REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE

DEPARTEMENT INFORMATIQUE

Mémoire de fin d'études
En vue de l'obtention du Diplôme de Master en informatique
Option : Conduite de projet informatique

THEME :

INTEGRATION DES SYSEMES DANS UN ENVIRONNEMENT REPARTI

Présenté par :

M^lle^ AIT AMEUR Thiziri

M^lle^ ALLALI Sabrina

Proposé et dirigé par :

M^r^ KERBICHE

PROMOTION : 2016-2017

Remerciement

Je tiens en premier lieu à remercier notre promoteur Mr KERBICHE pour son encadrement, et son suivi durant toute la durée du projet.

Que les membres de jury trouvent ici nos plus vifs remerciements pour avoir accepté de nous faire l'honneur, en évoluant ce modeste travail.

Notre profonde gratitude et sincères remerciements vont à tous les professeurs qui nous ont suivis durant notre parcours d'étude.

Nous tenons également à remercier nos parents pour leur soutien affectif à toutes les étapes de notre cheminement.

Dédicaces

Avant tout je remercie le DIEU qui m'a aidé à finir mon projet ;

Que Je dédie à :

Mes très chers parents qui ont partagé avec moi mes joies et mes peines, que dieu les protège.

Mes chers frères qui m'ont vraiment soutenu dans mes études et qui comptent beaucoup pour moi.

Toutes mes sœurs grandes et petites que j'aime énormément.

Mon binôme Sabrina ainsi qu'à toute sa famille.

Tous mes amis et tous ceux qui me connaissent.

Toute la promotion 2016-2017.

A.Thiziri

Dédicaces

Avant tout je remercie le DIEU qui m'a aidé à finir mon projet ;

Que Je dédie à :

Mes très chers parents qui ont partagé avec moi mes joies et mes peines, que dieu les protège.

Mes chers frères qui m'ont vraiment soutenu dans mes études et qui comptent beaucoup pour moi.

A ma sœur que j'aime énormément.

Mon binôme Thiziri ainsi qu'à toute sa famille.

Tous mes amis et tous ceux qui me connaissent.

Toute la promotion 2016-2017.

A.Sabrina

Table des matières

Chapitre II : Intégration des bases de données par les liens et les bases de données

Chapitre III : Analyse et conception

Chapitre III : Réalisation

Liste des figures

Introduction générale

Le système d'information est une rubrique essentielle du fonctionnement d'une organisation et l'un des facteurs majeurs de sa réussite. Aujourd'hui un système d'information n'est pas constitué d'une seule application, il a énormément évolué surtout avec l'évolution des technologies logicielles (objet, composant, services web, …), l'évolution des technologies matérielles et aussi avec l'évolution des organisations (fusion, acquisition, mondialisation). Comme conséquences de tous ces facteurs et avec le temps, les systèmes d'information deviennent de plus en plus complexes et hétérogènes. [1]

L'accès à des environnements répartis et l'échange de données sont des enjeux fondamentaux de l'entreprise. La garantie d'un système informatique disponible, fiable, sécurisé, performant, adapté et intégré devient un avantage capital.

L'interopérabilité est la capacité de communiquer, exécuter des programmes, transférer des données entre différentes unités fonctionnelles de manière à ce qu'un utilisateur n'ait besoin que de peu ou pas de connaissances sur les caractéristiques de ces unités.

L'intégration est souvent considérée comme allant plus loin que l'interopérabilité, en forçant une certaine dépendance fonctionnelle des applications. Alors que des systèmes interopérables peuvent fonctionner indépendamment. Un système intégré est donc composé d'applications interopérables mais ces applications ne réalisent pas nécessairement un système intégré. En complément de ces deux vues systémiques, certains systèmes peuvent n'être que compatibles. Ces systèmes n'interfèrent pas directement entre eux. Cela n'implique donc pas qu'ils soient capables d'échanger des services. Des systèmes interopérables sont, par définition, compatibles, au moins en partie, mais l'inverse ne l'est pas. [2]

Les patrons de conception sont une solution aux problèmes de conception, en spécifiant des solutions claires et élégantes à ces problèmes, ils représentent le vocabulaire commun des de l'expertise des concepteurs de logiciels. Ce vocabulaire leur fournit un niveau de description adéquat pour discuter les choix de conception ou de restructuration du système.

- **Problématique :**

  Comment réaliser l'intégration des bases de données conçues indépendamment, tout en préservant leurs autonomies locales ?

Notre objectif est de réaliser l'intégration en combinant entre deux notions : les patrons de conception définie ci-dessous et les liens de base de données, pour assurer l'intégration des données.

Afin de mieux comprendre l'enchainement des concepts fondamentaux de notre travail nous l'avons structuré en quatre chapitres. Dans le premier chapitre nous allons aborder l'interopérabilité et l'intégration dont nous allons détailler le sujet de l'interopérabilité et ces différents types, ainsi que les domaines d'intégration. Ensuite nous allons parler dans le deuxième chapitre des patrons de conception, de leur but et intérêts, et les liens de base de données, comment les créer et les manipuler.

En arrivant au troisième chapitre dont nous allons mettre en œuvre une conception appropriée représentant l'ensemble des solutions envisagées.

Enfin le dernier chapitre qui est la réalisation, nous allons définir les différents outils de développent et langages de programmation en plus les réponses des requêtes émises.

I.1. Introduction :

Le système d'information est aujourd'hui un élément central du fonctionnement d'une organisation, Cela permet à ces organisations de mener à bien des projets communs, ou de réaliser des coopérations afin de pouvoir continuer à progresser.

Dans ce contexte, l'interopérabilité et l'intégration des applications sont devenues de plus en plus importantes afin de pouvoir satisfaire à la fois les besoins des citoyens et ceux des organisations.

L'intégration consiste à réunir des parties d'un système développé de façon séparées pour obtenir un seul système homogène, et tous cela pour assurer l'interopérabilité, qu'on définit comme étant la possibilité des systèmes fonctionner avec d'autres produits ou systèmes existants ou futurs.

Les organisations doivent à la fois connecter les nombreuses applications hétérogènes existantes, exploiter des données issues de systèmes d'information différents et définir de nouveaux processus tout en garantissant à terme la cohérence du système [3].

Dans ce premier chapitre nous allons définir les systèmes en générale et les systèmes répartis en particulier, ensuite nous se focaliserons sur l'intégration de ces derniers, les niveaux d'intégration, on parle ainsi de l'interopérabilité, ses différents niveaux, ses types et ses normes de standardisation, enfin nous citons les services web et ses intérêts dans l'intégration et l'interopérabilité des systèmes informatiques.

I.2. Définition d'un système d'information :

Un système d'information (SI) est un ensemble de ressources : personnels, logiciels, données et matériels permettant la manipulation, le stockage l'échange et la diffusion des informations au sein d'une organisation pour répondre aux besoins des utilisateurs.

Cette définition laisse entrevoir la complexité de ce système, alors il est primordial de donner importance au système informatique.

Un système informatique est un regroupement de moyens informatiques et réseaux ayant pour finalité la gestion de l'information ainsi que l'automatisation de ces systèmes d'informations dans les entreprises. [1]

I.3. Système réparti :

I.3.1.Définition :

Un système réparti aussi appelé distribué Est un ensemble de machines autonomes connectées par un réseau sans mémoire partagée, dont lequel les ressources ne sont pas centralisés. Il est considéré comme un système de haute performance.

Il est peut-être défini comme un réseau d'entités calculantes ayant le même but commun : qui est la réalisation d'une tâche globale à laquelle chaque entité contribue par ses calculs locaux et les communications qu'elle entreprend, sans même qu'elle ait connaissance du dessein global du système. [4]

I.3.2. Architecture d'un système réparti :

Les systèmes répartis recouvrent diverses architectures depuis les architectures Client / Serveur jusqu'aux architectures totalement réparties :

Exemple de système reparti client/serveur :[4]

Figure I.1 : Exemple de système reparti client/serveur

I.3.3. Domaines d'utilisation des systèmes repartis :

On utilise des systèmes repartis dans plusieurs domaines :

- Internet qui est le plus grand système réparti.
- Systèmes à flots de données (« workflow »).

- Communication et partage d'informations comme
- Les bibliothèques virtuelles, collecticiels pour le travail coopératif, la presse et le commerce électronique
- Applications temps réel pour :
  - Le contrôle de procédés industriels, avionique, Localisation de mobiles.
- Toutes les applications qui nécessitent des utilisateurs ou des données réparties. [5]

I.3.4. L'objectif des systèmes répartis :

On utilise des systèmes répartis pour plusieurs raisons dans lesquels nous avons :

- Mise en commun d'un grand nombre de ressources à faible coûts.
- Adaptation de la structure d'un système à celle des Applications (géographique ou fonctionnelle)
- Réalisation de systèmes à haute disponibilité et flexibilité.
- Partage de ressources (programmes, données, services).
- Réalisation de systèmes à grande capacité d'évolution.
- Economie de logiciels, de matériels
- Raisons intrinsèques : adapter le système à l'application, BDD réparties, Web, systèmes bancaires…
- Besoin de communication et partage d'informations, de ressources, des services, comme des fichiers, BDD, unités de stockage…etc. :
- Accélérer le calcul et alléger la charge.
- Augmenter la fiabilité par duplication de machines, de données ce qui nous amène à réaliser à réaliser des systèmes à haute disponibilité.
- Diminution des coûts, des délais, augmentation de la disponibilité des données pour une bonne qualité du service.
- Réaliser des systèmes ouverts, évolutifs : adjonction facile de matériels et logiciels.
- Accès à distance à une ressource indisponible en local.
- Accès aux mêmes ressources depuis tous les endroits du système. [6]

I.4. Les bases de données reparties :

I.4.1. Définition d'une base de données :

La base de données est un ensemble structuré et organisé de données permettant de stocker et de retrouver l'intégralité d'information en rapport avec un thème ou une activité ; Il représente un système d'information de telle sorte qu'elles puissent être consulté pars des utilisateurs ou par des programmes

I.4.2. Définition d'une base de données repartie :

Une base de donnée repartie est un ensemble structurés et cohérent de données gérées par des sites différents, stocké sur des processeurs distincts, elles apparaissent à l'utilisateur comme une base unique.

Un système de bases de données réparties ne doit donc en aucun cas être confondu avec un système dans lequel les bases de données sont accessibles à distance (selon le principe client-serveur). Il ne doit non pas confondu avec un système multi-bases. Dans ce dernier cas, chaque utilisateur accède à différentes bases de données en spécifiant leur nom et adresse, et le système se comporte alors simplement comme un serveur de bases de données et n'apporte aucune fonctionnalité particulière à la prochaine.

Au contraire, un système de base de données est suffisamment complet pour décharger les utilisateurs ou transaction sur des données gérées par différents SGBD sur plusieurs sites. [7]

I.4.3. Raisons de répartition de données :

- Les pressions pour la distribution :
  - Il devient impératif de décentraliser l'information ;
  - Augmentation du volume de l'information ;
  - Augmentation du volume des transactions ;
  - Besoin de serveurs de bases de données qui fournissent un bon temps de réponse sur des gros volumes de données.
- Prédiction d'accroissement :
  - Vitesse des microprocesseurs ;
  - Capacité des DRAM ;
  - Débit des disques.

- Pour améliorer le débit des E/S
  - Partitionnement des données ;
  - Accès parallèle aux données ;
  - Utiliser plusieurs nœuds (avec un bon coût/ performances), et les faire communiquer par un réseau. [6]

I.4.4. Buts de répartition de bases de données :

Les bases de données réparties ont une architecture plus adaptée à l'organisation des entreprises décentralisées :

- Plus de fiabilité : les bases de données réparties ont souvent des données répliquées, la panne d'un site n'est pas très importante pour l'utilisateur, qui s'adresse à autre site.
- Meilleures performances : réduire le trafic sur le réseau est une possibilité d'accroitre les performances. Le but de la répartition des données est de les rapprocher de l'endroit où elles sont accédées. Répartir une base de donnée sur plusieurs sites permet de répartir la charge sur les processeurs et sur les entrées/sorties.
- Faciliter l'accroissement : l'accroissement se fait par l'ajout de machines sur le réseau. [6]

I.5. Intégration :

I.5.1. Définition de l'intégration :

L'intégration est une modalité de résolution de problème, dans la mesure où elle nous permet de rassembler des vues divergentes et même conflictuelles. L'intégration désigne alors à la fois la mise en cohérence globale et la simplification en vue d'obtenir une unicité du référentiel et une homogénéité des informations. [8]

En d'autre terme L'intégration de système consiste à réunir au sein d'un même système d'information, des parties développées de façon séparées.

I.5.20. Intégration technologique et informationnel :

Le terme intégration apparaît dans cette dimension comme la simple incorporation d'une information au sein d'un système (matériel et/ou logiciel) qui a pour objet de la traiter.
Cette intégration informationnelle s'est construite autour de différentes approches :

- L'enrichissement des bases de données cherchant à intégrer les logiques applicatives à partir de règles évènementielles, de structures de données évoluées (graphes, arbres…), et d'interaction supportées par les données (triggers…).
- Le développement d'applications d'entreprise complète (ERP) intégrant et standardisant de nombreux domaines fonctionnels.
- La standardisation de protocoles et normes d'échanges réduisant le cout des transactions entre applications (EAI, CORBA, DCOM, J2EE…).

Face à ces situations, on peut distinguer plusieurs niveaux d'intégration au plan informatique et technologique. [9]

I.5.3. Niveaux d'intégration :

L'intégration peut concerner un ou plusieurs niveaux de système d'information et des applications de l'entreprise. Linthicum définit deux niveaux d'intégration : l'intégration au niveau des données et l'intégration au niveau des traitements et processus métiers. Ce dernier type peut être à son tour représenté par le niveau d'interfaces d'applications, le niveau des méthodes et le niveau des interfaces utilisateurs [10]. De son côté Janarthanan et al, définit les trois niveaux d'intégration : données, interface utilisateur et méthode. [11]

En fait, toutes ces classifications d'intégration sont très similaires, et comme nous pouvons le remarquer, les approches d'intégration par données, intégration par interfaces utilisateurs et intégration par méthodes.

I.5.3.1. Intégration des données :

L'intégration à ce niveau se réalise par la migration des données d'une base de données à une autre sans avoir besoin d'introduire des changements sur la logique des autres applications. Ceci peut être décrit tant qu'extraction de l'information à partir d'une base de données, avec un traitement éventuel, la transformation et l'injection de cette information dans une autre base de données. Ce type d'intégration est utilisé dans le cas où les applications à intégrer n'ont pas une interface des utilisateurs et des APIs.

Les trois grandes étapes de l'intégration de données sont :

- L'extraction depuis une source : une base de données, un fichier, une application, un email...
- La transformation : modification de format, jointure entre plusieurs sources, mise en conformité...

- Le chargement dans une cible : une autre base de données, une autre application, un autre fichier...etc. [12]

Il existe plusieurs technologies permettant d'adresser l'intégration au niveau données qui sont [11] : les middlewares d'accès aux bases de données (ODBC : Open DataBaseConnectivity, JDBC : Java DataBase Connectivity), les middlewares orientées messages (MOM, Message brokers), les outils de réplication de données, et les outils de fédération de données (ETL : Extract, Tansform and Load) ;

Nous parlons aussi de l'intégration avec les liens de base de données (DataBase Links) et les patrons de conception c'est ce que nous allons utiliser dans notre projet pour réaliser cette intégration.

I.5.3.2 Intégration de l'interface utilisateur :

Cette approche est aussi appelée screen-scrapping [11]qui permet de relier la logique

d'intégration dans l'interface des utilisateurs. L'intégration à ce niveau se fait par le développement d'une interface commune (partagée) pour exposer les différentes applications isolées en intra et inter-entreprises.

Un exemple de ce type d'intégration, un client qui veut s'inscrire dans un site (créer une boîte e-mail par exemple) peut accéder directement à la base de données du serveur distant et enregistrer toutes ses informations une fois qu'il clique sur le bouton « submit ». Il peut aussi extraire et échanger des données avec des applications distantes.

I.5.3.3 Intégration des traitements :

L'intégration à ce niveau permet à des applications de partager les fonctionnalités offertes par d'autres applications hétérogènes et indépendantes et cela par l'invocation des services qu'elles exposent. Le mécanisme le plus simple permettant de réaliser ce type d'intégration est l'utilisation des APIs (***A**pplication **P**rogramming **I**nterface*), ou encore les web services.

I.5.3.4. Intégration du processus métier :

L'intégration des applications par processus est une approche très intéressante, elle permet le partage de la logique d'affaires, ainsi que l'intégration de nouvelles applications à travers les applications existantes. Cela peut être fait pour créer une nouvelle interface ou intégrer une

nouvelle application. Cette approche est basée généralement sur la mise en place d'un moteur d'orchestration : BPI Engine ou encore un moteur de workflow : WFMS (***W**orkflow **M**anagement **S**ystem*), afin de relier et intégrer les applications dans un processus Commun.

I.5.4. Architectures d'intégration : [15]

Les architectures d'intégration constituent le socle de mise en œuvre des différentes technologies d'intégration, Dans la section qui suivra, nous présentons la notion de l'architecture orientée services, qui est considérée comme étant la meilleure solution pour l'intégration des applications, et la notion de service web, qui est proposée comme étant la technologie la plus efficace pour la réalisation de SOA.

I.5.4.1. Architectures Orientées Services (SOA) :

La notion des architectures orientées services ne date pas d'hier puisque la première utilisation de ce concept remonte vers les débuts des années 90. Il a néanmoins fallu attendre la fin de ces années pour que cette notion se popularise notamment grâce aux modèles D'Architectures de services web.

L'Architecture Orientée Service (SOA) est un style architectural dedéveloppement et d'intégration *dynamique* des applications d'entreprise. Elle permet notamment de structurer le système d'information en une collection de services, la brique de base de cette architecture. Ces derniers peuvent être librement (indépendamment du matériel) inter-opères et réutilisés par d'autres systèmes d'information via une interface commune, qui est le bus de services.

Les systèmes basés sur les architectures orientées services offrent une solution flexible au problème d'intégration des systèmes d'information, de protocoles, de sources de données, et de processus.

I.5.4.2.Architectures Orientées Services et l'Intégration :

Les systèmes basés sur l'architecture orientée services sont une réponse efficace aux problématiques d'intégration que rencontre l'entreprise en termes d'encapsulation, de réutilisation, de composition, d'interopérabilité, et de réduction de couplage entres les systèmes d'information d'entreprise.

I.5.5. Services Web :

Les services web sont des composants logiciels encapsulant des fonctionnalités métier de L'entreprise et accessibles via des protocoles standards du web. Une définition plus précise des services web est fournie par le magazine de développement Programmez. Il définit un service web comme « une manière standardisée d'intégration des applications basées sur le Web en utilisant les standards ouverts XML, SOAP, WSDL, UDDI et les protocoles de transport de l'Internet. XML est utilisé pour représenter les données, SOAP pour transporter les données, WSDL pour décrire les services disponibles, et UDDI pour lister les fournisseurs de services et les services disponibles »

I.6. Interopérabilité :

I.6.1. Définition d'interopérabilité :

L'interopérabilité est la capacité que possèdent deux ou plusieurs systèmes ou composants à échanger des informations puis à exploiter les informations venant d'être échangées, il signifie la possibilité d'opérer réciproquement.[3]

I.6.2. L'interopérabilité en informatique :

L'interopérabilité en informatique est la capacité d'un produit ou d'un système informatique à fonctionner avec d'autres produits ou systèmes existants ou futurs. C'est en d'autres mots, la possibilité des systèmes à fonctionner ensemble et à communiquer entre eux, et ce sans restriction d'accès ou de mise en œuvre.

Pour permettre cette communication, il est donc nécessaire d'utiliser un langage commun. On parle alors de normes et de standards ouverts, que chaque produit ou système se devra respecter pour assurer son interopérabilité.

L'interopérabilité est générale et ne concerne pas a priori des éléments ou systèmes particuliers. Elle existe au travers de normes

L'interopérabilité est peut-être totale ou partielle, on peut déterminer dans quelle mesure des systèmes sont interopérables en jugeant de leur respect des normes, si un logiciel ne respecte qu'une partie d'une norme il ne pourra peut-être pas dialoguer correctement avec un autre programme, voire pas du tout. Seul le respect d'une norme donnée conduit à une interopérabilité réelle. [15]

I.6.3. Interopérabilité et compatibilité :

L'interopérabilité n'est pas une simple compatibilité. Il ne s'agit pas seulement de permettre à deux systèmes de communiquer entre eux, mais aussi de lire et de modifier les informations et contenus de manière fiable en garantissant que n'importe quel système présent ou futur puisse s'interconnecter.

On doit distinguer la compatibilité de l'interopérabilité, on dit que deux systèmes sont compatibles s'ils peuvent se comprennent et se communiquer entre eux c'est-à-dire le format des données d'un logiciel sont compatibles avec le format des données de l'autre, donc la compatibilité résulte d'un travail de traduction d'un format vers un autre, mais l'interopérabilité est générale, et ne peut pas être une interopérabilité avec quelque chose de particulier, elle s'appuie sur des formats ouverts qu'elle doit absolument respecter. [16]

I.6.4. Les normes et les standards de l'interopérabilité :

Une norme définit les règles, caractéristiques, et recommandations applicables à des produits, services, méthodes, processus ou organisations, émise par un organisme de normalisation comme ISO et l'AFNOR, destinée à harmoniser l'activité d'un secteur comme le langage C, le HTML…etc.

Un standard est un format élaboré par un petit nombre d'acteurs et adopté par des consortiums, des forums, c'est-à-dire des organisations non officielles comme par exemple les CD qui sont standardisés, ils sont lisibles sur n'importe quel lecteur CD ou lecteur DVD.

Les standards ouverts sont diffusés gratuitement, librement comme PDF sous Adobe, Le standard Open Document Format (ODF) du RGI qui est le seul standard d'applications bureautiques totalement ouvert et indépendant de tout éditeur basé sur le XML pour afficher, stocker et éditer des documents.

Alors que les standards fermés sont des modèles propriétaires comme le format de fichier Word de Microsoft.

La norme joue un doublerôle dans la réalisation de l'interopérabilité. Elle est tout d'abord un indicateur de la façon dont le dialogue entre les différents acteurs doit s'opérer. Elle est également une passerelle de communication, qui va pouvoir éventuellement s'adapter aux besoins changeants des éléments. Quand deux systèmes sont créés pour respecter une norme ou un standard ouvert, ils sont interopérables. [16] [voir annexe A]

I.6.5. Domaines de l'interopérabilité :

L'interopérabilité est considérée comme une importante voire critique dans de nombreux domaines dont l'informatique comme les réseaux téléphoniques et l'internet, le médical au sens large, l'économie, les activités ferroviaires, l'électrotechnique, l'aérospatial, le domaine militaire et l'industrie en générale.

I.6.6. Réalisation de l'interopérabilité :

L'interopérabilité peut se réaliser de différentes manières : soit par le biais d'une collaboration suffisante de la part de l'acteur premier ou dominant, qui donne une information complète sur les interfaces de ses systèmes ; soit par la définition commune de ces interfaces, à travers des formats et protocoles ouverts et normalisés. Si l'utilisation de formats ouverts n'est pas la seule voie à l'interopérabilité, elle est sans conteste la meilleure. [17]

I.6.7. Les différents niveaux de l'interopérabilité :

L'interopérabilité fait intervenir trois niveaux distincts mais indissociables :
Le premier niveau est **l'interopérabilité technique**, qui vise le transport de données et la sécurisation du transport ; le deuxième est **l'interopérabilité syntaxique**, qui concerne le format des données ;et un troisième qui est **l'interopérabilité sémantique**, qui vise à faciliter la compréhension mutuelle entre intervenants (hommes et systèmes d'information) dans les échanges. La mise en œuvre de l'interopérabilité sémantique requiert donc en premier lieu un consensus entre les professionnels de santé sur les contenus à échanger ou à partager, et l'harmonisation des terminologies utilisées en commun. [17]

I.6.7.1. L'interopérabilité technique :

Elle est définie pour rendre possible la circulation des données entre systèmes différents, Cette interopérabilité est ce qui permet à divers outils de communiquer. En ce qui concerne les logiciels, le terme sert à désigner la capacité de différents programmes à échanger des données au moyen d'une série de formats d'échange et de protocoles standardisés et partagés.

Cette interopérabilité passe ainsi par la définition d'interfaces standardisées rendant possible une relative convergence entre des systèmes hétérogènes. Exemples des standards qui permet cette interopérabilité comme les services web REST et SOAP [voir annexe A], et

plus globalement l'architecture orientée services, le CORBA, le COM ou ODBC qui sont autant d'intergiciel (middleware). [18] [Voir annexe B]

I.6.7.2. L'interopérabilité syntaxique :

Elle propose une intégration de premier niveau, que l'on peut appeler intégration syntaxique, en définissant notamment la nature, le type de messages échangés, aussi les formats de données de « bas niveau », comme le stockage des caractères alphanumériques dans un code comme l'ASCII, aussi les standards de formatage des données comme XML et SQL, et des standards de représentation des systèmes d'informations comme UML et BPMN.

Elle conduit à la notion de système ouvert permettant d'assumer l'hétérogénéité des composants (interfaces, langages déprogrammation, etc.). Ce premier niveau est toutefois des significations perçues par les différents utilisateurs d'un système. Une intégration de second niveau, l'intégration sémantique basée sur l'interopérabilité sémantique, est donc nécessaire qui prolonge et complète la précédente. Les interopérabilités technique et syntaxique ne suffisent pas, ce qui a conduit à investir un troisième niveau, celui de l'interopérabilité sémantique ;[19]

I.6.7.3. L'interopérabilité sémantique :

Concerne la capacité à interpréter la nature et le sens des informations échangées. Il faut donc donner une sémantique aux informations, et la transmettre à tous les systèmes concernés par les échanges, ces systèmes combinent alors les informations reçues avec d'autres afin de les traiter de manière appropriée dans le respect de cette sémantique. Il ne s'agit donc plus seulement de permettre la communication entre systèmes et applications informatiques, mais de garantir une cohérence entre leurs manières de décrire et d'interpréter les informations échangées.

Cette l'interopérabilité consiste à reconnaître aussi que deux termes se réfèrent à la même entité, même si différentes terminologies sont utilisées, C'est à ce niveau que les ontologies, l'interopérabilité des métadonnées et le système de médiation peuvent être utilisées pour confronter cette problématique :[19]

- ✓ **Les ontologies :**

Une ontologie sert de moyen efficace pour la représentation de la sémantique des données de sorte qu'elle soit interprétable, sont ainsi développées pour fournir aux diverses

applications un moyen de trouver des significations communes entre différents vocabulaires[20]. Il existe trois techniques pour la réalisation des ontologies qui sont :

- **Le mapping d'ontologies** : qui a comme objectif la représentation des correspondances entre les ontologies. Ceci permet, par exemple, d'interroger des bases de connaissances hétérogènes en utilisant une interface commune ou en transformant des données entre différentes représentations ; [voir annexe B]
- **La fusion d'ontologies :** qui permet de créer une nouvelle ontologie, appelée l'ontologie fusionnée capturant les connaissances des ontologies d'origine. Le défi est alors d'assurer que toutes les correspondances et les différences entre les ontologies soient correctement prises en compte dans l'ontologie résultante ;
- **L'alignement d'ontologies** : il permet d'établir des liens sémantiques entre les concepts et des relations inter-ontologies, est un processus de découverte des correspondances entre deux ontologies source.[21]

✓ **L'interopérabilité des métadonnées :**

C'est un ensemble structuré d'informations décrivant une ressource quelconque. Elles permettent de faciliter l'accès au contenu informationnel d'une ressource informatique, elles sont stockées dans un registres de métadonnées, qui est mis en œuvre par la norme ISO/CEI 11179 du gouvernement américain.[22]

Le « Dublin Core » est la principale initiative visant à la convergence des éléments de métadonnées à utiliser. C'est un schéma de métadonnées générique qui peut servir de base à des registres de métadonnées.

✓ **L'approche médiation :**

La médiation repose sur un composant essentiel appelé le médiateur. Ce dernier est un patron de conception qui fournit une interface unifiée pour un ensemble d'interfaces d'un sous-système. Est un module logicielqui est chargé de répondre àdes besoins à partir de connaissances mises à sa disposition, enrecevant directement la requête d'un usager et devant la traiter, celui-ci doit localiser l'information nécessaire pour répondre à la requête, résoudre les conflits schématiques et sémantiques, interroger les différentes sources et intégrer les résultats partiels dans une réponse homogène et cohérente ;

Un composant secondaire, appelé wrapper, sert d'interface avec les SI, ce wrapper résout les conflits syntaxiques en présentant les données dans le modèle de médiation. On distingue deux types de médiation selon la manière de résoudre les conflits de données :

- La médiation de schémaqui construit au préalable une base d'informations prenant en compte les SI participants pourpermettre au médiateur de faire son travail d'intégrateur
- La médiation de contexte où lemédiateur est guidé par des informations à caractère sémantique pour résoudre dynamiquementune requête sans connaissance à priori des SI participants.[23]

I.6.8. Les types d'interopérabilité :[24]

Il existe plusieurs types de l'interopérabilité, ci-dessous on cite quelques types :

I.6.8.1. Interopérabilité des bases de données :

Notre intérêt dans ce projet est de permettre à un utilisateur d'interroger n'importe quelle base de données sans se soucier du logiciel ni de la structure qui accueillent ces données : l'interrogation se fait via l'ontologie, puis un moteur retranscrit la requête pour la base de données cible en utilisant la technique mapping [voir annexe B], enfin les résultats sont renvoyés à l'utilisateur en respectant la structure de l'ontologie. Avec ce procédé, il est donc techniquement possible d'interroger plusieurs bases en même temps, même si leur format natif diffère complètement.

I.6.8.2. L'interopérabilité multilingue :

Avec la diffusion d'Internet, les échanges mondiaux de fichiers se sont multipliés, ce qui pouvait poser des problèmes d'interopérabilité, pour les fichiers textes ; pour cette raison Unicode a été créé.

Dans Unicode [voir annexe C], les métadonnées sont enregistrées dans le format de codage de caractères UTF-8, qui accepte la plupart des navigateurs web depuis 1998.

I.6.8.3. L'interopérabilité du web :

Différents navigateurs et versions de navigateurs peuvent être utilisés pour consulter le web,Les serveurs web doivent donc être interopérables avec différents logiciels clients. Pour cette raison, ils utilisent un langage connu, le HTML, combiné à d'autres standards, tels que HTTP, SVG (Scalable Vector Graphics) ou JPEG.

I.7. Conclusion :

L'intégration des systèmes et l'interopérabilité des services est la meilleure solution, pour mieux garantit la cohérence, la réactivité, l'adaptabilité et la maîtrise de l'hétérogénéité du matériel, des logiciels, des données, des processus, des applications dans les environnements repartis.

Ce chapitre a été consacré à la présentation les concepts généreux d'intégration y compris ses différents niveaux, ses technologies. De plus nous avons définis l'interopérabilité ainsi que ses niveaux tels que l'interopérabilité technique et sémantique, aussi ses avantages et ses inconvénients. [5]

Le prochain chapitre, a pour objectif de présenter une étude approfondie sur les liens de base de données et nous allons citer les patrons de conception qui ont pour but de régler les problèmes liés à l'intégration.

II. 1. Introduction :

Le but ultime de la programmation orienté objet est de permettre aux programmeurs d'ajouter des fonctions sans tout réécrire, faciliter la maintenance des programmes et le plus important est de réutiliser des programmes ou des méthodes déjà testées sans être obligés à réécrire tout le code, c'est de là qu'on peut parler des patrons de conception.

L'objectif de notre thème est la création et la manipulation des liens entre les différentes bases de données existante. Un lien de base de données (DB Link) est un objet Oracle permettant de créer un lien entre plusieurs bases.

Les patrons sont des solutions éprouvées à des problèmes spécifiques et récurrents, il décrit un problème devant être résolu, une solution, et le contexte dans lequel cette solution est considérée. En génie logiciel un patron de conception (design pattern) est une solution classique à un problème récurrent dans la conception d'applications orientées objet. [5]

Dans ce présent chapitre nous allons présenter les différents patrons de conception selon GoF, ensuite nous allons aborder les liens de base de données et ses différents types existants.

II. 2. Les patrons (patterns) :

II.2.1. Définition :

Un patron décrit à la fois un problème qui se produit très fréquemment dans l'environnement et l'architecture de la solution à ce problème de telle façon que l'on puisse utiliser cette solution des milliers de fois sans jamais l'adapter deux fois de la même manière.

Appelés également « motifs », ils représentent des modèles permettent de résoudre des problèmes de modélisation, à différentes échelles. [25]

II.2.2. Présentation d'un pattern : [25]

Un patron généralement possède quatre éléments essentiels qui sont :

- **Nom du pattern :**

  Utilisé pour décrire le pattern, ses solutions et les conséquences en un mot ou deux

- **Problème :**

  Description des conditions d'applications. Explication du problème et de son contexte.

- **Solution :**

  Description des éléments (objets, relations, responsabilités, collaboration) permettant de concevoir la solution au problème ; utilisation de diagrammes de classes, de séquences, …

- **Conséquences :**

  Description des résultats (effets induits) de l'application du pattern sur le système.

II.2.3. Topologie des Patrons :

Il existe cinq types de patrons qui sont :

- **Idiomes ou patterns de codage :**

Les Idiomes sont des patterns de bas niveau spécifiques à un langage de programmation. Ils décrivent comment implémenter des aspects particuliers des composants ou leurs relations, avec les caractéristiques d'un langage donné.

- **Anti-patterns :**

Un anti-pattern est une solution souvent utilisée, mais largement inefficace pour un problème. Le terme a été utilisé à l'origine pour désigner un modèle qui a mal tourné. Tout comme un modèle viable décrit le passage d'un problème à une solution valide, un anti pattern décrit le passage d'un problème à une mauvaise solution. De plus, en ajoutant plus de difficultés à celles qui existaient à l'origine. Exemple : **Anti-patrons de développement :**

Abstraction inverse : L'abstraction inverse se produit lorsque l'on construit un objet logiciel avec une interface qui n'offre pas des fonctions dont les développeurs qui l'utilisent ont besoin, alors qu'il pourrait les offrir. L'interface n'offre que des fonctions plus complexes. Le

résultat est que l'utilisateur de l'objet doit se servir des fonctions complexes fournies par l'objet pour programmer un comportement simple.

Exemple : avoir un objet qui ne fait que des calculs en virgule flottante, et être obligé d'utiliser cet objet pour faire du calcul avec des entiers.

Patron d'architecture :

Un patron d'architecture représente les schémas d'organisation structurelle de logiciels correspondent à des styles architecturaux classiques comme pipes, filters, brokers, blackboard, MVC, ….etc. il concerne la structure générale d'un logiciel, sa subdivision en unités plus petites, comporte des guides de bonnes pratiques et des règles générales qui ne peuvent pas être traduites directement en code source.

Ils interviennent dans les plus hauts niveaux du développement logiciel ils ont souvent été découverts dans de gros projet.

- **Schéma de présentation des patterns d'architecture :**
  - Nom
  - Exemple
  - Contexte
  - Problème
  - Solution
  - Structure
  - Dynamique (scénarios)
- **Patron organisationnel :**

Modes d'organisation sont des structures de parenté, généralement à une organisation professionnelle, qui aident l'organisation professionnelle, et aident l'organisme à atteindre ses objectifs. Les motifs sont généralement inspirés par l'analyse de plusieurs organisations professionnelles et de trouver des structures communes dans leurs réseaux sociaux. Ces modèles sont collectés et organisés en langues de motifs, qui sont publiés en tant que fondation pour l'amélioration des processus et la conception organisationnelle, en grande partie dans la communauté de développement logiciels.

- **Patron de conception :**

Un patron de conception est un groupe d'objets coopérants liés par des relations et des règles qui expriment les liens entre un contexte, un problème de conception qui apparaît répétitivement et sa solution, comme c'est défini dans ce qui suit :

II.2.5. Les Patrons de conception (design patterns) :

II.2.5.1. Historique :

L'origine des Design Patterns remonte au début des années 70 avec les travaux de l'architecte Christopher Alexander. Celui-ci remarque que la phase de conception en architecture laisse apparaître des problèmes récurrents. Il cherche alors à résoudre l'ensemble de ces problèmes liés à des contraintes interdépendantes (solidité de la structure, étanchéité...).

Pour cela Alexander établi un langage de 253 patterns, qui couvrent tous les aspects de la construction (comme par exemple la façon de concevoir une charpente).

Dans les années 90, l'idée de Christopher Alexander va être reprise et étendue au domaine de la conception des logiciels. Le concept de Design Pattern est développé dans un ouvrage publié en 1995 par le « Gang of Four ». Cette équipe qui regroupe Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides intitule ce livre « Design Patterns : Elements of Reusable Object-Oriented Software ». Celui-ci présente 23 Design Patterns qui font aujourd'hui référence dans le monde de l'informatique. [25]

II.2.5.2. Définition de patron de conception :

En développement logiciel, un patron de conception est une bonne pratique en réponse à un problème de conception d'un logiciel. Il décrit une solution standard, utilisable dans la conception de différents logiciels.

Un patron de conception est issu de l'expérience des concepteurs de logiciels. Il décrit un arrangement récurrent de rôles et d'actions joués par des modules d'un logiciel, et le nom du patron sert de vocabulaire commun entre le concepteur et le programmeur, D'une manière analogue à un motif de conception en architecture, le patron de conception décrit les grandes lignes d'une solution, qui peuvent ensuite être modifiées et adaptées en fonction des besoins.

Les patrons de conception décrivent des procédés de conception généraux et permettent en conséquence de capitaliser l'expérience appliquée à la conception de logiciel. Ils ont une influence sur l'architecture logicielle d'un système informatique. [26]

II.2.5.3 Le choix du patron et les problèmes de leur utilisation :

Les patrons de conception se différent au niveau de complexité. En effet, l'application, de certains d'entre eux, n'est pas un processus évident. Il faut cerner le problème pour pouvoir identifier la solution adéquate et donc le patron adéquat. Une fois que le patron à utiliser est identifié, il faut l'appliquer au modèle concerné. Pour cela, le concepteur doit pouvoir définir le rôle de chaque élément du modèle pour établir une correspondance avec les éléments du patron.[25]

Une fois le patron appliqué, il faudra vérifier si la sémantique du modèle est toujours respectée, comme il faudra s'assurer que de futures modifications ne violent pas les contraintes sémantiques et structurelles imposées par le patron.

L'implémentation des patrons dans un langage de programmation nécessite une mise en correspondance entre les éléments de conception du patron et les constructions du langage de programmation. Cela n'est pas toujours aussi évident, dans le cas du langage Java, par exemple, on ne peut pas implémenter l'héritage multiple.

II.2.5.4. Mise en œuvre des patrons :

Une fois que nous avons choisi un modèle de conception, on lit le patron une fois pour une vue d'ensemble, on prête une attention particulière aux sections applicabilité et conséquences pour s'assurer que le modèle est droit pour votre problème.

- On retourne et on étudie les sections Structure, Participants et Collaborations.
- On assure bien comprendre les classes et les objets dans le modèle et comment ils se rapportent les uns aux autres.
- On regarde la section Code d'exemple pour voir un exemple concret de patron dans le code. L'étude du code nous aide à apprendre comment implémenter le modèle.
- On choisit les noms pour les participants de modèle qui sont significatifs dans l'application de contexte.
- On définit les classes. On déclare leurs interfaces, établir leur héritage et définir les variables d'instance qui représentent les données et références d'objet. On identifie les classes existantes dans notre application.
- On définit les noms propres à l'application pour les opérations dans le modèle. Encore ici, les noms dépendent généralement de la demande. On utilise les responsabilités et les collaborations associées à chaque opération comme guide. Être aussi cohérente dans nos conventions de dénomination. Par exemple, nous pouvons utiliser le préfixe "Créer" préfère désigner une méthode d'usine.
- Mettre en œuvre les opérations pour s'acquitter des responsabilités et collaborations dans le modèle. [25]

II.2.5.5. Les avantages des patrons de conception :

L’utilisation des Design Patterns offre de nombreux avantages comme :

- Ils permettent de répondre à un problème de conception grâce à une solution éprouvée et validée par des experts, ainsi on gagne en rapidité et en qualité de conception ce qui diminue également les coûts ;
- Les Design Patterns sont réutilisables et permettent de mettre en avant les bonnes pratiques de conception ;

- Les Design Patterns étant largement documentés et connus d'un grand nombre de développeurs ils facilitent également la communication. Si un développeur annonce que sur ce point du projet il va utiliser le Design Pattern Observateur il est compris des informaticiens sans pour autant rentrer dans les détails de la conception (diagramme UML, objectif visé...etc.) ;
- Niveau d'abstraction plus élevé améliorant la construction des logiciels ;
- Réduction de la complexité du développement de logiciel

II.2.5.6. Technique de modélisation d'un patron de conception :

Lors de la modélisation d'un patron de conception on doit prendre en compte sa vue interne et sa vue externe.

Vu de l'extérieur, un patron de conception est présenté par une collaboration paramétrée. Comme une collaboration, un pattern fournit un ensemble d'abstraction dont la structure et le comportement coopèrent pour exécuter une fonction utile. Les paramètres de la collaboration désignent les éléments qu'un utilisateur de ce pattern doit lier. Le pattern de conception devient alors un canevas qu'on utilise dans un contexte particulier en fournissant des éléments qui correspondent aux paramètres de canevas.

Vu de l'intérieur, un patron de conception est simplement une collaboration et il est représenté avec ses parties structurelles et comportementales. En générale, on modélise l'intérieur de cette collaboration à l'aide d'un ensemble d'interactions (pour l'aspect comportemental). Les paramètres de la collaboration désignent certains de ces éléments structurels, qui l'aide de l'abstraction tirée de ce contexte.

Pour modéliser un pattern il faut :

- Identifier la solution courante du problème récurrent et la réifier comme un mécanisme ;
- Modéliser le mécanisme comme une collaboration en fournissant ses aspects structurels et comportementaux ;
- Identifier les éléments du pattern de conception qui doivent être rattachés à des éléments dans un contexte spécifique et les représenter comme des paramètres de la collaboration.

II.2.5.7. Patron de conception du GoF :

Les Designs Patterns GoF ont été conçus par quatre informaticiens Erich Gamma, Richard Helm,Ralph Johnson et John Vlissides, surnommés le Gang of Four (d'où le terme GoF pour ces patterns) ces derniers sont les auteurs du livre {design patterns : Elements of Reusable Object-Oriented Software (Addison-Wesley, 1st Edition, 1994. isbn: 0- 201-63361-2)}.

cet important ouvrage aborde le sujet de la programmation orientée objet et introduit le concept des patrons de conception et est devenu un classique de la littérature en génie logiciel avec de nombreuses rééditions.

Ils proposent des solutionsélégantes, et toujours différentes pour résoudreplusieursproblèmes récurrents rencontrés par lesarchitectes logiciels ces solutions sont présentées en 23 orienté objet design patterns de programmation.

II.2.5.8. Description d'un patron conception du GOF :

GOF ont proposé un formalisme pour la représentation des patrons, ils ont adopté une représentation uniforme et structurée. Chaque patron est décrit et documenté de la même façon. Les propriétés utilisées sont :

- **Nom**: le nom significatif du patron.
- **Intention :** décrit ce que fait le patron, son but, quel problème particulier il résout.
- **Alias**: énumère, s'il en existe, d'autres noms communs du patron.
- **Motivation**: donne un exemple de problème de conception pouvant être résolu par application du patron. L'illustration par un exemple facilite la compréhension de la description abstraite (donnée par la suite) du patron.
- **Indications d'utilisation**: décrit les situations où on peut utiliser le patron.
- **Structure** : décrit la structure du patron en utilisant des diagrammes de classes. Des diagrammes de collaboration sont aussi utilisés pour la description de l'interaction entre les classes du patron.

- **Participants** : définit les classes (ou objets) intervenantes dans le patron et leurs responsabilités.
- **Collaborations**: décrit comment les participants interagissent pour assumer leurs responsabilités.
- **Conséquences**: décrit comment le patron atteint ses objectifs, quels sont les compromis et les résultats de son utilisation.
- **Implémentation** : présente des astuces, techniques et pièges qu'il faut connaître lors de l'implémentation du patron. Cette partie propose aussi, quand il en existe, des solutions typiques du langage utilisé.
- **Exemple de code**: donne des fragments de code pour illustrer la façon dont on peut implémenter le patron.
- **Utilisations connues** : contient des exemples d'utilisation du patron dans des systèmes réels.
- **Patrons apparentés** : cite les patrons qui sont reliés avec celui en cours de traitement et leurs différences.

II.2.5.9. Catégories de patron de conception du GOF :

Le GoF présente 23 design pattern organisés en 3 classes selon leur rôle et leur domaine d'application (classe vers objet) :

- **Rôle :**

- **Patron de Création :**

Description de la manière dont un objet ou un ensemble d'objets peuvent être créés, initialisés, et configurés : isoler le code relatif à la création, l'initialisation afin de rendre l'application indépendante de ces aspects ;

- **Patron de Structure :**

Description de la manière dont doivent être connectés des objets de l'application afin de rendre ces connections indépendantes des évolutions futures de l'application : découplage de l'interface et de l'implémentation de classes et d'objets ;

- **Patron de Comportement :**

Description de comportements d'interaction entre objets : gestion des interactions dynamiques entre des classes et des objets.

- **Domaine :**

- **Portée de classes** : relation entre classes et leurs sous-classes (héritage), Ces relations sont établies statiquement.
- **Portée d'objets :** Larelation entre classes et leurs sous-classes (composition), Ces relations sont établies dynamiquement et modifiées à l'exécution.

II.2.5.10. Catalogue de patrons de conception (GoF) :

L'organisation de 23 patrons de conception de GoF est comme suit :

		Catégorie		
		Création	Structurel	comportemental
portée	Classe	Fabrication	Adaptateur (classe)	Interprète
				Patron de méthode
	Objet	Fabrique abstraite	Adaptateur (objet)	Chaîne de responsabilités
		Monteur	Pont	Commande
		Prototype	Composition	Iterateur
		Singleton	Décorateur	Médiateur
			Façade	Mémento
			Poids mouche	Observateur
			Procuration	Etat
				Stratégie
				Visiteur

Figure II.1 : Tableau des types de patrons de conception

Voici un résumé des patrons présentés dans le tableau ci-dessous avec une brève définition :

II.2.5.10.1. Patrons de Création :

- **La fabrique abstraite (Abstract factory)** : permet à une interface de créer des objets sans leurs classes concrètes.
- **Le monteur (Builder)** : sépare la construction d'un objet complexe de ses représentations comme ça, le même processus de création permet de créer différentes représentations.
- **La fabrication (Factory method)**: Définie une interface qui crée un objet mais permet aux classes qui héritent de cette interface de choisir laquelle instancier.

- **Le Prototype** : spécifie le type d'objet à créer en utilisant une instance prototypique et créer de nouveaux objets en copiant ce prototype.
- **Le Singleton** : assure qu'une classe a une seule instance et permet un accès global à cette classe.

II.2.5.10.2. Patrons de Structure :

L'Adaptateur (Adapter): Convertit une interface d'une classe à une autre interface que le client attend.

- Représentation du patron en UML :

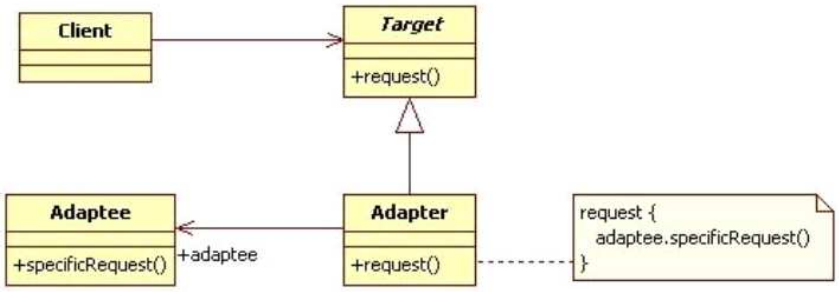


Figure II.2 : Patron de conception adaptateur

- **Le Pont (Bridge)**: découple une abstraction de son implémentation de manière que les deux peuvent varier indépendamment.
- **Le Composite**: compos des objets a des arbres de structure pour représenter partie entier hiérarchies, ils permes aux clients de traiter les objets individuels et les compositions des objets d'une manière uniforme.
- **Le Décorateur (Decorator)**: Attache dynamiquement des responsabilités additionnelles à un objet.
- **La Façade (aussi Frontal)**: donne une interface unifiée à un ensemble d'interfaces dans un sous-système, définie une interface d'un niveau élevé qui fait que le sous-système devient facile d'utilisation.
- **LePoidsmouche** (**Flyweight)** : utilise le partage pour supporter un grand nombre d'objet 'fine-grained' efficacement.

- **LeProcuration** (**Proxy)**: Fournit un remplaçant ou un paramètre fictif à un autre objet pour contrôler son accès.

II.2.5.10.3. Patrons de comportement :

- **La Chaîne de responsabilités (Chain of responsabilité)**: évite de coupler l'expéditeur d'une requête avec son receveur en donnant la chance à plus d'un objet de traiter cette requête. Enchaîne les objets receveurs et passe la requête à travers la chaîne jusqu'à ce qu'un objet la traite.
- **La Commande (Command) :** encapsule une requête en un objet, comme ça il permet de paramétrer les clients avec différentes requêtes, fille d'attendes ou historique de requêtes, et d'assurer le traitement des opérations réversibles.
- **L'Interprète (Interpreter) :** définit pour un langage donné une représentation pour sa grammaire avec un interpréteur qui utilise cette représentation pour interpréter des phrases du langage.
- **L'Itérateur (Iterator) :** fournit une manière pour accéder séquentiellement aux éléments d'un objet agrégé sans exposer sa représentation intérieure.
- **Médiateur (Mediator)** : définit un objet qui encapsule l'interaction d'un ensemble d'objets entre eux. Il permet la perte de couplage en empêchant les objets de s'interférer les uns les autres explicitement, et il permet à l'utilisateur de varier leurs interactions indépendamment.
- **LeMémento** : sans violer l'encapsulation, capture et extériorise un état interne d'un objet de telle façon que cet objet peut être restauré à cet état plus tard.
- **L'observateur (Observer)** : définie une dépendance « un à plusieurs » entre les objets de façon à ce que si un objet change d'état, tous les objets dépendants de lui sont mis à jour automatiquement.
- **L'état (State)** : permet à un objet de changer son comportement quand son état interne change.
- **La stratégie (Strategy) :** définie une famille d'algorithmes, encapsule chacun d'eux, et les rendre interchangeables. Il laisse l'algorithme varier indépendamment des clients qu'ils l'utilisent.

- **Le patron de méthode (Template method) :** définie le squelette d'un algorithme dans une opération, différant quelques étapes aux classes qui héritent. Il permet à ces classes de redéfinir certaines étapes d'un algorithme sans changer la structure de ce dernier.

- Représentation du patron en UML :

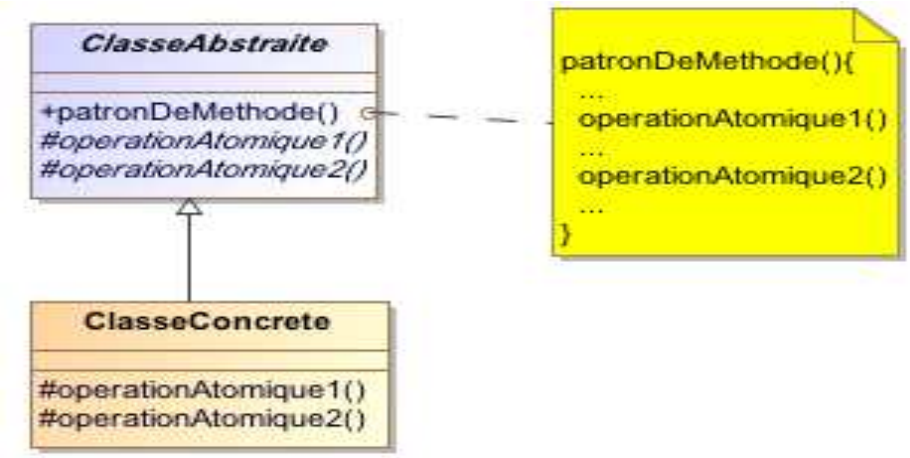


Figure II.3 : Patron de conception template method

- **Le visiteur (Visitor)** : représente une opération qui sera exécutée dans un élément d'une structure d'objet. Il permet de définir une nouvelle opération sans changer les classes des éléments sur lesquels elles opèrent.

II.2.5.11. Les avantages des patrons de conception :

L'utilisation des Design Patterns offre de nombreux avantages comme :

- Ils permettent de répondre à un problème de conception grâce à une solution éprouvée et validée par des experts, ainsi on gagne en rapidité et en qualité de conception ce qui diminue également les coûts ;
- Les Design Patterns sont réutilisables et permettent de mettre en avant les bonnes pratiques de conception ;
- Les Design Patterns étant largement documentés et connus d'un grand nombre de développeurs ils facilitent également la communication. Si un développeur annonce

que sur ce point du projet il va utiliser le Design Pattern Observateur il est compris des informaticiens sans pour autant rentrer dans les détails de la conception (diagramme UML, objectif visé...etc).

II.3. Lien de base de données répartis :

II.3.1. Définition :

d. Le pointeur de lien est effectivement défini comme une entrée dans une table de dictionnaire de données. Pour accéder au lien, vous devez être connecté à base de données locale qui contient l'entrée du dictionnaire de données.

Une base de données peut utiliser une liaison mémorisée dans la base de données A pour accéder aux informations de la base de données B, mais les utilisateurs connectés à la base de données B ne peuvent pas utiliser le même lien pour accéder aux données de A, ils doivent définir un lien qui est stocké dans le dictionnaire de données de leur base.

Dans la base de données A. Si les utilisateurs locaux sur la base de données B veulent accéder à des données sur la base de données A, une connexion de liaison de base de données permet aux utilisateurs locaux d'accéder aux données sur une base de données distante. Pour établir cette connexion, chaque base de données dans le système distribué doit avoir une expérience unique, nom global de base de données dans le domaine du réseau. Le nom de la base de données mondiale unique identifie un serveur de base de données dans un système distribué. [28]

II.3.2. L’avantage des liens de base de données :

Le grand avantage des liens de base de données est qu'ils permettent aux utilisateurs d'accéder aux autres utilisateurs d’objets dans une base de données distante. En d'autres termes, un utilisateur local peut accéder avec un lien vers une base de données distante sans être un utilisateur sur la base de données distante. [28]

II.3.3. Les types de liens :

Les liens de base de données sont privés, publiques ou globaux. Ces types de liens diffèrent selon les utilisateurs et leurs autorisations à accéder à la base distante :

- **Lien privé :** Crée un lien dans un schéma spécifique de base de données locale. Seule le propriétaire d'une base de données privée peut utiliser ce lien pour accéder aux objets de base de données dans la base de données distante correspondante.

L'utilisateur peut voir les liens de base de données existants à travers les requêtes suivantes :

- SELECT * FROM DBA_DB_LINKS
- SELECT * FROM ALL_DB_LINKS
- SELECT * FROM USER_DB_LINKS

- **Lien publique :** Crée un lien de base de données. Tous les utilisateurs dans la base de données peuvent utiliser le lien pour accéder aux objets de base de données dans la base de données distante correspondante.

- **Lien globale :** Crée un lien à l'échelle du réseau. Quand un réseau Oracle utilise un serveur d'annuaire, le serveur d'annuaire crée et gère automatiquement les liens de base de données globale comme les noms de service nets, pour chaque base de données Oracle dans le réseau. Les utilisateurs dans une base de données peuvent utiliser un lien pour accéder aux objets de la base de données distante. [28]

II.3.4. Les types des liens par rapport aux utilisateurs :** [28]

Une différence principale entre les liens de base de données est la façon dont les connexions à une base de données distante se produit. Les utilisateurs accèdent à une base de données à distance via les types de liens utilisateurs suivants :

Type de liens utilisateurs	Description
Lien utilisateur connecté	Les utilisateurs se connectent comme eux-mêmes, ce qui signifie qu'ils doivent avoir un compte sur la base de données distante avec le même nom d'utilisateur et mot de passe de leur compte sur la base de données locale.
Lien utilisateur fixe	Les utilisateurs se connectent en utilisant le nom d'utilisateur et mot de passe référencé dans le lien. Par exemple, si Jane utilise un lien utilisateur fixe qui se connecte à la base de données A de Eric avec son nom d'utilisateur et mot de passe Eric , puis elle se connecte comme Eric, Jane a tous les privilèges accordés dans A à Scott directement, et tous les rôles par défaut que Eric a été accordée dans la base de données A.
Lien utilisateur actuel	Un utilisateur local peut se connecter en tant qu'utilisateur global, sans enregistrer le mot de passe utilisateur global dans une définition de lien. Par exemple, Jane peut accéder à une procédure qu'Eric a écrit, il a l'accès au compte d'Eric et le schéma de Eric sur la base de données A.

Figure II.4 : Types des liens par rapport aux utilisateurs

II.3.5. Les caractéristiques des liens : [27]

Déterminer le type de liens de base de données à employer dans une base de données distribuée dépend sur les exigences spécifiques des applications à l'aide du système. On tient compte alors de ces caractéristiques lors de votre choix :

Type de lien	Caractéristiques
Lien de la base de données privée	Ce lien est plus sécurisé qu'un lien public ou global, car seul le propriétaire du lien privé, ou des sous programmes dans le même schéma, peut utiliser le lien pour accéder à la base de données distante.
Lien de la base de données publique	On peut créer un seul lien de base de données publique pour tous les utilisateurs dans une base de données pour accéder à la base distante.
Lien de base de données global	Lorsqu'un réseau Oracle utilise un serveur d'annuaire, l'administrateur peut facilement gérer les liens de base de données mondiaux pour toutes les bases de données du système. La gestion des liens de base de données est Centralisé et simple.

Figure II.4 : Les types des liens

II.3.6. Lien de base de données partagé : [28]

Une liaison de base de données partagée est un lien entre une base de données locale et la base de données distante. Le lien est partagé car plusieurs processus clients peuvent utiliser le même lien simultanément. Ces liens partagés diffèrent à partir de liens de base de données standards car les différents utilisateurs qui accèdent au même objet de schéma par un lien de base de données peuvent partager une connexion réseau.

Lorsqu'une base de données locale est connectée à une base de données à distance via une liaison de base de données, soit la base de données peut fonctionner en mode serveur dédié ou partagé. Le tableau suivant illustre ses possibilités :

Mode de base de données locale	Mode de base de données distante
Dedicated (serveur dédié)	Dedicated (serveur dédié)
Dedicated (serveur dédié)	Shared server (serveur partagé)
Shared server (serveur partagé)	Dedicated (serveur dédié)
Shared server (serveur partagé)	Shared server (serveur partagé)

Figure II.5 : Les différents modes de serveurs de base de données

II.3.7. Création des liens :

Un DBlink est un objet Oracle permettant de créer un lien entre plusieurs bases de données Oracle, ce lien peut être sur le même hôte, sur un hôte appartenant au domaine ou sur tout autre hôte joignable par le protocole TCP/IP ;

Pour créer un DBLink, il faut que le **tnsname.ora**[voir annexe] soit correctement renseigné concernant le **SID** de la base distante, ci-dessous la syntaxe pour créer les liens :

« CREATE [PUBLIC] DATABASE LINK <nom de lien de base de données> [CONNECT TO <utilisateur oracle> IDENTIFIED BY <mot de passe utilisateur oracle distant>] USING '<chaine de base de données>' ;» [29]

Où les significations des mots clés sont présentées dans le tableau suivant :

Mot	Signification
CONNECT TO	Il permet d'accéder à la base distante avec un nom d'utilisateur différent de celui en cours dans la session de la base locale.
<nom de lien de base de données>	Il correspond au nom de la base de données vers laquelle le DB Link pointe
<chaîne de base de données>	Est une chaîne de connexion SQL*NET valide (par exemple, sous Oracle, trouvée dans le fichier tnsnames.ora).

Figure II.6 :Tableau significatif du mot clé utilisé dans la syntaxe

Exemple1 :

On a une base de donnée automatique locale, on veut accéder à une base distante informatique avec un nom utilisateur : user1 et un mot de passe : user1, on créera le lien privé comme suit :

“CREATE DATABASE LINK lien2 CONNECT TO user1 IDENTIFIED BY user1 USING 'XE';”

Exemple2 :

Création de lien publique pour l’utilisateur connecté :

« CREATE PUBLIC DATABASE LINK lien1; »

Exemple3:

Création de lien publique pour l’utilisateur global :

« CREATE PUBLIC DATABASE LINK lien2 CONNECT TO CURRENT_USER USING base;

II.3.9. Utilisation de lien après création :

Après création de lien, il sera possible de faire la requête sur un ou plusieurs objets de la base, il suffit d’avoir des droits d’accès pour se connecter à la base distante, pour interroger une base de données, on doit ajouter ‘@’ et le nom du lien :

SELECT * FROM « nom de l’objet » @ « nom de lien » ;

Par exemple pour pouvoir accéder à l’objet etudiant_auto de la base informatique vers automatique on utilise le lien1 :

SELECT * FROM etudiant_auto @lien1;

II.4. Conclusion :

Dans ce chapitre nous avons abordé en premier les patrons en générale ainsi les patrons de conception qui sont une solution standard pour la conception de logiciel, ensuite les liens de base de données qui permettent la communication entre des bases distantes, En effet ces deux notions offrent un moyen d'intégration de base de données.

Dans le chapitre suivant on s'intéressera à l'analyse et la conception de notre cas d'étude, dont nous appliquerons quelques types de patrons et les liens de base de données crées.

III.1. Introduction :

Après avoir élaboré les différents concepts nécessaires à la réalisation de notre objectif de faire intégrer les différents sous-systèmes pour pouvoir satisfaire les besoins des utilisateurs de notre application nous passons à la phase analyse et conception, cette phase est précédée par une analyse profonde et bien réfléchie car elle est le reflet du futur système avant même sa concrétisation suivi d'une conception détaillée.

Pour cela l'objectif de ce chapitre est modéliser notre système intégré en utilisant UML qui nous permet de bien représenter les différents aspects de notre application et aussi les motifs de conception employé pour faire de l'intégration.

III.2. Modèle MVC (Model- Vue-Controler): [30]

Le patron de conception MVC est un patron qui architectural qui sépare les données

(Modèle), l'interface homme-machine (la vue) et la logique de contrôle (contrôleur).

Ce modèle de conception impose donc une séparation en trois couches :

- Le modèle : il représente les données de l'application. Il définit aussi l'interaction avec la base de données et le traitement de ces données.
- La vue : Elle représente l'interface utilisateur, elle n'effectue aucun traitement, elle se contente simplement d'afficher les données que lui fournit le modèle. Il peut tout à fait y avoir plusieurs vues qui présentent les données d'un même modèle.
- Le contrôleur : il gère l'interface entre le modèle et le client. Il va interpréter la requête de ce dernier pour lui envoyer la vue correspondante. Il effectue la synchronisation entre le modèle et les vues. Le contrôleur doit donc :
    - Analyser la requête, soit extraire les informations dans l'URL.
    - Faire une mise à jour du modèle si nécessaire, en lui passant des paramètres.
    - Déterminer la vue à afficher et demander son affichage.

La synchronisation entre la vue et le modèle se passe avec le patron Observer. Il permet de générer des événements lors d'une modification du modèle et d'indiquer à la vue qu'il faut se mettre à jour.

Voici un schéma des interactions entre les différentes couches :

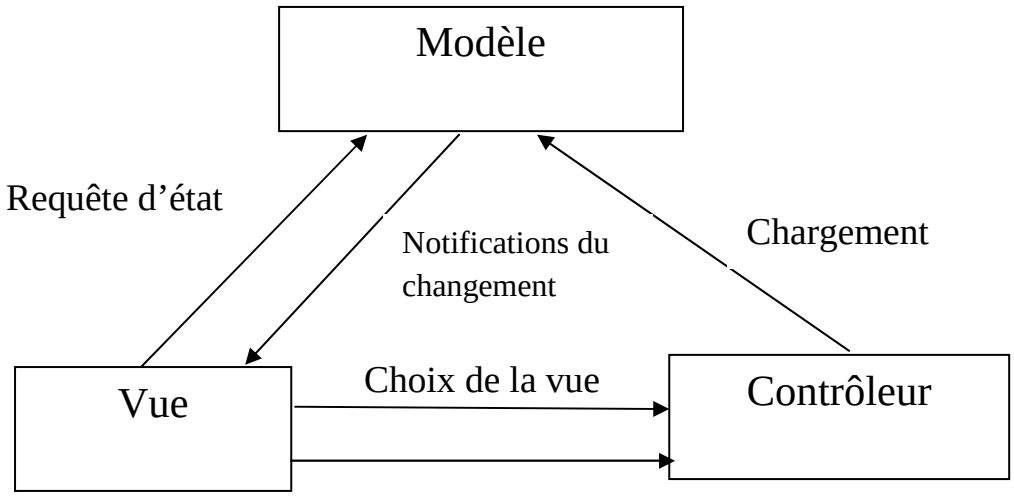


Figure III.1 : Architecture du modèle MVC

III.2.1. Développement sans respecter le modèle MVC :

Dans la plupart des architectures normalisées pour structurer une application, si vue modifié les données, toutes les vues concernées par la modification doivent être mise à jour, alors il y aura des inconvénients :

- Réutilisation réduite.
- Temps de développement augmente
- Coût de maintenance augmente
- Minimiser la possibilité d'extension
- Moindre lisibilité.

III.2.2.Développement en respectant le modèle MVC :

Un avantage apporté par ce modèle est la clarté de l'architecture qu'il impose. Cela simplifie la tâche du développeur qui tenterait d'effectuer une maintenance ou une amélioration sur le projet.

En effet, la modification des traitements ne change en rien la vue. Par exemple on peut passer d'une base de données de type SQL, à XML en changeant simplement les traitements d'interaction avec la base, et les vues ne s'en trouvent pas affectées. D'où :

- Possibilités de réutilisation
- Temps de développement diminue
- Facilité de maintenance
- La possibilité d'extension
- Plus de lisibilité

III.2.3.Avantages du MVC :

Le modèle MVC a plusieurs avantages parmi eux :

- Il peut y avoir plusieurs vues sur le même modèle
- Plusieurs contrôleurs peuvent modifier le même modèle
- Toutes les vues seront notifiées des modifications. [30]

III.3. Représentation de cas d'étude :

Considérons un système à développer qui est constitué de quatre bases de données hétérogènes pour la gestion de l'étudiant au sein de la faculté génie électrique et informatique. Comme le montre la figure suivante :

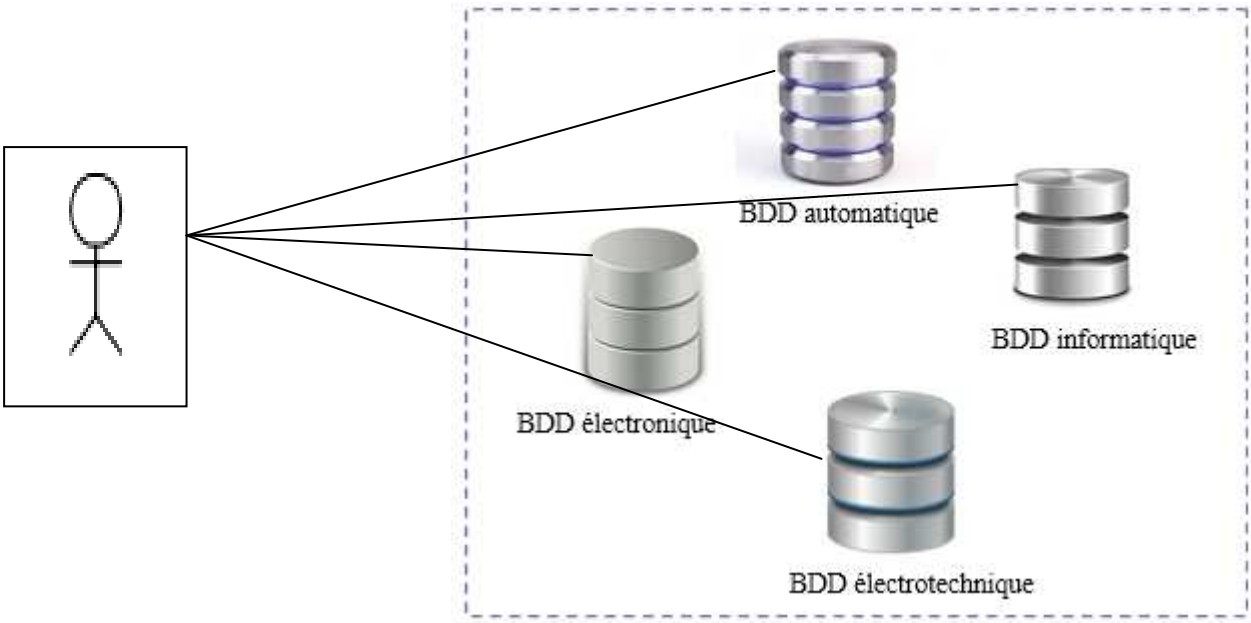


Figure III.2 : Faculté génie électrique et informatique

Les bases des données sont définies selon les tabes suivantes :

III.3. Représentation de cas d'étude :

Site 1 : base de données « informatique »

Schéma relationnel :

Etudiant_inf(id_etud, nom , prénom ,date_nais,lieu_nais ,adr_etud , id_spec*).

Spécialité_inf(id_spec ,opt, nom)

Module_inf(id _mod, coef , nom_mod , res_mod, id_exam*).

Examen_inf(id_exam , nom_exam,typ_exam,heur_exam ,dur_exam, date_exam).

Semestre(id_sem, nom_sem, anni_univ, id_spec*)

Note (id_note, note)

Avoir_inf (id _mod*, id_sem*)

Compose_inf(id_sem*, id_spec*)

Afficher_not_inf (id_exam*, id_note*)

Recup_not_inf (id_etud*, id_note*)

Site 2 : la base de données « automatique »

Schéma relationnel :

Etudiant_auto(num_inscr, nom, prénom, date_nais, lieu_nais,adr, id_form*)

Specialite_auto(id_form,opt, nom_form,)

Module_auto(id_modul, nom_modul, res_modul,id_exame*)

Examen_auto(id_exame**,**nom_exame,note_exame,typ_exame,dat_exame,heur_exame, dur_exame)

Semestre_ auto (id_semestr, nom_semestr, anni_univ, id_form*)

Note (id_note, note)

Avoir_auto (id_modul* ,id_semestr*)

Compose_auto(id_semestr*, id_form*)

Afficher_not_auto (id_exam*, id_note*)

Recup_not_auto (id_etud*, id_note*)

Site 3 : la base de données « électronique »

Schéma relationnel :

Etudiant_elec(cod_etud, nom , prénom ,date_nais,lieu_nais , adr_etud , cod_for*)

Specialite_elec(cod_for ,nom_for, option)

Module_elec(cod_mod, nom_mod ,res_mod ,coef,cod_exam*)

Examen_elec(cod_exam , nom_exam,typ_exam,dure_exam ,date_exam, heure_exam)

Semestre_elec (cod_sem, nom, anni_univ, cod_for*)

Note (id_not, note)

Avoir_elec (cod_for*,cod_sem*)

Compose_auto(cod_sem*, cod_for*)

Afficher_not_auto (cod_exam*, id_not*)

Recup_not_auto (cod_etud*, id_not*)

Site 4 : base de données « électrotechnique »

Schéma relationnel :

Etudiant_eth(id, nom, prénom, date_nais, lieu_nais, adr, cod_spec*)

Specialite_eth(cod_spec, opt , nom_spec,)

Module_eth(cod , coef, nom_mod, res_mod, cod_eprv*)

Examen_eth (cod_eprv**,** nom_eprv,dur_eprv, dat_eprv, heur_eprv, typ_eprv)

Semestre_ eth (cod_semestr,nom, anni_univ, cod_spec*)

Note (cod_note, note)

Avoir (cod_modul* ,code_sem*)

Compose_auto(cod_semestr*, cod_eprv*)

Afficher_not_auto (cod_eprv*, cod_note*)

Recup_not_auto (cod_estd*, cod_note*)

III.4. Le problème traité :

Le principal problème est comment développer un nouveau système capable de réaliser l'intégration entre les bases de données existantes (informatique, électronique, automatique, électrotechnique).

III.5. Solution proposé :

La solution est de permettre à l'utilisateur d'accéder à des données stockées dans les quatre sources de données (informatique, électronique, automatique et électrotechnique) à partir d'une seule interface.

III.6. Les diagrammes représentatifs de futur système :

Dans cette section nous allons présenter une description générale de la solution proposée ci auparavant par des éléments graphiques en utilisant le langage de modélisation UML ainsi que les différents patrons de conception mise en place pour obtenir une unicité et une homogénéité des données de notre système.

III.6.1. Diagramme de cas d'utilisation :

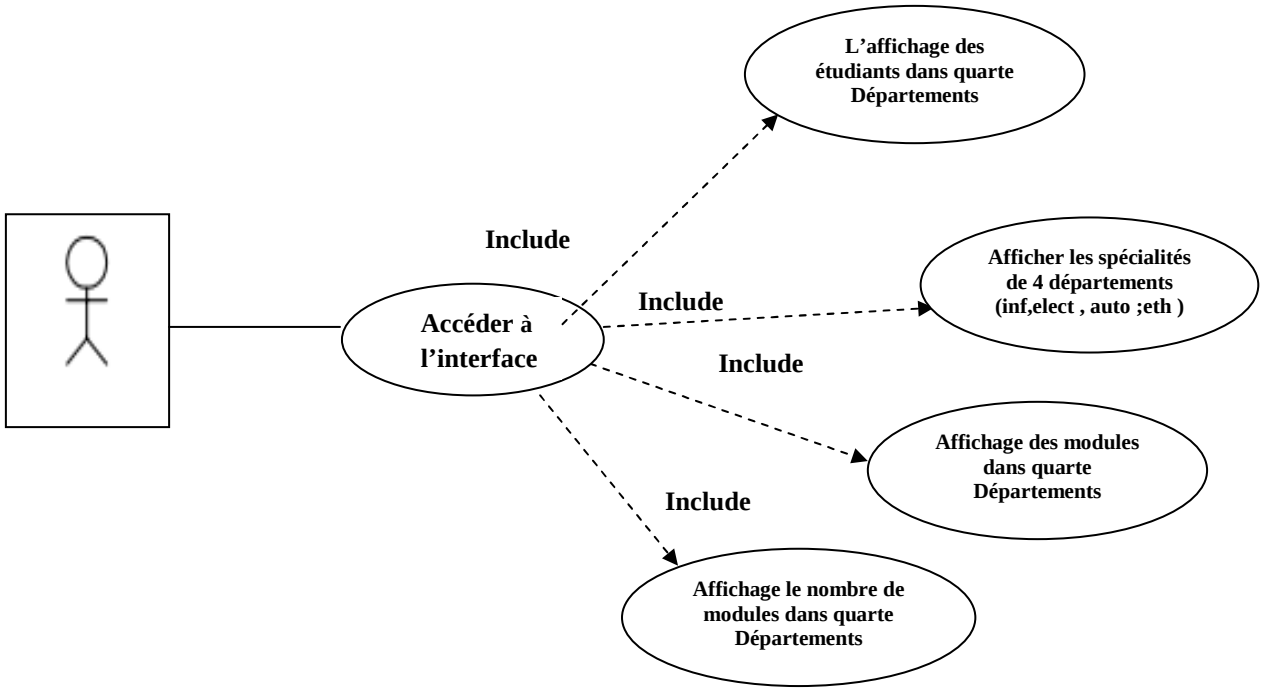


Figure III.3 : Diagramme de cas d'utilisation de futur système

III.6.2. Diagrammes de séquence :

III.6.2.1. Diagramme de séquence pour cas d'utilisation 1 « l'affichage des étudiants dans quartes départements » :

- **Requête 1 :**

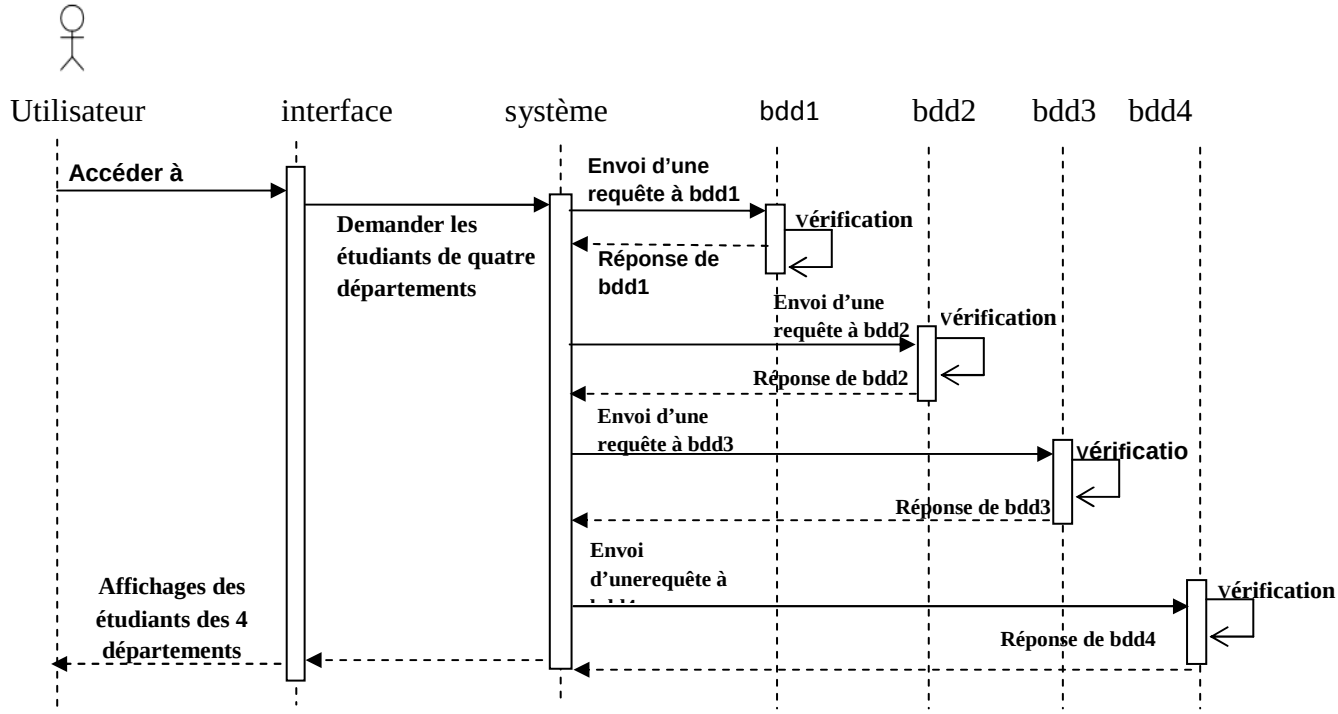
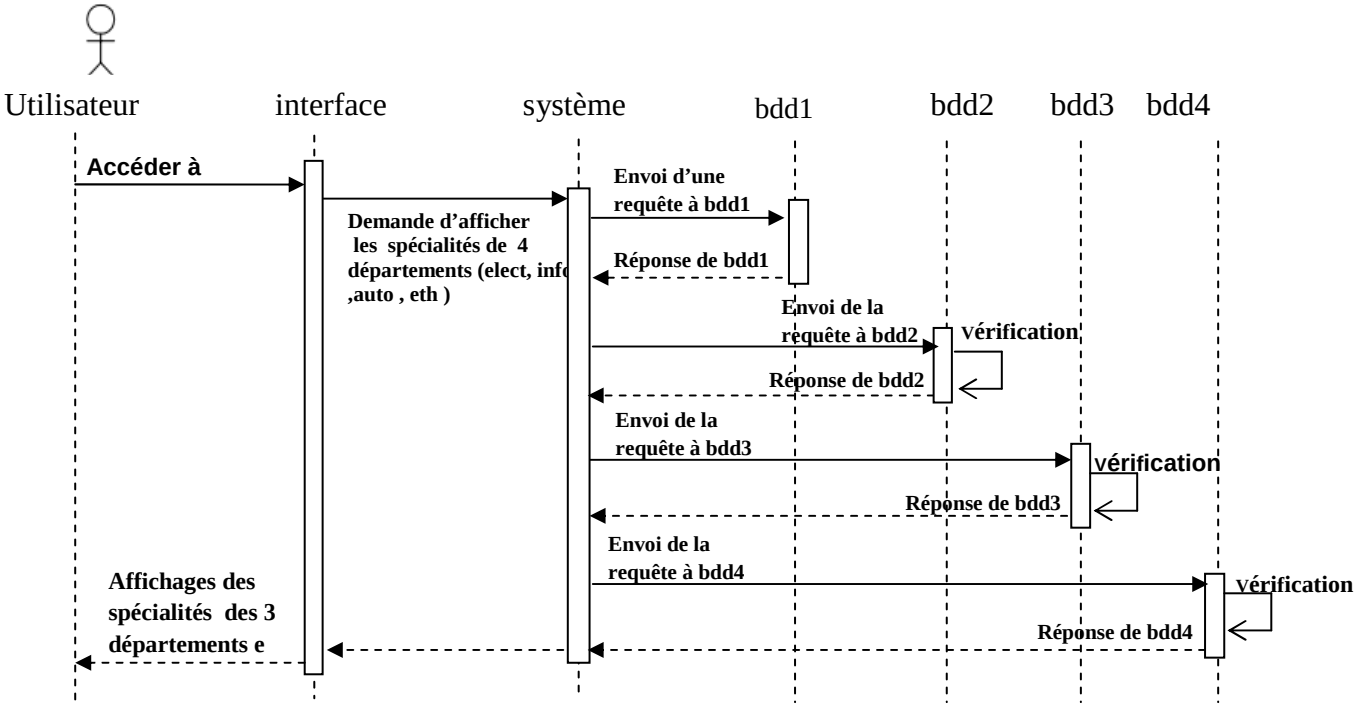


Figure III.4. Diagramme de séquence pour cas d'utilisation 1 « l'affichage des étudiants dans quartes départements » :

III.6.2.2. Diagramme de séquence pour cas d'utilisation 2 « Afficher les spécialités de quatre départements (inf, elect, auto, eth) »

- **Requête 2 :**



III.5. Diagramme de séquence pour cas d'utilisation 2 « Afficher les spécialités de quatre départements (inf, elect, auto, eth) »

III.6.2.3. Diagramme de séquence pour cas d'utilisation 3 « Afficher les module dans lesquatredépartements (info, elect, auto, eth) » :

- **Requête 3 :**

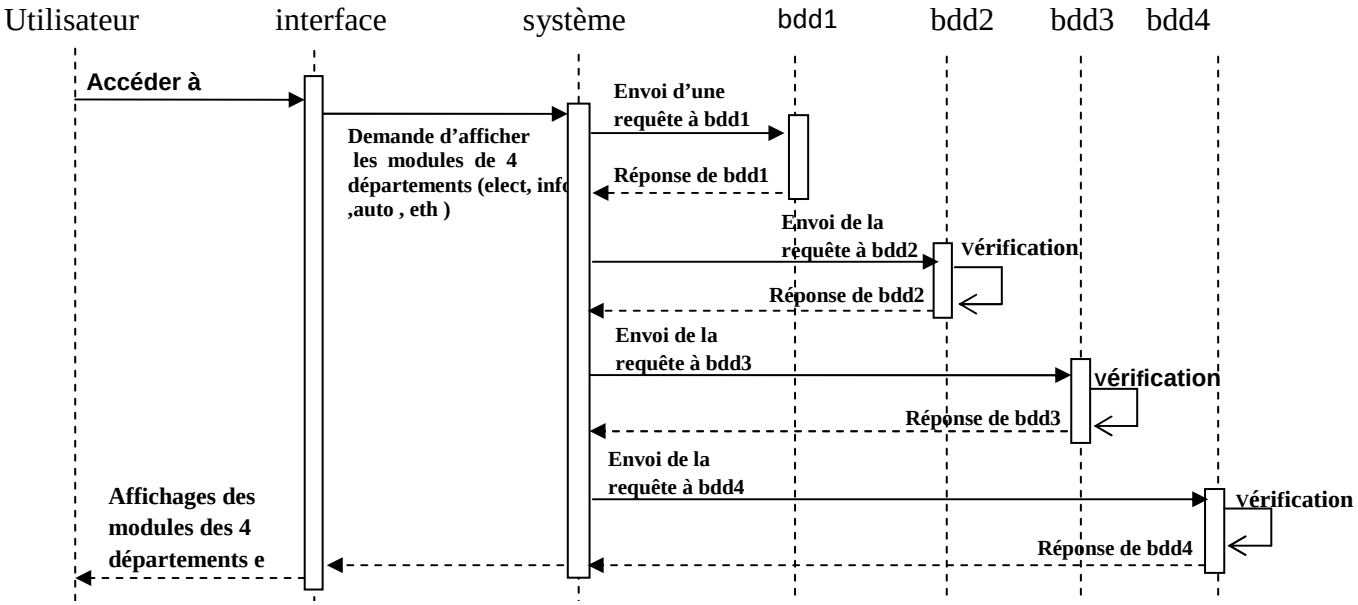


Figure III.6 : Diagramme de séquence pour cas d'utilisation 3 « Afficher les modules de quatre départements (inf, elect, auto, eth) »

III.6.2.4. Diagramme de séquence pour cas d'utilisation 4 « «affichager le nombre de module dans lesquatredépartements (info, elect, auto, eth) » :

- **Requête 4 :**

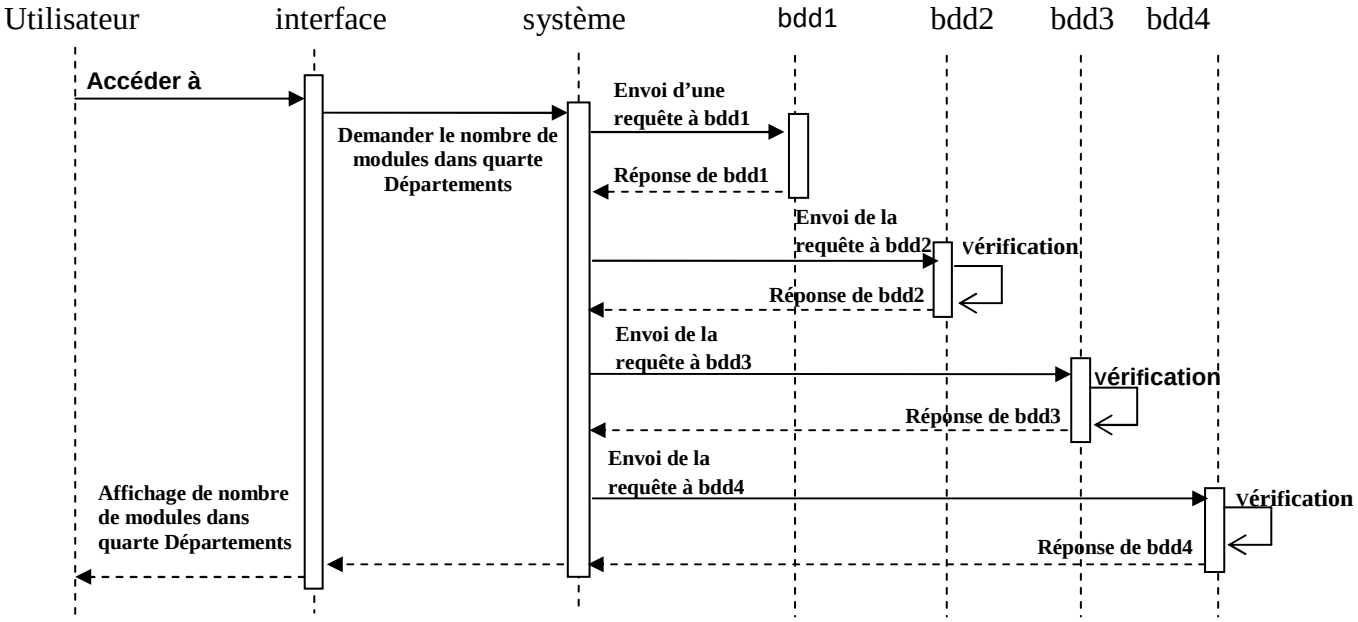


Figure III.7 : Diagramme de séquence pour cas d'utilisation 4« Afficher les nombre de modules dans quatre départements (inf, elect, auto, eth) »

III.7. Patrons de conception :

Pour résoudre les problèmes liés à l'intégration des quatre bases de données on a adopté le patron de conception façade et adaptateur et Template Method afin de pouvoir afficher les résultats souhaités à utilisateur. Dans ce qui suit on va éclairer ces trois patrons puis on les appliquera sur notre cas.

III.7.1. Patron façade :

Le patron de conception **façade** a pour but de cacher une conception et une interface ou un ensemble d'interfaces complexes difficiles à comprendre. La façade permet de simplifier cette complexité en fournissant une interface simple du sous-système. Habituellement, la façade est réalisée en réduisant les fonctionnalités de ce dernier mais en fournissant toutes les fonctions nécessaires à la plupart des utilisateurs.

III.7.1.1. Les avantages de l'utilisation de façade :

- Simplifier l'utilisation et la compréhension d'une bibliothèque logicielle car la façade possède des méthodes pratiques pour les tâches courantes,
- Rendre le code source plus lisible.
- Diminuer le couplage entres classes.
- Rassembler une collection d'API complexes en une unique et meilleure API (orientée tâches utilisateurs).
- **Exemple :**

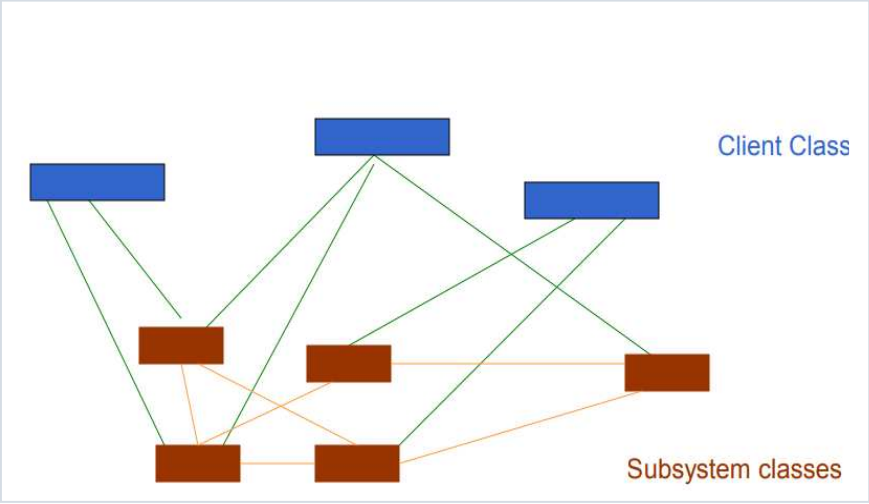


Figure III.8 : Système avant application de façade

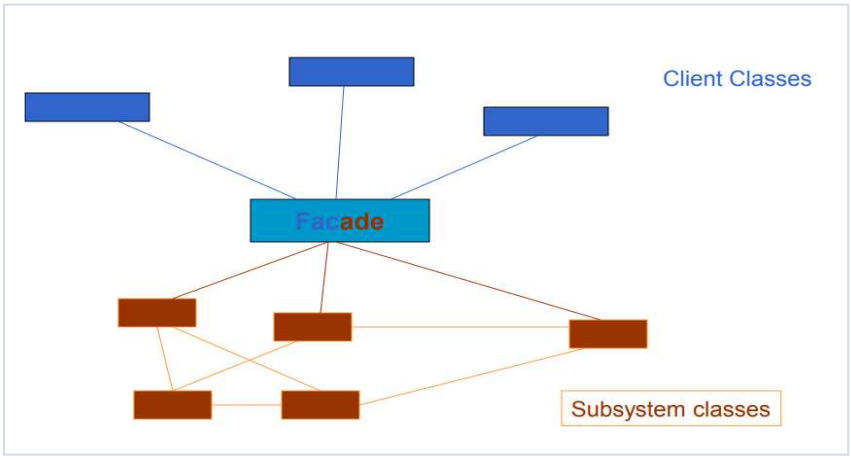


Figure III.9: Système après application de façade

- **L'utilisation de façade pour récupérer les étudiants inscrit dans les quatre départements**

L'utilisation de façade nous a permet de faciliter l'accès aux quarte tables (étudiant_info, étudiant_auto, étudiant_elect, étudiant_eth) en fournissant une interface « étudiant » qui regroupent tous les attributs utiles à étudiants. Comme le montre le diagramme suivant :

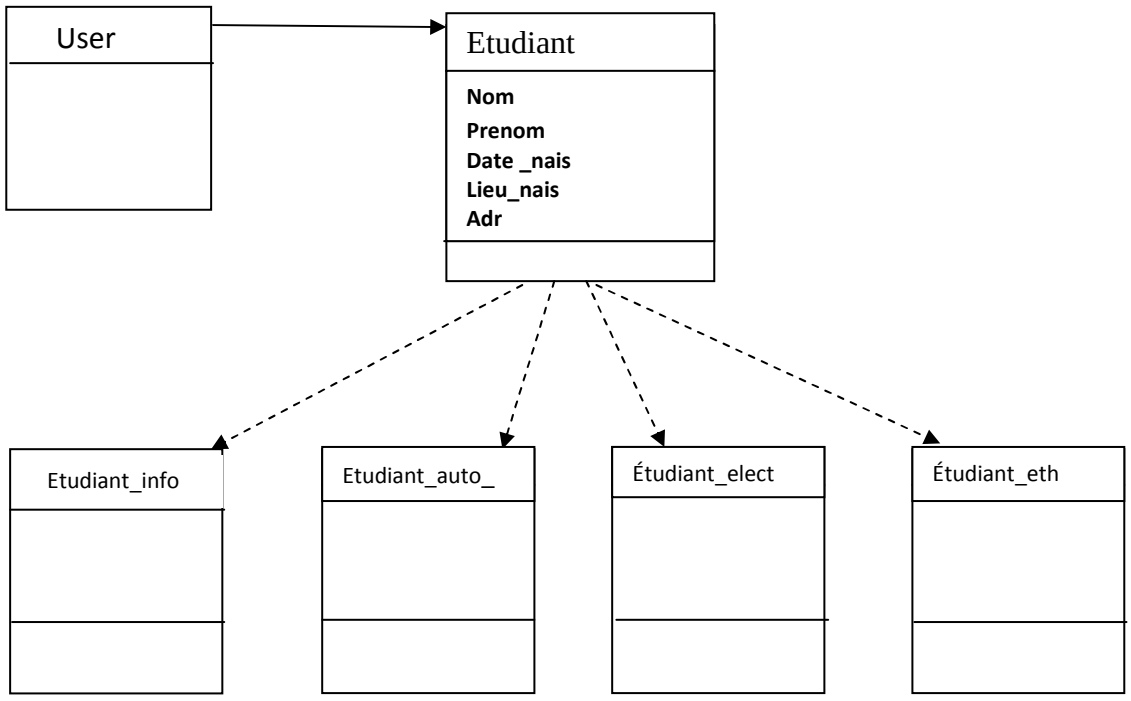


Figure III.10 : L'utilisation de façade pour récupérer les étudiants inscrit dans les quatre départements

III.7.2. Patron adaptateur :

Le patron de conception adaptateur permet de convertir l'interface en une autre classe interface que le client attend. Adaptateur fait fonctionner un ensemble des classes qui n'auraient pas pu fonctionner sans lui, à cause d'une incompabilité d'interface.

III.7.2.1. Les avantages de l'utilisation de l'adaptateur :

- Résoudre un problème d'incompatibilité d'interfaces (API) :

• Un client attend un objet dans un format donné.

• Les données sont encapsulées dans un objet qui possède une autre interface.

- **Utilisation de patron adaptateur pour afficher les spécialités de 3 départements (elect, auto, eth)**

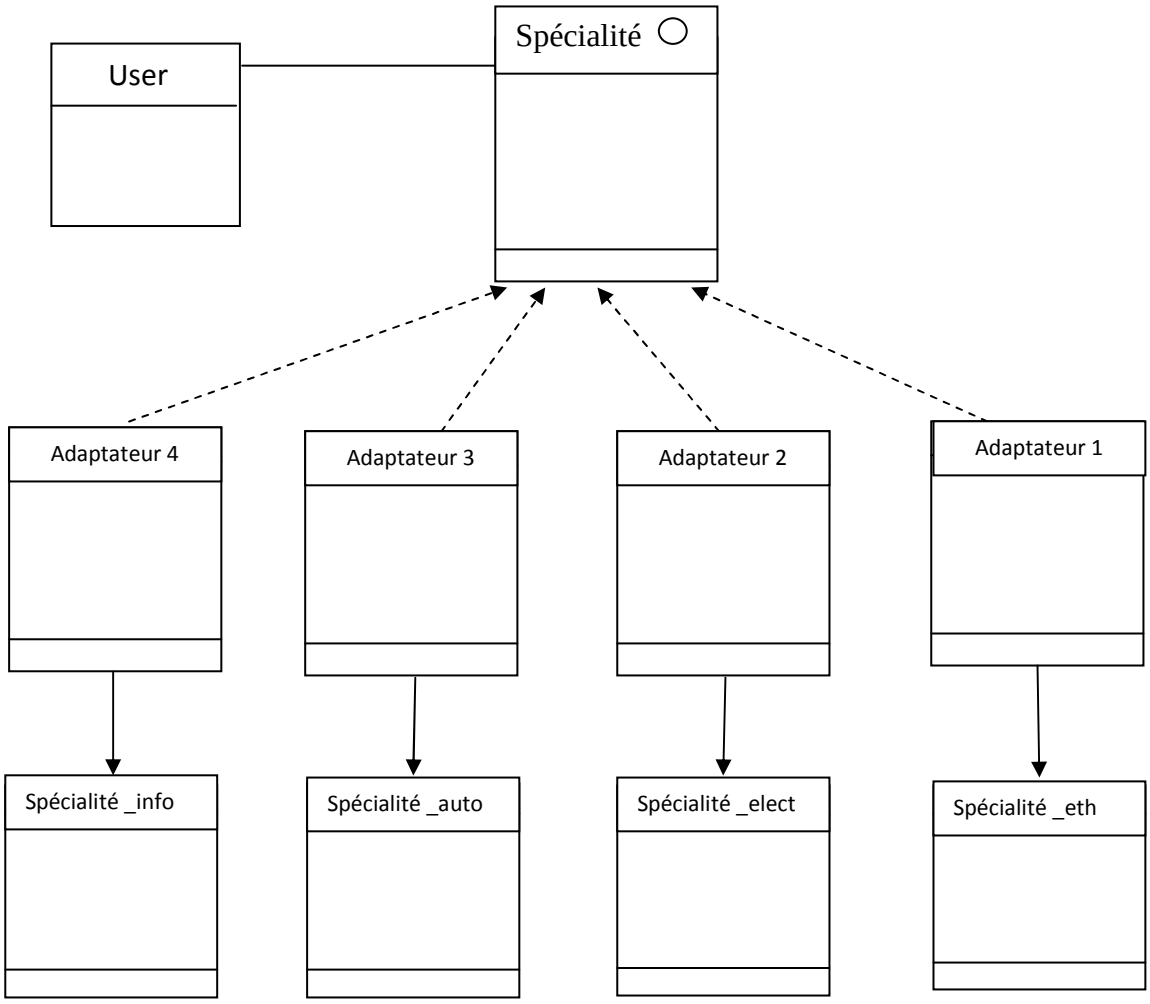


Figure III.11 :Utilisation de patron adaptateur Afficher les spécialités de quatredépartements (inf, elect, auto, eth)

III.7.3. Patron Template Method (patron de méthode) :

Le patron de méthode est un patron de conception comportemental qui :

- Découpe un algorithme en étapes et ce sont certaines étapes de l'algorithme qui peuvent varier.
- Un patron de méthode définit le squelette d'un algorithme à l'aide d'opérations abstraites dont le comportement concret se trouvera dans les sous-classes, qui implémenteront ces opérations
- Délégation de certaines étapes aux sous-classes.
- Patron de méthode permet aux sous-classes de redéfinir certaines étapes d'un algorithme sans modifier la structure de celui-ci.

III.7.3.1. Ces avantages :

- Fixer clairement des comportements standards qui devraient être partagés par toutes les sous-classes, même lorsque le détail des sous-opérations diffère ;
- Factoriser du code qui serait redondant s'il se trouvait répété dans chaque sous-classe.

III.7.3.2. Ces Spécificité :

La technique du patron de méthode a ceci de particulier que c'est la méthode de la classe parent qui appelle des opérations n'existant que dans les sous-classes. C'est une pratique courante dans les classes abstraites, alors que d'habitude dans une hiérarchie de classes concrètes c'est le contraire : ce sont plutôt les méthodes des sous-classes qui appellent les méthodes de la super-classe comme morceau de leur propre comportement.

- **Utilisation de patron Template Method pour trouver le nombre de spécialités dans chaque département :**

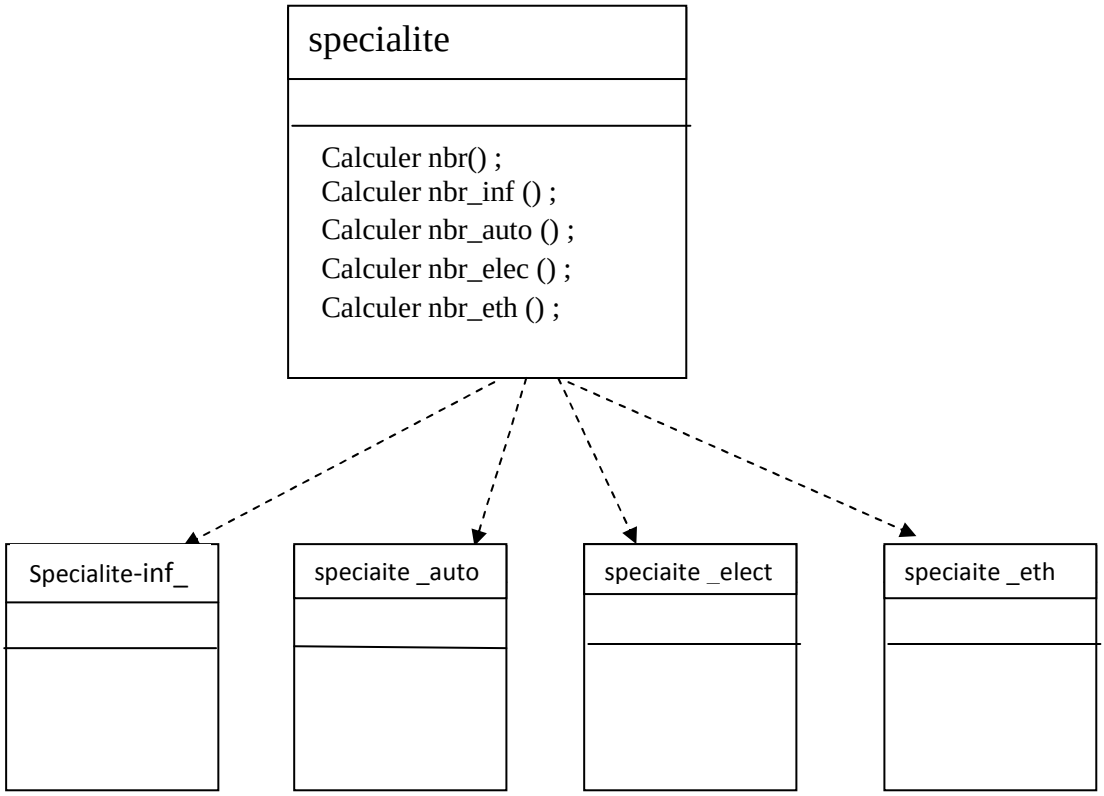


Figure III.12 : Utilisation de patron Template Method pour trouver le nombre de spécialités dans chaque département :

III.8. Conclusion :

A l'issue de ce chapitre on a présenté notre cas d'études en suite on a entouré l'objectif de notre futur système, pour les éteindre nous sommes basées sur quelques patrons de conceptions qui répondent à nos besoins, aussi sur le langage UML pour la modélisation graphique de notre futur système, le chapitre suivant sera consacré à la réalisation de notre projet.

IV.1. Introduction :

On s'intéressera dans ce chapitre à l'implémentation des patrons de conception qu'on a déjà cité dans le chapitre précèdent, et à la réalisation des liens entres les quatre bases de données, au premier lieu nous présentons les outils de développement et les langages de programmation avec lesquels nous avons travaillé, ensuite nous exposons des captures d'écrans de quelques requêtes testées sous NetBeans.

IV.2. Outils de de développement :

IV.2.1. Oracle 10g : [31]

Oracle est un système de gestion de base de données édité par la société d'Oracle Corporation, leader mondial de base de données.

La société *Oracle Corporation* a été créé en 1977 par Lawrence Ellison, Bob Miner, et Ed Oates. En 1979 l'entreprise change son nom en devenant *Relational Software Incorporated(RSI),* et commercialise un système de gestion de base de données relationnelles (*SGBDR* ou *RDBMS pour Relational Database Management System)* nommé *Oracle*.

Oracle est écrit en langage C et est disponible sur de nombreuses plates-formes dont AIX (IBM), Solaris (SUN), HP/UX (Hewlett Packard), Windows NT (Microsoft).

Oracle 10g est la première version qui supporte les expressions rationnelles. (Le *g* signifie *grid*), un des atouts marketing de la 10g est en effet qu'elle supporte le « *grid computing* »que ce soit sur des serveurs Windows, Linux ou UNIX, Oracle Databases 10g pulvérise tous les records de performances et l'évolutivité. Autre caractéristiques clé, la base de données Oracle permet de migrer d'un serveur unique vers le Grid Computing sans avoir à modifier une seule ligne de code.

Oracle Database 10g permet de meilleurs résultats grâce notamment à l'automatisation des tâches administratives et à des fonctionnalités de sécurité et de conformité réglementaire sans équivalent sur le marché. Avec *Real Application Clusters* (RAC), l'option ajoutée pour le logiciel de base de données Oracle produit par *Oracle Corporation,* il fournit un logiciel pour le clustering et la haute disponibilité dans les environnements de base de données Oracle . Enfin avec une large gamme de version et des coûts d'exploitation réduits, la

solution phare d'oracle s'impose comme le choix privilégié de toutes les entreprises et ce que soit leur taille.

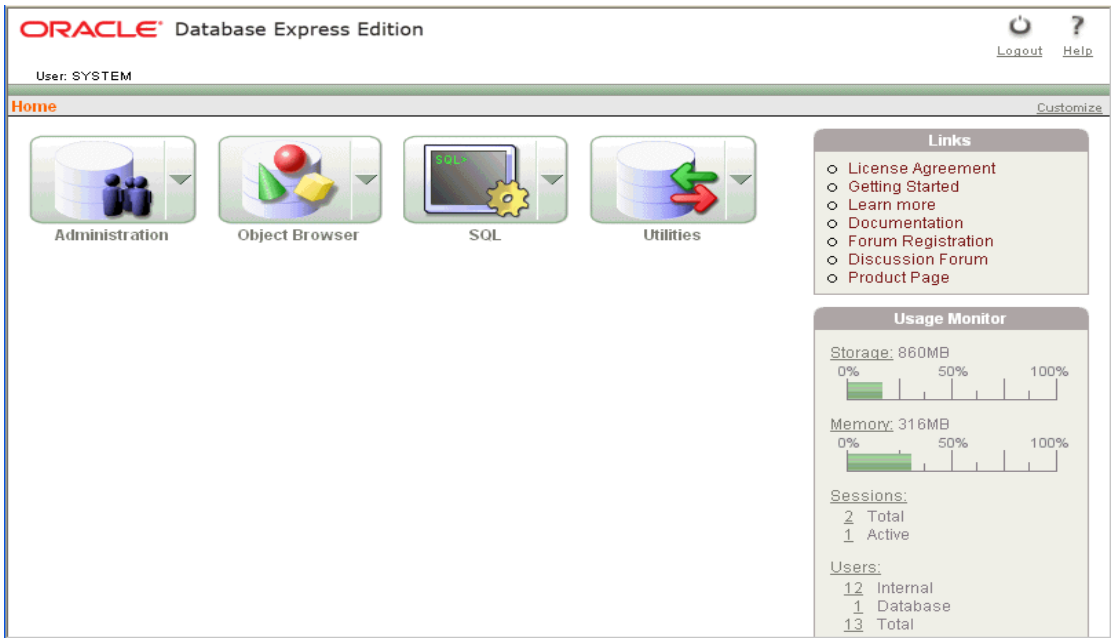


Figure IV.1 : Page d'accueil d'Oracle 10g

IV.2.1.1. Les fonctionnalités d'Oracle 10 g :

Oracle 10g permet d'assurer plusieurs nouvelles fonctionnalités dont on cite :

- La gestion des accès concurrents.
- Gestion des serveurs
- Performance and Scalability
- Grid Computing
- Intégration des informations
- Security and Directory
- Business Intelligence
- Gestion des contenus
- Services localisés
- Développement d'Applications.

IV.2.2.2. Les composants d'Oracle :

Outre la base de données, la solution Oracle est un véritable environnement de travail constitué de nombreux logiciel permettant notamment une administration graphique d'Oracle,

de s'interfacer avec des produits divers et d'assistants de création de bases de données et de configuration de celles-ci ;

On peut classer les outils d'Oracle selon diverses catégories :

- Les outils d'administration
- Les outils de développements
- Les outils de communication
- Les outils de génie logiciel
- Les outils d'aide à la décision

IV.2.2.3 Outils de développement d'Oracle :

Oracle propose également de nombreux outils de développement permettant d'automatiser la création d'application s'interfaçant avec la base de données. Ces outils de développement sont :

- Oracle Designer : Oracle Designer est l'outil case d'Oracle pour concevoir un système d'information et le générer. Après avoir généré le système d'information, on peut éditer le code généré avec Oracle Developer.
- SQL*Plus : qui est une interface permettant d'envoyer des requêtes SQL et PL/SQL à la base de données. SQL*Plus permet notamment de paramétrer l'environnement de travail (formatage des résultats, longueur d'une ligne, nombre de ligne par page, …)
- Oracle Développer : Il s'agit d'une suite de produits destinés à la conception et à la création d'application client-serveur. Il est composé de quatre applications :
  - Oracle Forms (anciennement SQL*Forms) : Un outil permettant de réaliser des états.
  - Oracle Reports (SQL*ReportWriter) : Un outil permettant de réaliser des états.
  - Oracle Graphics : Un outil de génération automatique de graphiques dynamiques pour présenter graphiquement des statistiques réalisées à partir des données de la base.
  - Procédure Builder : Un outil permettant de développer des procédures, des fonctions et des packages.

IV.2.2.4. Architecture du SGBD Oracle :

Une base de données Oracle est constituée de plusieurs éléments :

- Des processus chargés en mémoire sur le serveur
- Des fichiers physiques stockés sur le serveur
- D'un espace mémoire sur le serveur appelé SGA (System Global Area).

IV.2.2. PL/SQL Développer :

Le langage PL/SQL est un langage L4G (Langage quatrième génération), fournissant une interface procédurale au SGBD Oracle. Le langage PL/SQL intègre parfaitement le langage SQL en lui apportant une dimension procédurale.

En effet, le langage SQL est un langage déclaratif non procédurale permettant d'exprimer des requêtes dans un langage relativement simple. En contrepartie il n'intègre aucune structure de contrôle permettant par exemple d'exécuter une boucle itérative.

Ainsi le langage PL/SQL permet de manipuler de façon complexe les données contenues dans une base de données Oracle en transmettant un bloc de programmation au SGBD au lieu d'envoyer une requête SQL. De cette façon les traitements sont directement réalisés par le système de gestion de base de données, Cela a pour effet notamment de réduire le nombre d'échanges à travers le réseau des procédures écrites dans un autre langage (de troisième génération, généralement le langage C).

Le langage PL/SQL permet de définir un ensemble de commandes contenus ans ce que l'on appelle un « bloc » PL/SQL, ce dernier peut lui-même contenir des sous-blocs.

IV.2.3. NetBeans IDE 8.0.2 :

NetBeans est un environnement de développement intégré (IDE) pour développer principalement avec Java, mais aussi avec d'autres langages, en particulier PHP, C/C++ et HTML5. Il offre toutes les facilités d'un IDE moderne (éditeur en couleurs, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web). Il est aussi une plate-forme que demande Framework pour les applications Java Desktop et d'autres. L'IDE NetBeans est écrit en Java et peut fonctionner sur Windows, Linux, Solaris et d'autres plateformes supportant un compatible JVM (Java Virtuel Machine) ;

Le développement d'applications sur la base de la plate-forme NetBeans consiste à la réalisation de « modules » qui s'insèrent dans la plate-forme et en étendent dynamiquement les fonctions ;

NetBeans comprend un explorateur de bases de données qui supporte toutes les bases de données relationnelles pour lesquelles un connecteur JDBC existe (selon les versions des gestionnaires de bases de données): JavaDB (Derby) MySQL, PostgreSQL, Oracle, Microsoft SQL Server, IBM Redistribuable DB2, …

NetBeans propose différents outils pour l'exploitation de web services. Il supporte JAX-WS services, standards JAX-RPC Web Service, SOAP et RESTful Web Services, JBI Java Business Intégration, Java Architecture for XML Binding API (JAXB), Mobile Java ME Web services. Il permet l'utilisation des web services Google Maps, StrikeIron, Yahoo News Search. Il supporte par ailleurs l'intégration de services fournis par quelques acteurs clés (Google, Facebook, Yahoo, YouTube, …).

La plate-forme offre des services réutilisables communs aux applications de bureau, permettant aux développeurs de se concentrer sur la logique spécifique à leur application, Parmi les fonctionnalités :

- Gestion de l'interface utilisateur (par exemple, les menus et barres d'outils)
- Gestion des paramètres des utilisateurs
- Gestion du stockage (sauvegarde et le chargement tout type de données)
- Gestion de la fenêtre
- NetBeans Bibliothèque visuelle
- Outils de développement intégrés.

La figure suivante montre la page d'accueil de NetBeans IDE 8.0.2 :

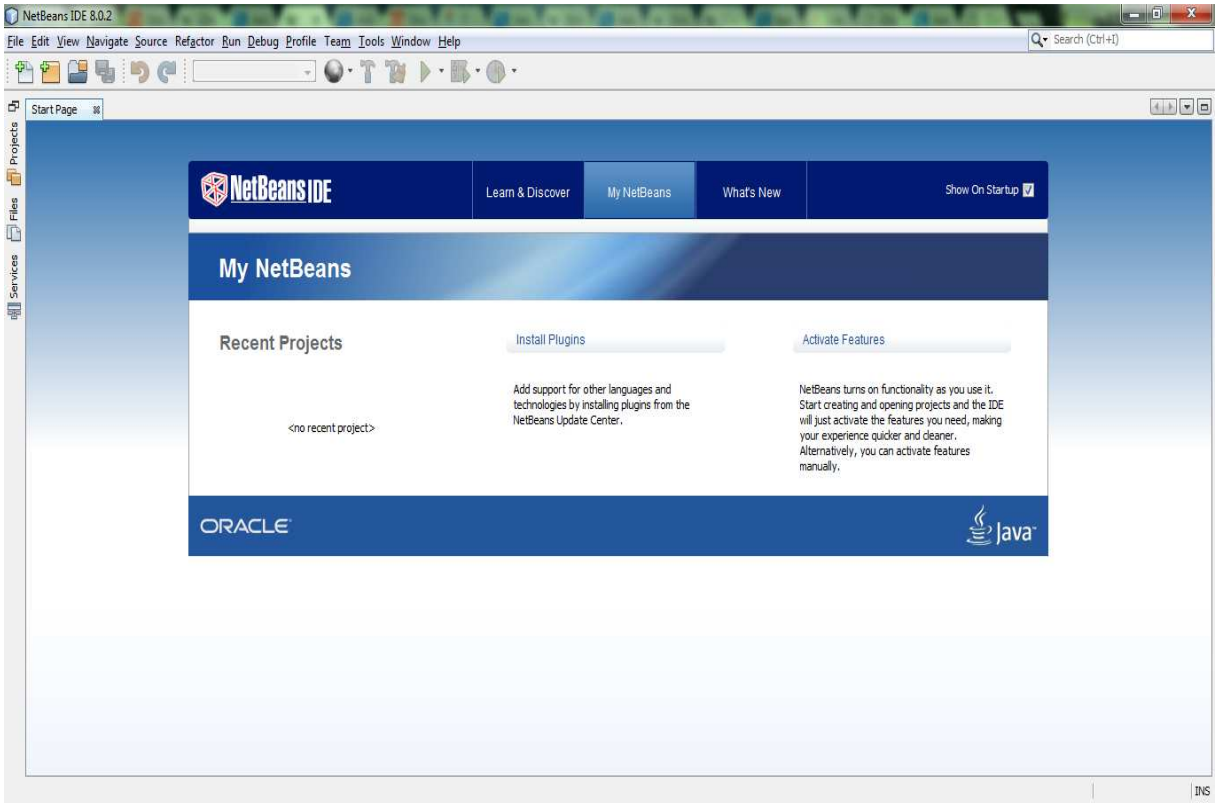


Figure IV.2 : Page d'accueil de NetBeans IDE 8.0.2

IV.3. Réalisation :

Nous présentons dans cette partie les liens de base de données créés, et quelques captures d'écran montrant les réponses des requêtes testées.

IV.3.1. Les liens de base de données :

Le schéma suivant montre tous les liens de base de données utilisés entre les quatre bases :

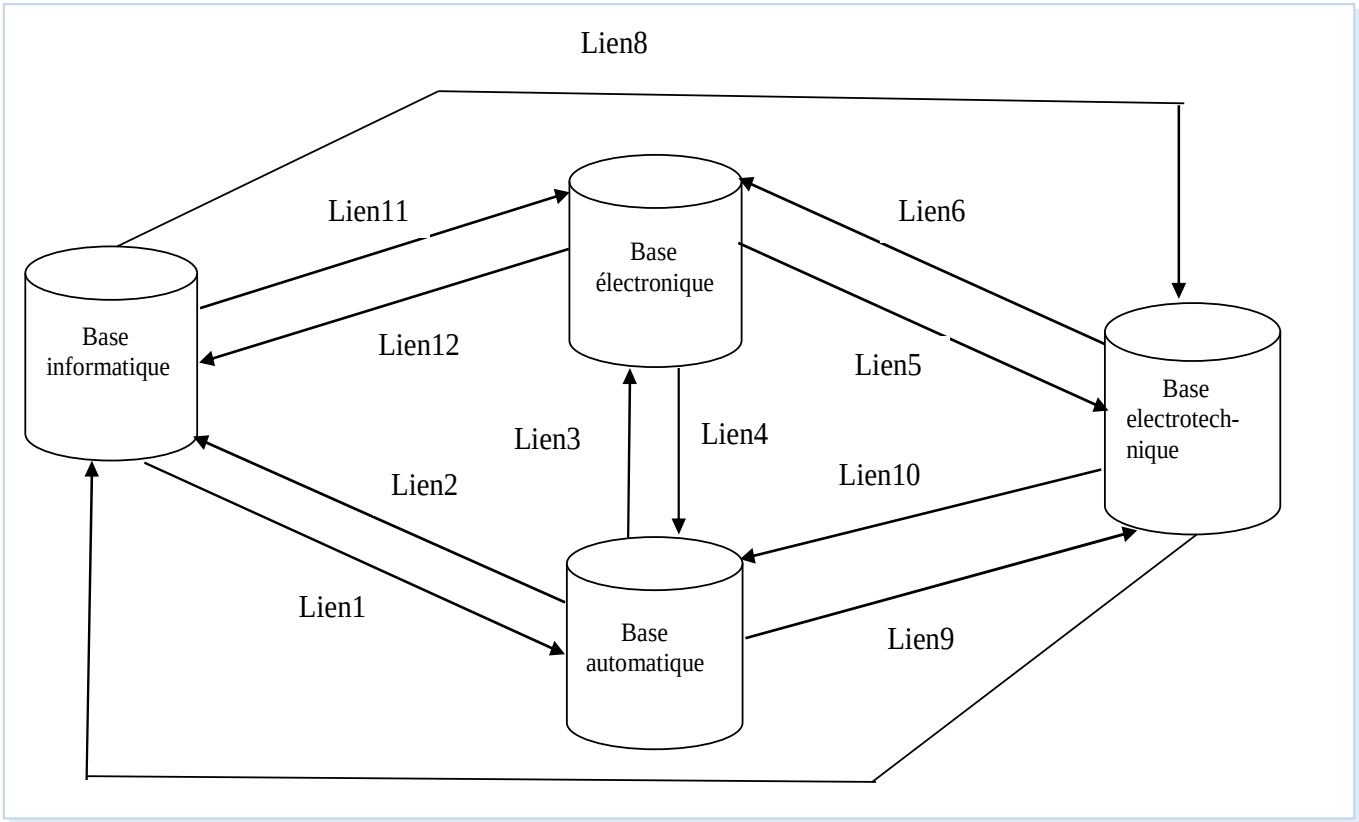


Figure IV.3 : Schéma des liens de bases de données effectués

Les liens montés ci-dessus permettent l'intégration de données hétérogènes, avec ces liens on peut récupérer des données dans les bases en faisant une connexion à une seule base seulement, exemple : avec les liens 2.3 et 9 on accède à toutes les données à partir de la base automatique.

IV.3.2. Les fenêtres de réponses des requêtes :

Les captures suivantes montrent les résultats d'utilisations des liens et les patrons de conceptions façade et adaptateur :

IV.3.2.1. Patron façade :

L'utilisateur émet la requête « select * from etudiant », et la classe etudiant qui est la façade fait tout le traitement et permet de fournir une réponse globale en utilisant les liens 1,8 et 9 comme c'est montré dans la figure suivant :

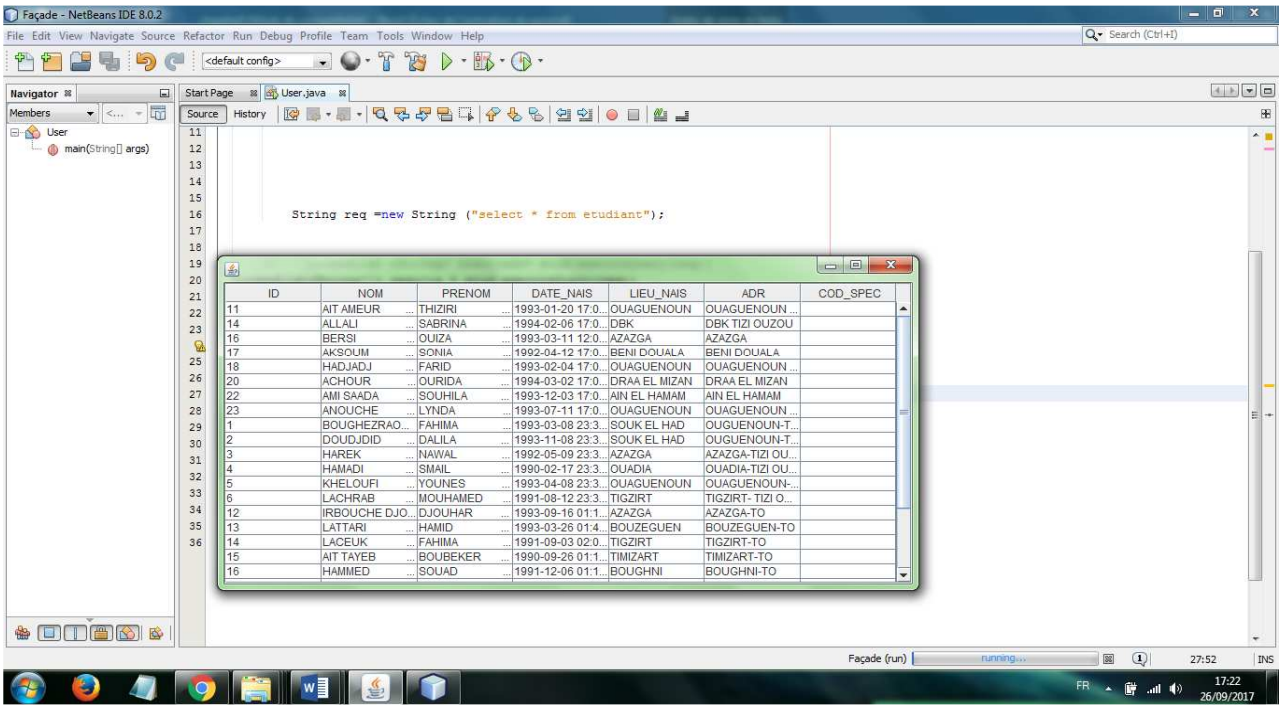


Figure IV.4 : Résultat d'application de patron façade pour récupérer la liste des étudiants

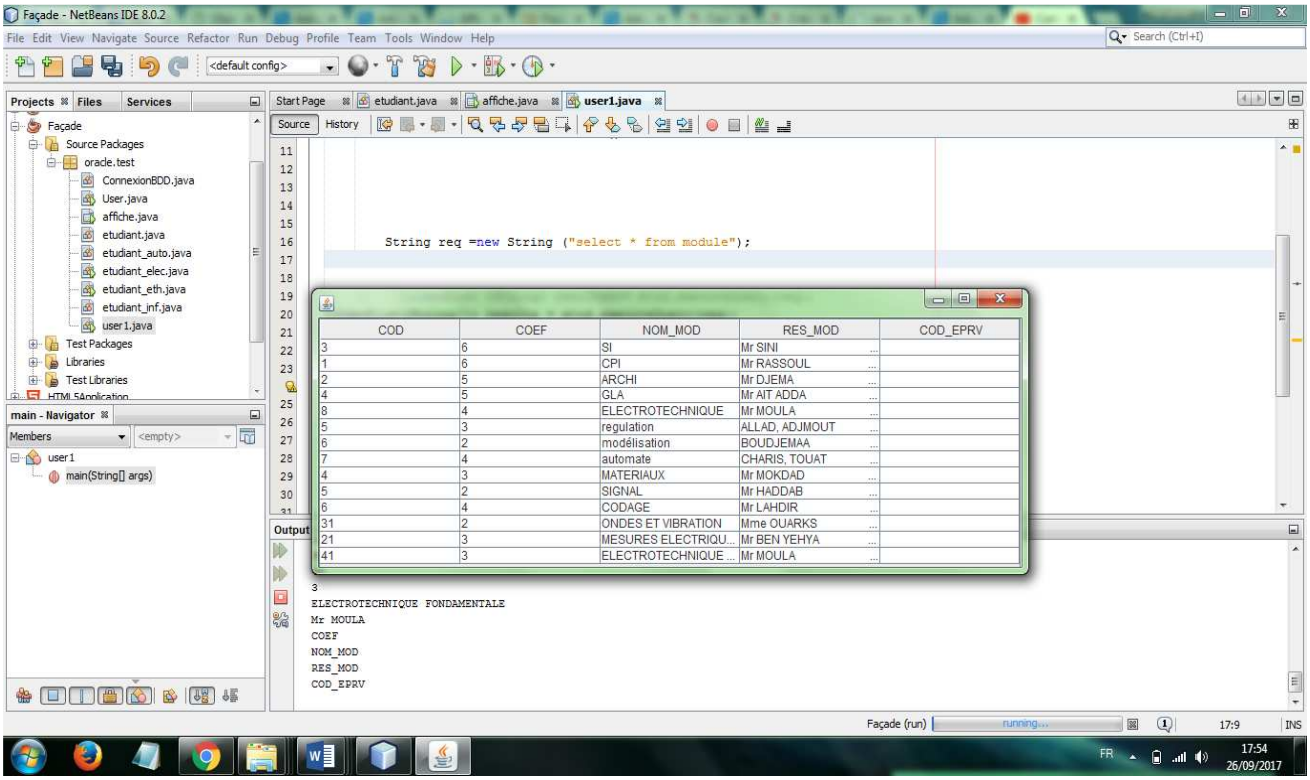


Figure IV.5 : Résultat d'application de patron façade pour récupérer la liste des modules

IV.3.2.2. Patron adaptateur :

L'utilisateur émet la requête « select * from specialite », et la classe specialite Permet de faire collaborer les quatre classes qui n'auraient pas pu se faire à cause de l'incompatibilité de leurs attributs, la classe specialite permet de fournir une réponse globale en utilisant les liens 1,8 et 11 comme c'est montré dans la figure suivant :

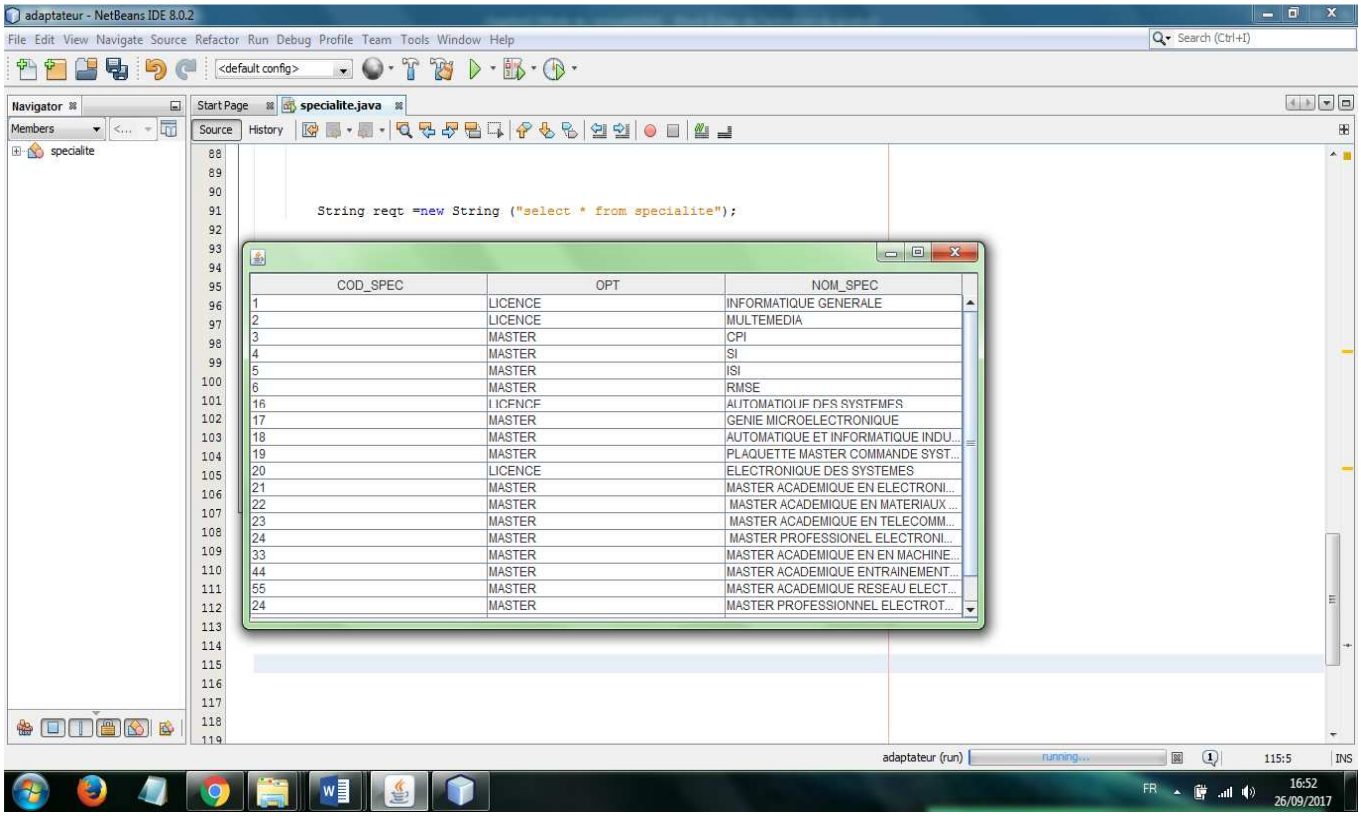


Figure IV.6 : Résultat d'application de patron adaptateur pour récupérer la liste des Spécialités

IV.3.2.3. Patron Template Method :

L'utilisateur émet la requête « select count (*) from module », et la classe module interroge les quatre classes et fournit une réponse globale en utilisant les liens 1,8 et 11 comme c'est montré dans la figure suivant :

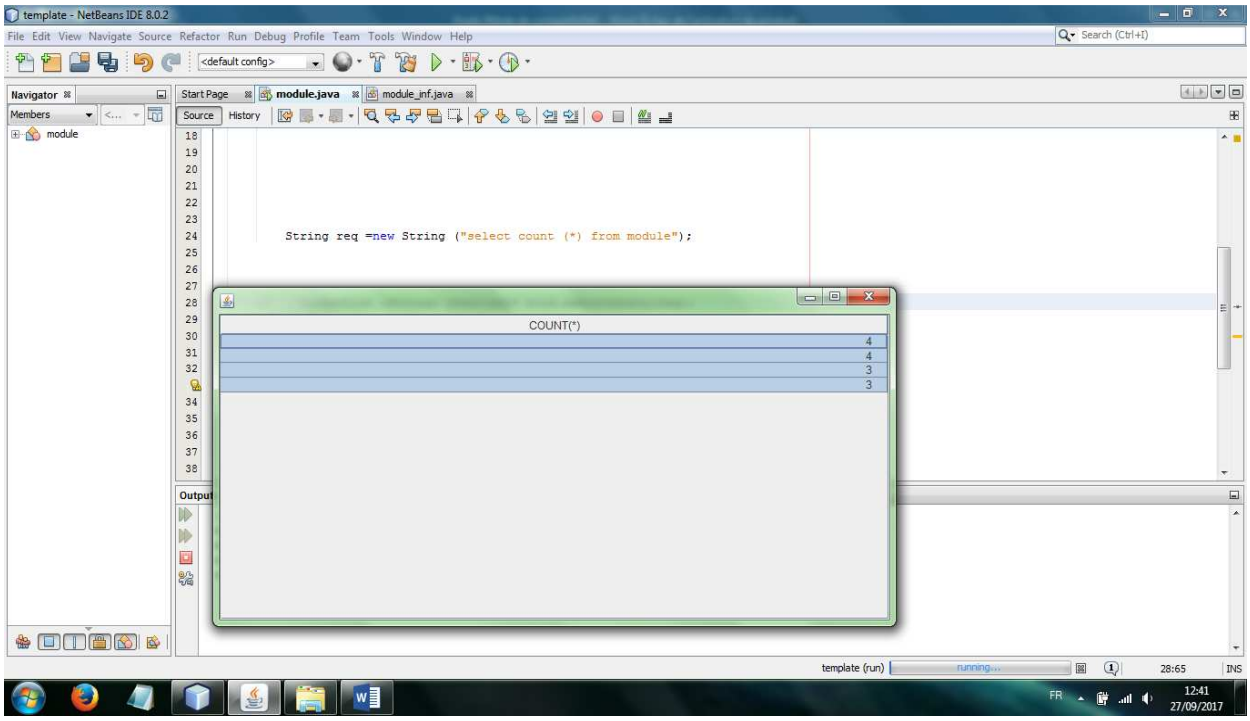


Figure IV.7 : Résultat d'application de patron Template Method pour récupérer le nombre de modules dans chaque département.

IV.4. Conclusion :

Oracle 10g est un moyen de développement efficace pour montrer la possibilité d'intégrer plusieurs sous-systèmes pour avoir un seul système homogène au moyen de lien de base de données et les patrons de conception.

L'utilisation de lien de base de données a permet aux requêtes SQL de faciliter l'accès aux autres bases de données que ce soit distantes ou bien locales situées sur une même machine d'où notre cas, et l'utilisation des patrons de conception garantissant l'accès à des données des différents sous-systèmes sans les consulter chacun à part et cela au moyen des interfaces communes.

Conclusion générale :

En vue de la complexité et l'hétérogénéités des systèmes actuels. Il convient par conséquent de les intégrer afin de les faire communiquer et de les faire coopérer. Certains systèmes pour des problèmes d'interopérabilité de causes diverses, sont appelés à migrer vers de nouvelles versions. L'objectif étant de rendre le système le plus agile et le plus réactif possible, tout en préservant le patrimoine informationnel existant. Il s'agit de la problématique d'intégration et d'interopérabilité des systèmes dans desenvironnements répartis. [4]

Au sein de notre projet nous avons essayé de faire l'intégration des données et de traiter le problème de l'hétérogénéités des bases de données, et pour ce faire nous avons opté pour les patrons de conception du GOF et les liens de bases de données.

Pour réaliser cette intégration nous avons utilisé certains patrons de conception qui répondent le plus à nos besoins avec les liens de base de données d'oracle, alors ces deux issus nous ont permet d'envoyer des requêtes, et faire l'union des réponses des quatre bases développées l'une indépendamment de l'autre.

Pour assurer cette intégration, le travail est partitionné en quatre chapitres, dans le premier chapitre nous avons parlé de l'interopérabilité et d'intégration. Ensuite nous avons abordé dans le deuxième chapitre des patrons de conception, et les liens de base de données.

En arrivant au troisième chapitre dont nous avons mis en œuvre une conception ;

Enfin le dernier chapitre qui est la réalisation, ou nous avons défini les différents outils de développent et langages de programmation en plus les réponses des requêtes émises.

On conclut qu'au cours de ce projet nous avons acquispas mal de connaissances sur lespatrons de conception, et les liens de bases de données.

Résumé

Le système d'information est une rubrique essentielle du fonctionnement d'une organisation et l'un des facteurs majeurs de sa réussite. Aujourd'hui un système d'information n'est pas constitué d'une seule application, il a énormément évolué surtout avec l'évolution des technologies logicielles (objet, composant, services web, …), l'évolution des technologies matérielles et aussi avec l'évolution des organisations (fusion, acquisition, mondialisation). Comme conséquences de tous ces facteurs et avec le temps, les systèmes d'information deviennent de plus en plus complexes et hétérogènes.

L'interopérabilité est la capacité de communiquer, exécuter des programmes, transférer des données entre différentes unités fonctionnelles de manière à ce qu'un utilisateur n'ait besoin que de peu ou pas de connaissances sur les caractéristiques de ces unités.

L'intégration est souvent considérée comme allant plus loin que l'interopérabilité, en forçant une certaine dépendance fonctionnelle des applications. Alors que des systèmes interopérables peuvent fonctionner indépendamment. Un système intégré est donc composé d'applications interopérables mais ces applications ne réalisent pas nécessairement un système intégré. En complément de ces deux vues systémiques, certains systèmes peuvent n'être que compatibles. Ces systèmes n'interfèrent pas directement entre eux. Cela n'implique donc pas qu'ils soient capables d'échanger des services. Des systèmes interopérables sont, par définition, compatibles, au moins en partie, mais l'inverse ne l'est pas.

Notre objectif était de réaliser l'intégration de base de données en utilisant les patrons de conception et les liens de base de données.

Webliographie

[1] : https://www.universalis.fr/encyclopedie/systemes-d-information/

[2] : https://hal.archives-ouvertes.fr/file/index/docid/119423/filename/HDR_H._Panetto.pdf

[3] :https://www.ekito.fr/people/linteroperabilite-pourquoi-sy-interesser/

[4] : http://beru.univ-brest.fr/~singhoff/ENS/UE_systemes_repartis/CM/sd.pdf

[5] : http://www.human-side.com/wilber/Integration/Integration.htm

[6] : http://beru.univ-brest.fr/~singhoff/ENS/UE_systemes_repartis/CM/sd.pdf

[7] : https://www.memoireonline.com/02/11/4278/m_Conception-et-realisation-dune-base-de-donnees-repartie-sous-oracle--cas-de-lhebergement-d1.html

[8]:http://www.strategie-aims.com/events/conferences/8-xveme-conference-de-l-aims/communications/2261-fusion-dentreprises-et-integration-des-systemes-dinformation

[9] : http://docinsa.insa-lyon.fr/these/2008/millet_p-a/these.pdf

[12] :http://www.altic.org/decouvrir-notreoffre/datamanagement/integration-de-donnees

[15] :http://www.bimfrance.net/glossaire/interoperabilite/

[16] :http://jalon.unice.fr/Members/pstaccin/Fichiers/staccini-pascal-p02.pdf/at_download/file

[17] :https://www.esante.lu/portal/fr/

[18] :https://www.e-health-suisse.ch/fr/technique-semantique/interoperabilite-technique.html

[19] : https://cours.unjf.fr/file.php/135/Cours/C2iD4-LutteCyberCrim/D4-3%20Bezzazi/co/S09_GC03_1.html

[20] :http://dl.acm.org/citation.cfm?id=1331740.1331888&preflayout=flat

[22] :www.archipel.uqam.ca/2435/1/M11000.pdf

[23] : https://fr.wikipedia.org/wiki/M%C3%A9diateur_(patron_de_conception)

[24] : https://*www*.liris.cnrs.fr/Documents/Liris-6339.pdf

[25] : https://fr.wikipedia.org/wiki/M%C3%A9diateur_(patron_de_conception)

[26] :https://dpt-info.u-strasbg.fr/~blansche/files/ogl_cours_5.pdf

[27] : http://www.aide-oracle.net/2009/04/dblink-interroger-une-base-distante.html

[28]:https://docs.oracle.com/cd/B28359_01/server.111/b28310/ds_concepts002.htm#ADMIN12085:

[29] : http://www.aide-oracle.net/2009/04/dblink-interroger-une-base-distante.html

[30] : http://www.wikilivre.com

[31] : http://www.Oracle.com

Bibliographie

[10]: David S. Linthicum, Enterprise Application Integration, Ed: Addison Wesley, 2001 ISBN: 0-201-61583-5.

[11]: S.Janarthanan, A. Ekambaram, M.M. Kusum, N.K. Nair, Integrating heterogeneous systems using Enterprise Application Integration, research & analysis report, pp (1- 20).

[13] : Saïd Izza, Intégration des systèmes d'information industriels : une approche flexible basée sur les services sémantiques, thèse de doctorat de l'école supérieure nationale des Mines de Saint-Étienne, novembre 2006.

[14] : Mémoire Magister,« Systèmes distribués ». Melle. Soumia BENDEKKOUM.

[21] :Mémoire de Doctorat « Contribution à la conception d'une plateforme d'anticipation et d'aide à la décision en management de projet » MELLAL Leila , Université de –BATNA 2

[28]: Oracle DatabaseAdministrator's Guide- 11*g* Release 2 (11.2)