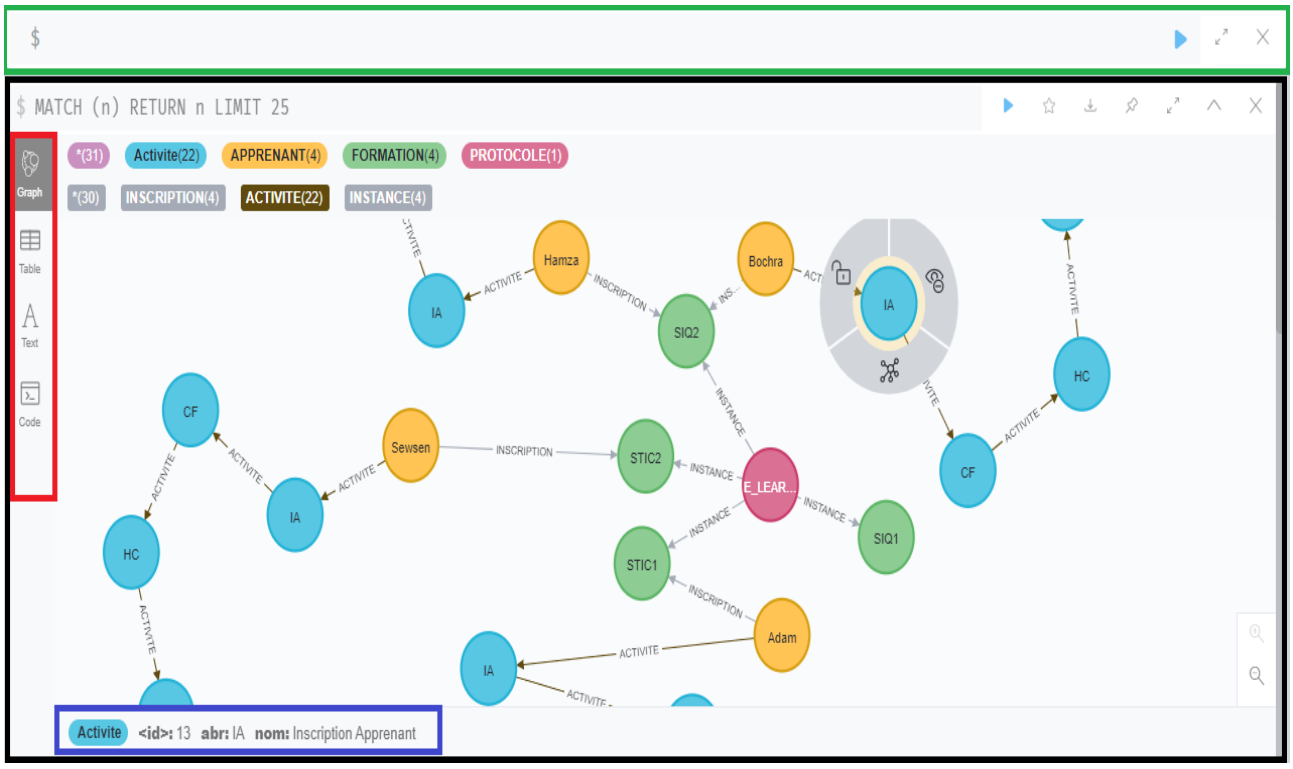

FIGURE 6.2 – Le navigateur Neo4j avec l'exemple du e-learning

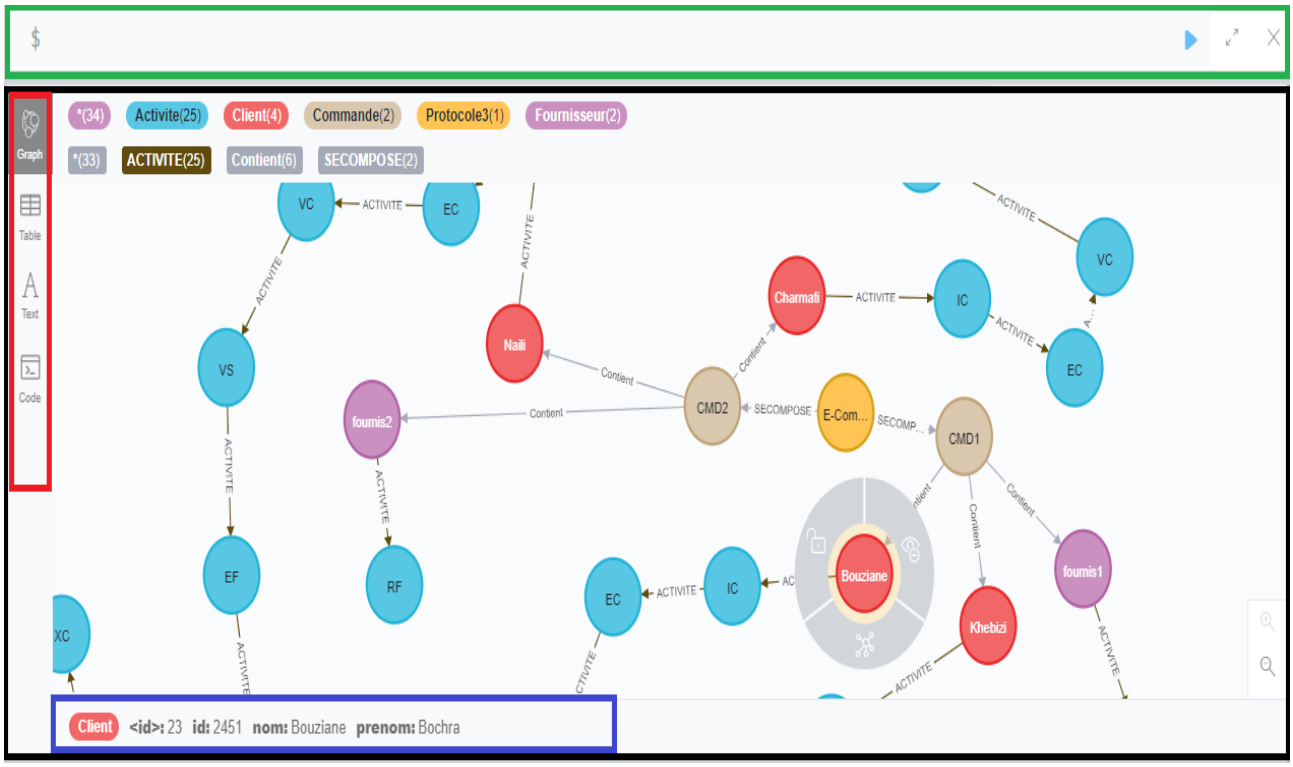

FIGURE 6.3 – Le navigateur Neo4j avec l'exemple du e-commerce

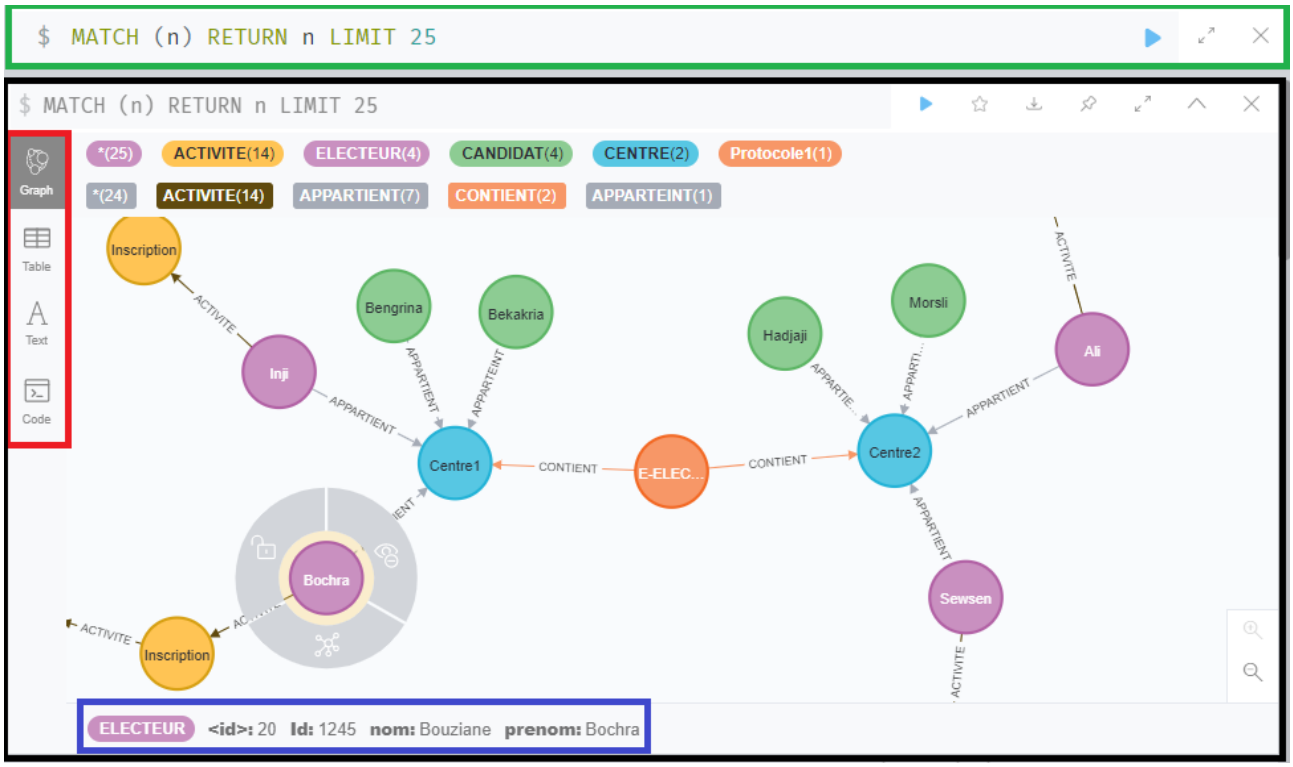

FIGURE 6.4 – Le navigateur Neo4j avec l'exemple du e-gouvernement

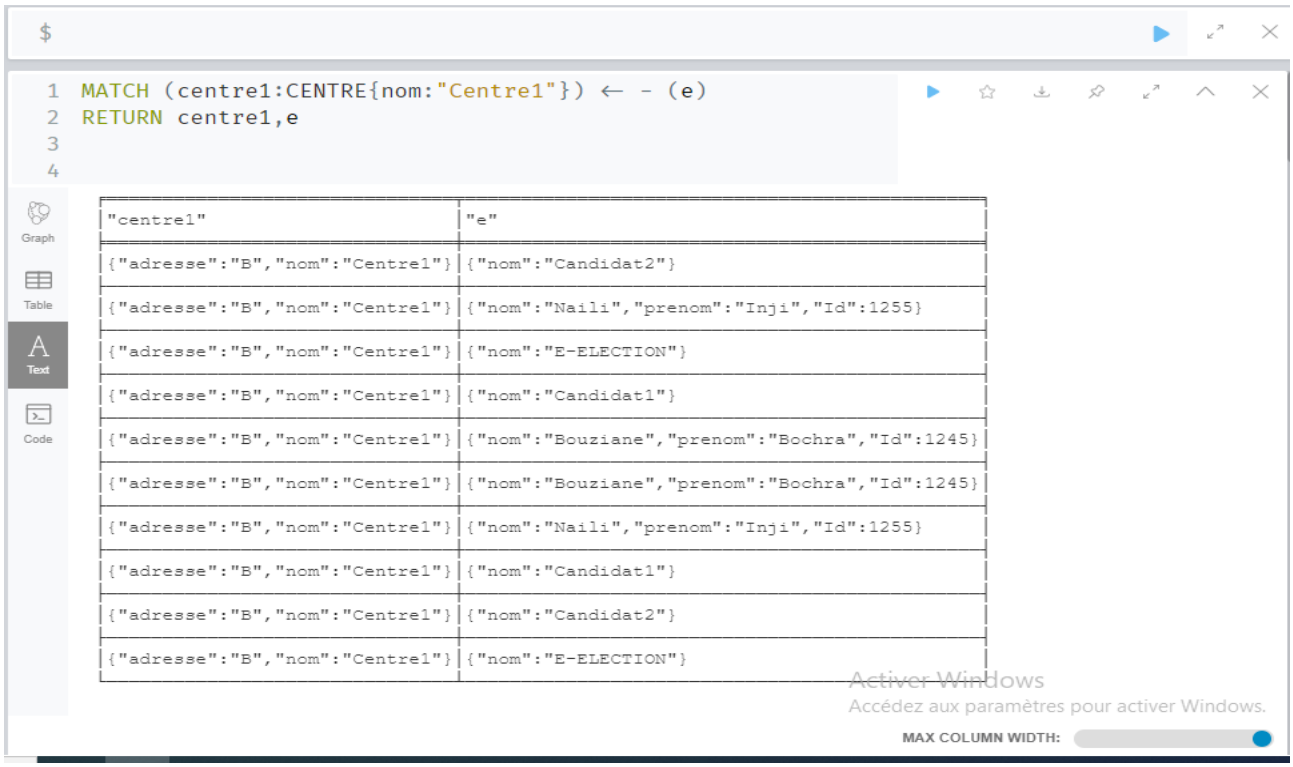

FIGURE 6.5 – exemple de la requête 1

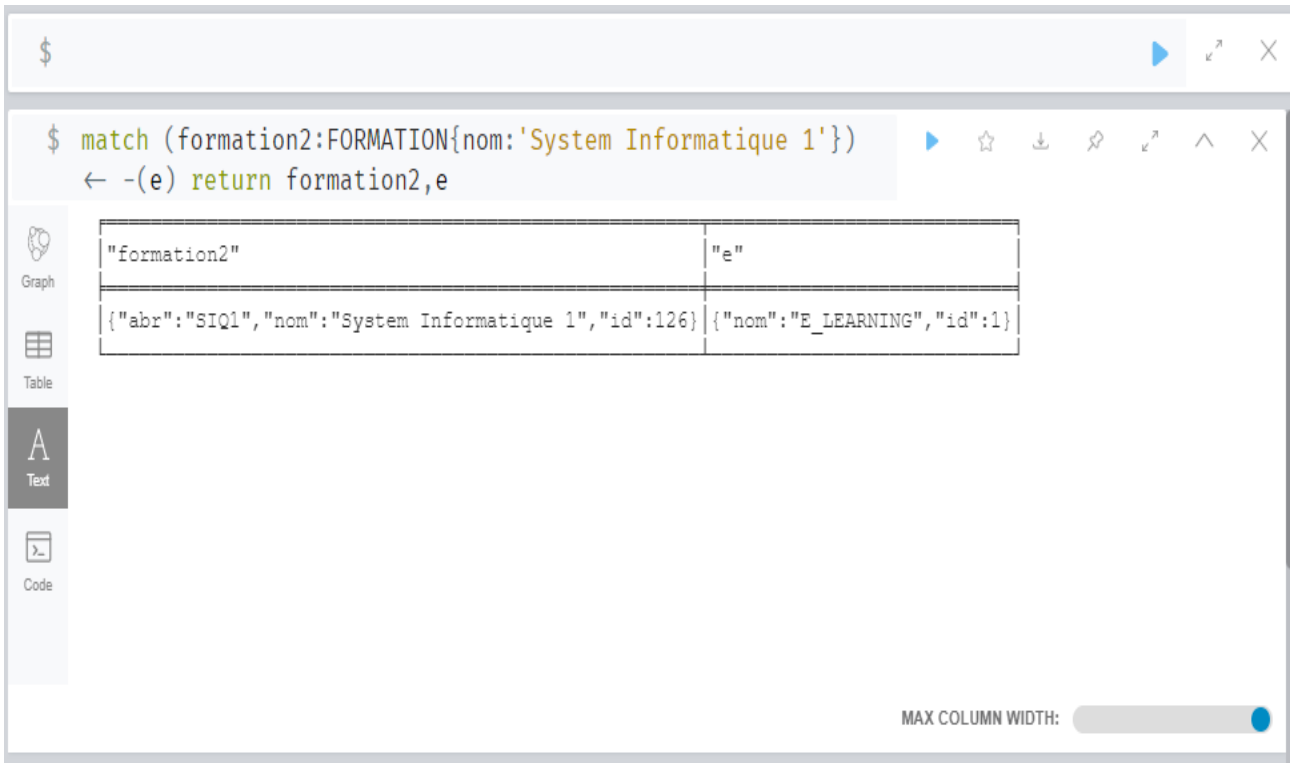

FIGURE 6.6 – exemple de la requête 2

FIGURE 6.7 – exemple de la requête 3

FIGURE 6.8 – exemple de la requête 4

FIGURE 6.9 – exemple de la requête 5

FIGURE 6.10 – exemple de la requête 6

6.4 conclusion

Notre dernier chapitre a comporté l'implémentation de l'application en conformité avec ce qui a été prévu au début ayant pour thème le stockage et l'interrogation des données des PM en exploitant les bases de données graphes.

Ainsi, il apparait que notre application a pris en compte les aspects essentiels entourant ce thème :

- Architecture générale de notre système ainsi que les fonctionnalités principales.
- Les captures d'écran sur la création des BDDG de quelques exemples de processus métiers (e-learning, e-commerce, e-election).
- Les captures d'écran sur les requêtes des BDDG représentants les exemples des PM utilisés.

  Cependant, malgré les résultats prévus qui ont été obtenus et compte tenu que notre modèle utilisé est une première version, il demeure clair que cette dernière(l'implémentation de notre approche) reste à améliorer pour aboutir à des résultats plus performants.

Chapitre 7

Conclusion Générale

Dans ce travail, nous nous sommes intéressés au domaine des PM et nous avons traité le problème de stockage et d'interrogation des données lors de la gestion des PM. L'objectif est de surmonter le déficit constaté dans le domaine en raisons de la prise en charge de l'aspect flux de message au détriment des données manipulées. Dans notre approche nous avons exploité les BDG.

Aussi, pour pallier à certaines insuffisances conceptuelles et fonctionnelles, dans notre approche nous avons suggéré la représentation des PM au moyen des AFD, méthode qui a montré son efficacité dans les représentations des PM et qui a apporté beaucoup davantages. En ce sens, un modèle de base pour la représentation des PM a été conçu et formalisé, puis enrichi par la prise en compte des données.

De même, pour le stockage et l'interrogation des PM, nous avons opté pour le paradigme émergent des BDG qui constituent de nos jours, une nouvelle tendance très prometteuse pour la gestion des données flexibles, inter-connectées massives. d'autres part, leurs performances sont meilleures que les bases de données relationnelles et objets.

sur le plan pratique, nous avons utilisé l'outil **Neo4j** pour l'implémentation de notre approche avec le langage déclaratif **Cypher** pour l'interrogation de données des PM concernés. Aussi, nous avons illustré notre étude par des exemples réels (*e-learning, e-gouvernement, e-commerce*).

Aux termes de ce travail, nous avons capitalisé des acquis suivants :

— Du point de vue théorique : Compréhension et utilisation des PM dans des domaines multiples et leur enrichissement par les données ainsi que la maitrise des outils permettant leur représentation (*Bonita, BPM...*).

— Du point de vue opérationnel : la capacité de répondre à des problèmes existants au niveau des entreprises et leur apporter les solutions appropriées au moyen de l'outil **Neo4j** et son langage déclaratif d'interrogation des données **Cypher** dans le cadre de l'approche PM et la connaissance de la gestion des diverses phases de cycle de vie des PM ainsi que le suivi de leurs instances d'exécution (*log files*).

— Du point de vue méthodologique : j'ai appris à comprendre et à aborder de manière scientifique une question de recherche liées aux PM, en commençant par cerner le problème, analyser l'état de l'art et enfin proposer une solution, la concevoir et l'expérimenter.

Perspectives Ainsi, il est constaté que l'introduction des données dans l'approche des PM a beaucoup amélioré l'efficacité de cette dernière. Mais la demande de plus en plus importante dans l'amélioration des résultats de l'entreprise ainsi que l'efficacité des rendements attendus dans beaucoup de domaine(e-learning,...) demandent et exigent des résultats encore plus intéressants et importants.

De même, la technologie de l'informatique et les réalisations de matériels informatiques de plus en plus performants laissent à prévoir que cette approche (l'utilisation des PM) connaitra sûrement de grands progrès réalisables au bénéfice de l'ensemble des concernés (chercheurs, entreprises, clients).

Pour ma part, j'espère et je souhaite que ce modeste travail réalisé permettra une participation dans la compréhension de l'utilisation de l'approche des PM et qu'il sera encore enrichi par d'éventuels futurs travaux d'exploration théorique et pratique qu'est l'apport de l'informatique au niveau des entités faisant appel à cette technologie (entreprises,...).

Bibliographie

[1] L'équipe Focus Performance. Focus performance. https://www.pyx4.com/blog/processus-qualite-definition-iso-9001/, February 2019.

[2] Tourmen Claire. Activité, tâche, poste, métier, profession : quelques pistes de clarification et de 19 :15 à 20, hs 2007.

[3] Mathias Weske. *Business Process Management*. Second Edition, 2012.

[4] What is bpms (business process management system) ? https://www.creatio.com/page/bpms, February 2020.

[5] Business process modeling techniques with examples. April 2021.

[6] Boualem Benatallah, Fabio Casati, and Farouk Toumani. Web service conversation modeling : A cornerstone for e-business automation. *IEEE Internet Computing*, 8(1) :46–54, 2004.

[7] Eric Simon and Kilian Stoffel. State machines and petri nets as a formal representation for systems life cycle management. In *Proceedings of IADIS International Conference Information Systems*, pages 275–272, 2009.

[8] sparxsystems. https://www.sparxsystems.fr/resources/uml2_tutorial/uml2_activitydiagram.html, February 2021.

[9] Qu'est-ce qu'un diagramme de séquence uml ? https ://www.lucidchart.com/pages/fr/diagramme-de-sequence-uml, February 2021.

[10] Muhammad Haadi. L'évolution de la base de données. 10 2010.

[11] Openclassrooms. Administrez vos bases de données avec mysql. https://openclassrooms.com/courses/administrez-vos-bases-de-donnees-avec-mysql/introduction-14.

[12] Bases de données. https://www.base-de-donnees.com/comprendre-bases-de-donnees/les-4-types-de-bases-de-donnees/, February 2021.

[13] Robert Polding. Bases de données : évolution et changement. 08 2018.

[14] Andreas Meier. *Introduction pratique aux bases de données relationnelles*. Springer Science & Business Media, 2005.

[15] Lemagit. 03 2021.

[16] Marc Berges. Object orientation in the literature and in education. *it - Information Technology*, 60(2) :69–77, 2018.

[17]

[18] Eric Van der Vlist. *XML schema*. O'Reilly Media, Inc., 2002.

[19]

[20] Georges Gardarin, Georges Gardarin, France Informaticien, Georges Gardarin, and Georges Gardarin. *XML : Des bases de données aux services Web*, volume 28. Dunod, 2002.

[21] https://fr.wikipedia.org/wiki/XPath#:~:text=XPath%20est%20le%20langage%20de%20requ%C3%AAte%20%C3%A9l%C3%A9mentaire%20dans%20XSLT.&text=XPath%20peut%20%C3%AAtre%20utilis%C3%A9%20comme,partiellement%20aussi%20dans%20XML%20Schema.

[22] https://fr.wikipedia.org/wiki/XQuery.

[23] https://aws.amazon.com/fr/nosql/document/.

[24] https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_orient%C3%A9e_documents.

[25] telechargercours.com. Théorie de graphe. https://telechargercours.com/theorie-de-graphe/.

[26] Michel Domenjoud. Bases de données graphes : un tour d'horizon. http://blog.octo.com/bases-de-donnees-graphes-un-tour-dhorizon/, 2012.

[27] Margaret Rouse. Base de données orientée graphes. http://www.lemagit.fr/definition/Base-de-donnees-orientee-graphes.

[28] https://www.lebigdata.fr/base-de-donnees.

[29] Andreas Meyer, Sergey Smirnov, and Mathias Weske. Data in business processes. *EMISA Forum*, 31 :5–31, 2011.

[30] Richard Hull, Jianwen Su, and Roman Vaculin. Data management perspectives on business process management : Tutorial overview. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 943–948, New York, NY, USA, 2013. ACM.

[31] Sherif Sakr and Ahmed Awad. A framework for querying graph-based business process models. pages 1297–1300, 01 2010.

[32] PIET Boekhoudt, Herk Jonkers, MICHIEL Rougoor, and N Mastorakis. Graph-based analysis of business process models. In *Mathematics and Computers in Modern Science, Proc. of the WSES/MIUE/HNA International Conference*, pages 227–235. Montego Bay, Jamaica, 2000.

[33] Henry M Franken and Wil Janssen. Get a grip on changing business processes. *Knowledge and process management*, 5(4) :208–215, 1998.

[34] Dick AC Quartel, Luís Ferreira Pires, Marten J Van Sinderen, Henry M Franken, and Chris A Vissers. On the role of basic design concepts in behaviour structuring. *Computer networks and ISDN systems*, 29(4) :413–436, 1997.

[35] A. Khebizi, H. Seridi-Bouchelaghem, B. Benatallah, and F. Toumani. A declarative language to support dynamic evolution of web service business protocols. *SOCA*, 11(2) :163–181, Jun 2017.

[36] Fabio Casati, Malu Castellanos, Umeshwar Dayal, and Norman Salazar. A generic solution for warehousing business process data. In *Proceedings of the 33rd international conference on Very large data bases*, pages 1128–1137. VLDB Endowment, 2007.

[37] Alexander Koller. Semantic construction with graph grammars. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 228–238, 2015.

[38] Asma Hassani and Sonia Ayachi Gahnouchi. A framework for business process data management based on big data approach. *Procedia Computer Science*, 121 :740–747, 2017.

[39] Nenad Jukic, Boris Jukic, and Mary Malliaris. Online analytical processing (olap) for decision support. In *Handbook on Decision Support Systems 1*, pages 259–276. Springer, 2008.

[40] Hajo A. Reijers Kees M. van Hee. Using formal analysis techniques in business process redesign. 1806 :15, 03 2002.

[41] Amin Jalali. Graph-based process mining. *arXiv preprint arXiv :2007.09352*, 2020.

[42] Wasim Sadiq and Maria E Orlowska. Analyzing process models using graph reduction techniques. *Information systems*, 25(2) :117–134, 2000.

[43] Ricardo et Piattini Mario Fernández-Ropero, María et Pérez-Castillo. Graph-based business process model refactoring.

[44] Siham Rim Boudaoud, Khaoula Es-Salhi, Vincent Ribaudy, and Ciprian Teodorov. Relational and graph queries over a transition system. In *IEEE EUROCON 2015-International Conference on Computer as a Tool (EUROCON)*, pages 1–6. IEEE, 2015.

[45] David Dominguez-Sal, Peter Urbón-Bayes, Aleix Giménez-Vanó, Sergio Gómez-Villamor, Norbert Martínez-Bazan, and Josep Lluís Larriba-Pey. Survey of graph database performance on the hpc scalable graph analysis benchmark. In *International Conference on Web-Age Information Management*, pages 37–48. Springer, 2010.

[46] N. Lohmann. Where did i go wrong ? explaining errors in business process models. *CEUR Workshop Proceedings*, 1140 :8–16, 01 2014.

[47] Model checking.

[48] Stefan Esser and Dirk Fahland. Storing and querying multi-dimensional process event logs using graph databases. In *International Conference on Business Process Management*, pages 632–644. Springer, 2019.

[49] Veronica Gacitua-Decar and Claus Pahl. *Towards Reuse of Business Processes Patterns to Design Services*, pages 15–36. 11 2009.

[50] Patrice Briol. *Bpms-L'automatisation des processus métiers*. Lulu. com, 2014.

[51] Ann Rosenberg, Greg Chase, Rukhshaan Omar, James Taylor, and Mark von Rosing. *Applying real-world BPM in an SAP environment*. Galileo Press Bonn, Bosten, 2011.

[52] N Pin, S Riku, and H Måns. Open source power on bpm-a comparison of jboss jbpm and intalio bpms. *T-86.5161 Special Course in Information Systems Integration*, 2007.

[53] developpez.com. Les bibliothèques de java. https://www.developpez.net/forums/d1133872/java/general-java/debuter-java/bibliotheques-java/.

[54] tutorialspoint. Jdbc tutorial. https://www.bing.com/search?q=jdbc&cvid=867f056043e0419ba3d3b2342c700d24&aqs=edge.0.69i59j0l5j69i60j69i61j69i60.1664j0j1&pglt=299&FORM=ANNTA1&PC=U531.

[55] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph databases : new opportunities for connected data.* " O'Reilly Media, Inc.", 2015.

[56] A Boulmakoul and Z Besri. Analyse structurale et ontologie des organisations pour la gouvernance des processus du système d'information. *INTIS'2014*, page 60, 2014.

[57] M Maxime Crochemore. *UFR LANGAGE, INFORMATIQUE, TECHNOLOGIE.* PhD thesis, UNIVERSITE PARIS 8, 1999.

[58] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph databases : new opportunities for connected data.* " O'Reilly Media, Inc.", 2015.

.