Projet de SGBD

[IT204]

Rapport de projet

3/12/2007

Gestion d'une agence immobilière

Encadrant :

Mohamed MOSBAH

Équipe

Amal KHABOU

Mohamed Amine EL AFRIT

Mohamed Mehdi MAKHLOUF

Table des matières

Table des figures

Introduction

La conception des bases de données est une tâche assez délicate nécessitant beaucoup de rigueur pour éviter les problèmes de redondance et d'autres défauts d'intégrité.
Dans le cadre de ce projet on été amené à réflechir sur la création d'une base de données dont le but est la gestion d'un parc immobilier. Cette application concrète, nous a immédiatement confronté aux différentes difficultés et interrogations qui peuvent se poser lors de la création d'une base de données.
Le client est une agence de gestion de biens qui cherche à organiser des quartiers, immeubles, logements dans différentes villes, gérer des revenus et contrôler l'état global. Notre but est donc de collecter les données, les organiser, les modéliser pour faciliter leur gestion finale en tant qu'informations utiles à notre agence.
Ainsi ce projet se compose de plusieurs parties :

- Analyse du sujet et spécification d'un cahier des charges.
- Conception : dans le cadre des aboutissements de la première étape, on a élaboré un modèle entité association permettant de répondre aux fonctionnalités de notre projet.
- Modèle relationnel : le modèle conçu précédamment nous permet de passer à létape de modélisation relationnelle qui permet de concevoir de proche en proche la cohérence et l'intégrité de notre future base de données.
- Réalisation : elle consiste à la mise en place de la base de données ainsi des différentes commandes SQL assurant sa bonne gestion.
- Utilisation : on y explique le fonctionnement des interfaces réalisées pour une meilleure communication avec la base de données.

Chapitre 1

Analyse du problème

Le projet s'articule autour de la création de la base de données d'une agence immobilière pour lui permettre une bonne gestion de son parc locatif : elle devrait permettre lórganisation des informations qui concernent ses clients ainsi que le suivi quotidien de son évolution financière, dans ce contexte on a essayé d'envisager le bon modèle des donnés qui permettra d'aboutir à cette finalité.

1.1 Analyse et aboutissements

Le sujet impose bien la considération des entités de base ville, quartier, immeuble et logement qui permettent la gestion financière et géographique du parc immobilier : charges perçues, recherche de logement dans une ville donnée, dans un quartier donné .... Ainsi on a considéré ces quatre entités de base liées entre elles par des simples associations d'appartenance (voir modèle entité association du chapitre suivant).

Puis on s'est intéréssé aux clients de l'agence, ces derniers sont principalement les propriétaires et les personnes intervenant lors de la mise en place dún bail. Lors d'une première analyse du sujet on a essayé d'exhiber tous les types de personnes concernés par un bail et un logement : il y a biensur les locataires qui peuvent être ou pas des signataires du bail et inversement il y a les signataires qui eux sont des occupants du logement considéré ou ne le sont pas comme : les garants et les signataires payants. Vu la variété des clients de l'agence, on a fini par considérer l'entité personne qui englobent toutes les personnes et on a géré les différents caractères à travers des associations. Ainsi l'entité personne est liée à l'entité logement par l'association **appartienta** définissant le propriétaire du logement, et elle liée à léntité bail par 2 associations : **signe_par** et **definit_locataire**. En fait lors de la mise en place d'un bail, on s'intérésse aux personnes qui l'ont signé, une telle personne a un type bien déterminé : soit un garant, soit un signataire payant ou un simple signataire et dans ce cas il sera obligatoirement un occupant du logement concerné sinon pourquoi signera-t-il le contrat ? Le bail définit aussi les locataires du logement en question et ceci à travers l'association appropriée.

D'autre part, on a considèré que l'entité bail est liée à l'entité logement. Ce choix se justifie par le fait qu'un bail renseigne bien sur la disponibilité du logement : loué ou non. Ainsi le bail est lié au logement par l'association **fait_lóbjet**, aux personnes par les associations **signe_par** et **definit_locataire** mais aussi à l'entité facture pour identifier celles qu'il génère, en effet pendant sa durée de vie un bail génère chaque mois une facture dont la validation séffectue suite à son payement soit, par le remplissage du champ date_payement.

Cette analyse du sujet nous a permis d'élaborer une première approche du problème qui sera améliorer en considèrant un ensemble d'hypothèses pour pouvoir réaliser les fonctionalités envisagées.

1.2 Hypothèses et Règles de gestion

- Toute personne locataire, signataire du bail ou propriétaire aura un numéro dans notre agence. On considère que cést son numéro de sécurité sociale, bienque ça ne soit pas assez significatif.

- Tout appartement a un seul propriétaire.

- Chaque appartement a un numéro absolu **num_logement** relativement à tous les logements gérés par notre agence.

- Chaque immeuble a un nom unique qui le distingue de tous les autres.

- On suppose que chaque quartier a un nom unique qui le distingue de tous les autres ainsi on ne peut pas trouver deux quartiers ayant le même nom dans deux villes différentes.

- Un bail est signé par plusieurs personnes qui ne sont pas tous des occupants du logement en question.

- Lors de la signature du bail , on récupère à la fois toutes les informations concernant les locataires même si ils ne signent pas le bail autrement dit on ne se contente pas d'avoir le nombre des occupants lors de la signature du contrat .

- La caution est égale à deux fois le loyer.

- Le loyer pour chaque logement est une somme fixe payée mensuellement et elle est précisée dans la facture.

- Les charges mensuelles est une somme variable payée mensuellement et elle est précisée dans la facture.

- Les personnes qui signent le bail ne sont pas toutes concernées par le règlement du loyer d'où límportance de la précision du type du signataire. En fait un garant peut signer le bail sans être responsable du règlement du loyer mensuellement. Ainsi on suppose que seuls les signataires_payant se chargent des payements des loyers.

- La gestion du règlement du loyer dans notre modèle se fait de la façon suivante : une facture n'est validée que lorsqu'elle est payée c'est à dire que sa **date_payement** est bien remplie, normalement les loyers sont à règler au début de chaque mois donc on considère qu'il y a un retard de payement si le règlement a été effectué aprés les 10 premiers jours du mois en cours. En effet, lágence síntéresse aux retards de payement à la fin de chaque mois.

- Quand il s'agit d'un retard du loyer, on récupère la liste de tous les locataires du logement en question même ceux qui ne sont pas concernés par le payement.

- Chaque propriétaire aura 95% du loyer, les 5% seront récupéres par l'agence à part les frais fixes payés par le propriétaire annuellement. Cette règle de gestion explique juste la façon de calculer le revenu des propriétaires.

- Dans le cas où le proprétaire d'un immeuble ou d'un logement décide de ne plus le louer, on ne les supprime pas de la base, on opérera juste un suppression logique. En fait une suppression physique signifie une suppression des baux ainsi des factures en relation avec eux c'est à dire des pertes d'informations nécessaires pour la gestion des historiques : on effectuera cette gestion par des vues.

1.3 Fonctionalités

L'application devra assurer plusieurs fonctionnalités :les opérations standard de consultation et insertion ainsi celles correspondant à la recherche d'informations particulières. Notamment :

- Liste des quartiers dans une ville donnée.
- Liste des immeubles dans un quartier donné.
- Liste des baux en cours dans une ville donné a une date donnée.
- Charges annuelles percues par un immeuble donné.
- Adresse d'un logement donné.
- Liste des logements occupes a une date donnée.
- Liste des logements vides dont le loyer_moy est inferieur a un prix donné.
- Revenus totaux d'une famille donée occupant un logement d'un immeuble donné.
- Loyer, avec les charges, paye mensuellement par un locataire.
- Total mensuel moyen des loyers collectes par un proprietaire donné.
- Liste des occupants ayant un retard de loyer pour un immeuble donné.
- Renouvellement de bail avec ou sans maintien des locataires.
- Historique des factures pour un logement donné.

Chapitre 2

Modèle conceptuel

2.1 Description des entités :

Dans notre schéma entité association on trouve les entités suivantes :

- **Ville** ; une ville étant définie par un code postal et un nom. Le code postal est la clé primaire car chaque ville a un code postal unique.

- **Quartier** ; un quartier est défini par un identifiant unique le numéro de quartier. Les autres attributs sont le nom du quartier et le code postal ou se trouve le quartier. Le code postal étant une clé étrangère de l'entité ville.

- **Immeuble** ; un immeuble est défini par un identifiant unique son numéro, un nom, le numéro du quartier dans lequel il se trouve, une date de construction, une date de dernière réfection, un complément d'adresse (le numéro et le nom de la rue) qui va nous permettre après de rassembler l'adresse d'un logement en faisant la concaténation avec le code postal. Il y a aussi l'attribut état qui nous informe si l'immeuble est disponible ou non car en cas de travaux l'immeuble n'est pas opérationnel. On remarquera que plusieurs immeubles peuvent avoir le même nom, cela arrive souvent, c'est pourquoi un immeuble possède un identifiant unique qui n'est pas son nom. On remarquera aussi qu'un immeuble ne peut pas être déterminé seulement grâce à son adresse car il arrive que dans certaines rues les numéros de rue ne soient pas indiqués, donc le nom de la rue, le code postal, ainsi que la ville ne permettent pas de déterminer un immeuble ; par contre le couple formé du nom d'un immeuble et de son adresse permet de déterminer de façon unique un immeuble.

- **Catégorie** ; une catégorie est définie par un identifiant unique son numéro. La catégorie est relative au logement. En effet, il nous a semblé plus plausible de réaliser une table catégorie plutôt que de mette seulement un attribut catégorie dans logement. L'entité catégorie comporte le nom de la catégorie (F1,studio), la surface du logement , le loyer minimal , maximal ainsi que moyen pour un genre de catégorie .

- **Etat** : A l'instar de l'entité catégorie, on a fait une entité état qui est identifie par un numéro unique et un nom. Cette entité décrit l'état du logement (à rénover, vétuste , bon).

- **Personne** ; L'entité personne rassemble tous les comptes clients : propriétaire, locataire et payant. Bien qu'on ait pas besoin des mêmes informations pour ces trois types de clients, une seule entité nous permet d'éviter le risque de duplication de données si un client s'avère en même temps payant et occupant à la fois par exemple. Une personne est définie par un numéro de sécurité sociale unique (nss). Les autres attributs sont le nom, le prénom, la date de naissance la profession, le revenu, la situation professionnel, et les prestations sociales.

- **Logement** ; un logement étant défini par un identifiant unique, son numéro. Les autres attributs sont le numéro de l'immeuble dans lequel il se trouve, le numéro de catégorie et d'état. Aussi le nss du propriétaire

et l'adresse du propriétaire font partie des attributs. Un logement peut être en travaux donc indisponible c'est pour cela qu'on a ajouté l'attribut disponibilité ainsi que la date de dernière réfection.

- **Bail** ; Le choix de bail comme entité était indispensable. Elle est caractérisée par un identifiant unique. Le bail fait intervenir un logement c'est pour cela qu'on a ajouté l'attribut le numéro du logement. Les autres attributs sont la caution versée, la date de signature ainsi que la date de fin du bail.
  Remarque : On ne peut pas résilié un bail avant la date de fin.

- **Facture** ; Une facture est caractérisé par un numéro unique, le numéro de bail auquel fait référence la facture, une date de paiement ainsi que les charges et le loyer mensuel pour une facture. A prima bord , on a pense à mettre les attributs de facture dans l'entité bail mais très vite on a constaté les limites de ce model. Donc notre choix s'est porté sur une entité facture qui va nous permettre de reconstituer un historique de paiement des loyers. D'autre part, il faut noter la différence entre cette entité et l'attribut loyer de la table catégorie. En effet, on peut modifier le loyer dans l'entité catégorie mais on peut plus le changer et il reste fixe dés la signature d'un bail.

2.2 Les associations

- Un quartier **est situé** dans une ville.
- Un immeuble **est dans** un quartier.
- Un immeuble **contient** un logement.
- Un logement **est dans** l'état état et il **est de type** catégorie.
- Un logement **fait l'objet** d'un bail.
- Un logement **appartient** à une personne.
- Un bail **est signé** par une ou plusieurs personnes.
- Un bail **définit** un ou plusieurs occupants.
- Un bail **génère** une facture.
- Une personne peut **avoir à charge** avec une ou plusieurs personnes. Le fait d'avoir toutes les personnes dans une même table nous aide à faire cet association.

2.3 Les contraintes

2.3.1 Les cardinalités

1. Une ville contient **0 ou N** quartier(s). Un quartier est contenu dans une **unique** ville.
2. Un quartier contient **0 ou N** immeubles, Un immeuble est localisé dans un **seul** quartier.
3. Un immeuble contient **0 ou N** logements, un logement est contenu dans 1 **seul** immeuble.
4. Un logement **a un seul** type de catégorie et d'état, alors qu'un type ou une catégorie peuvent être associés à **0 ou N** logements.
5. Un logement a un **seul** type de catégorie et d'état, alors qu'un type ou une catégorie peuvent être associés à **0 ou N** logements.
6. Un logement peut faire l'objet de **0 ou N** baux. Ceci nous permet d'avoir un historique des baux. Deux baux concernant le même logement ont nécessairement 2 périodes de validité disjointes. Un bail concerne un seul logement.
7. Un bail peut être signé par **1 ou N** personnes. Une personne peut signer **0 ou N** baux. De la même manière un bail définit **1 ou N** locataires et une personne peut faire l'objet de **0 ou 1** bail.
8. Un bail génère **1 ou N** factures. Et une facture est relative à un **seul** bail.
9. Une personne peut avoir **0 ou N** personne à charge.

2.3.2 Contraintes inter-relations :

Pour assurer la cohérence des données après insertion, modification ou suppression il faut exiger certaines règles :
Un quartier ne peut exister que dans une seule ville. Donc on peut pas insérer un logement sans avoir inséré auparavant l'immeuble dans lequel il se trouve et de même pour la ville. On ne peut jamais supprimer de bail même si le logement n'existe plus, sinon on perd l'historique du paiement. De même pour les factures , les logements, les immeubles et les villes. On fera de la suppression logique sur ces entités.

2.4 Schema conceptuel :

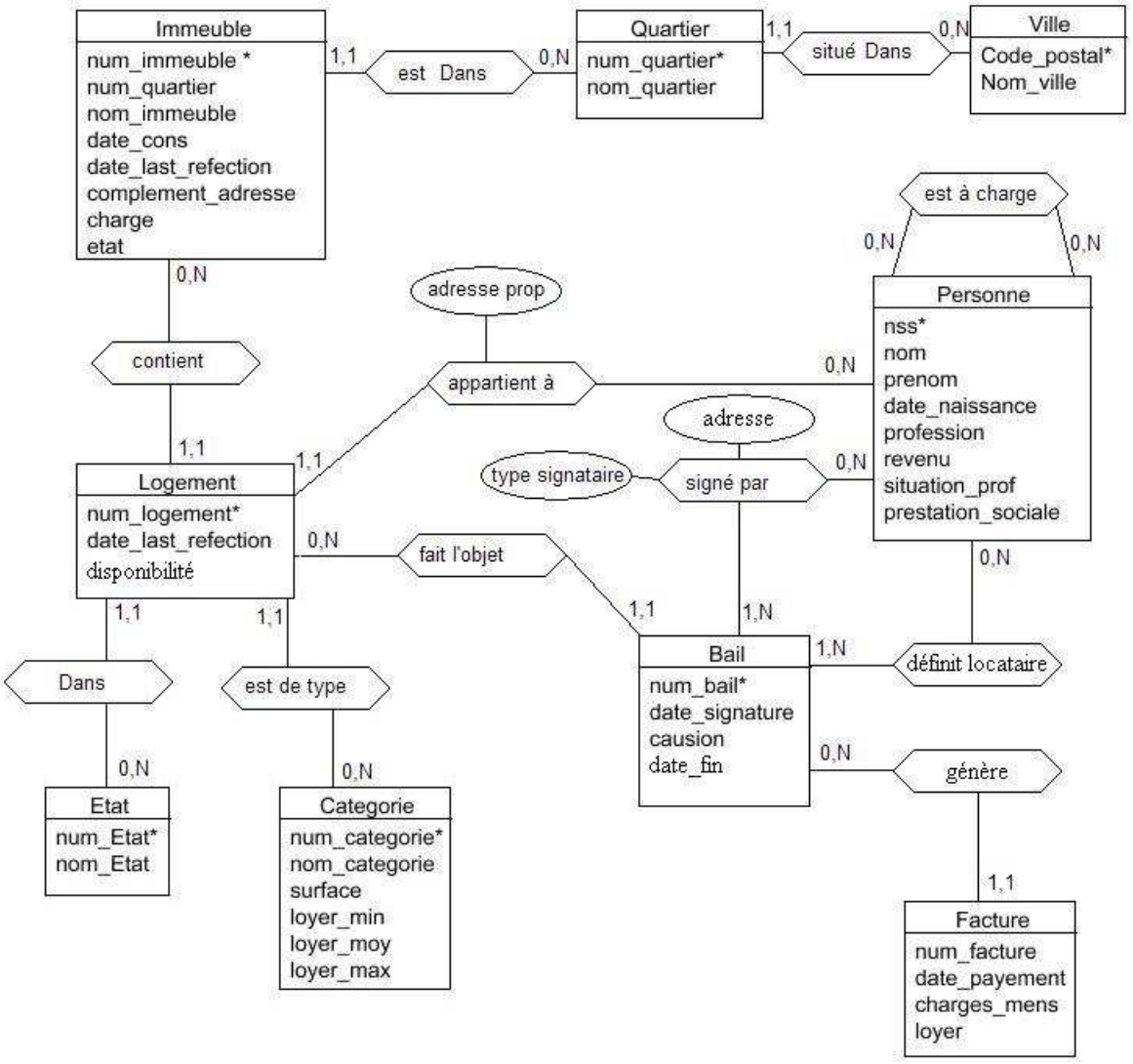

FIG. 2.1 – Schéma conceptuel

2.5 Limites du modèle

Le modèle ainsi conçu, bienqu'il réponde à la quasi totalité des fonctionnalités, présente certaines limites par rapport à la réalité. On a remarqué quelques unes lors de la réalisation, c'était assez tard pour les prendre en considération et essayer d'y répondre mais on a jugé intéressant le fait de les signaler, nottamment :

- Pour les logements, on confond le numéro du logement au numéro du logement dans l'agence, donc on n'autorise pas que deux logement de deux immeubles différents aient le même numéro, ce qui est assez déplacé par rapport à la réalité. En effet, un logement aurait pu avoir un numéro relatif à l'immeuble dans lequel il se trouve, et c'est ce dernier numéro qui figurera dans l'adresse du logement et non pas son numéro dans l'agence.
- On ne peut pas savoir la part du loyer payée par chaque personne dans le cas où plusieurs signataires du bail partagent le loyer, on n'a accés qu'à une seule information : la totalité du loyer a été payée ou pas encore. On aurait pu peut être ajouter une entité **payer** pour une telle gestion.
- On ne traite pas le cas où deux personnes ont le même nom et le même prénom à la fois et on suppose que ceci ne se présente pas au moins dans la base qu'on a créée pour tester nos requêtes, surtout que certaines ont comme paramètre le nom et le prénom d'une personne ce qui pourra induire des problème en cas de double.
- Pour les charges annuelles communes d'un immeuble on considère qu'il s'agit d'une somme fixe que l'agence sait estimer quelque soient les travaux nécessaires pour son entretien ce qui est assez simpliste comme démarche, il a fallu peut être la prendre comme une entité à part et non pas un simple attribut.

Chapitre 3

Modèle relationnel

3.1 Contraintes d'intgrité

	Nom attribut	Type	Non nul	Index	Conditions
VILLE	code_postal	int	√	PRIMARY KEY	
	nom_ville	char(20)	√		
QUARTIER	num_quartier	int	√		auto increment
	nom_quartier	char(20)	√		
	Code_postal	int	√		
IMMEUBLE	num_immeuble	int	√	PRIMARY KEY	auto increment
	num_quartier	int	√	FOREIGN KEY	
	nom_immeuble	char(20)	√		
	date_cons	date			
	date_last_refection	date			date_cons ≤ date_ref
	complement_adresse	char(40)	DEFAULT		
	etat	int	DEFAULT (dispo)		non dispo si supprimee
CATÉGORIE	num_categorie	int	√	PRIMARY KEY	auto increment
	nom_categorie	char(10)	√		
	surface	int	DEFAULT 0		
	loyer_min	int	DEFAULT 0		
	loyer_moy	int	DEFAULT 0		
	loyer_max	int	DEFAULT 0		
ETAT	num_etat	int	√	PRIMARY KEY	auto increment
	nom_etat	char(10)	√		
PERSONNE	nss	int	√	PRIMARY KEY	auto increment[1]
	nom	char(15)	√		
	prenom	char(15)	√		
	date_naissance	date	√		
	profession	char(30)	DEFAULT (NC)		
	revenu	int	DEFAULT 0		
	situation_prof	char(20)	DEFAULT (NC)		
	prestation_sociale	int	DEFAULT (0)		
LOGEMENT	num_logement	int	√	PRIMARY KEY	auto increment
	num_immeuble	int	√	FOREIGN KEY	
	num_categorie	int	√	FOREIGN KEY	
	num_etat	int	√	FOREIGN KEY	
	nss[2]	int	√	FOREIGN KEY	
	adresse_prop	char(80)	DEFAULT (NC)		
	date_last_refection	date	√		
	disponibilite	char(10)	DEFAULT (dispo)		

	Nom attribut	Type	Non nul	Index	Conditions
BAIL	num_bail	int	√	PRIMARY KEY	auto increment
	num_logement	int	√	FOREIGN KEY	
	date_signature	date	√		
	caution	int	DEFAULT (0)		
	date_fin	date	√		
FACTURE	num_facture	int	√	PRIMARY KEY	auto increment
	num_bail	int	√	FOREIGN KEY	
	date_payement	date	√		
	charge_mens	int	DEFAULT (0)		
	loyer	int	√		
SIGNE_PAR	nss	int	√	PRIMARY KEY	
	num_bail	int	√	PRIMARY KEY	
	type_sign	char(25)	DEFAULT (NC)		
	adresse_sign	char (60)	DEFAULT (ND)		
DEFINIT_LOCATAIRE	nss	int	√	PRIMARY KEY	
	num_bail	int	√	PRIMARY KEY	
EST_A_CHARGE	nss1	int	√	PRIMARY KEY	
	nss2	int	√	PRIMARY KEY	

3.2 Schéma relationnel

VILLE (code_postal, nom_ville)

QUARTIER (num_quartier, nom_quartier, code_postal)

IMMEUBLE (num_immeuble, num_quartier, nom_immeuble, date_cons, date_last_refection, complement_adress, etat)

CATÉGORIE (num_categorie, nom_categorie, surface, loyer_min, loyer_moy, loyer_max)

ETAT (num_etat, nom_etat)

PERSONNE (nss, nom, prenom, date_naissance, profession, revenu, situation_prof, prestation_sociale)

LOGEMENT (num_logement, num_immeuble, num_categorie, num_etat, nss, adress_prop, date_last_refection, disponibilite)

BAIL (num_bail, num_logement, date_signature, caution, date_fin)

FACTURE (num_facture, num_bail, date_payement, charge_mens, loyer)

SIGNE_PAR (nss,num_bail, type_sign, adress_sign)

DEFINIT_LOCATAIRE (nss,num_bail)

EST_A_CHARGE (nss1,nss2)

3.3 Dépendance fonctionnelle

code_postal	$\longrightarrow$	nom_ville
num_quartier	$\longrightarrow$	nom_quartier, code_postal
num_immeuble	$\longrightarrow$	num_quartier, nom_immeuble, date_cons, date_last_refection, complement_adresse, etat
num_categorie	$\longrightarrow$	nom_categorie, surface, loyer_min, loyer_moy, loyer_max
num_etat	$\longrightarrow$	nom_etat
nss	$\longrightarrow$	nom, prenom, date_naissance, profession, revenu, situation_prof, prestation_sociale
num_logement	$\longrightarrow$	num_immeuble, num_categorie, num_etat, nss, adresse_prop, date_last_refection, disponibilite
num_bail	$\longrightarrow$	num_logement, date_signature, caution, date_fin
num_facture	$\longrightarrow$	num_bail, date_payement, charge_mens, loyer
(nss, num_bail)	$\longrightarrow$	type_sign, adress_sign
(nss1, nss2)	$\longrightarrow$	*evident*

3.4 Normalisation de la base en 3 forme normale

VILLE (code_postal, nom_ville)

En premier lieu tous nos attributs sont atomiques et on dispose bien d'une clé unique qui est le code postal ainsi on a la 1ere forme normale.On respecte bien la 2eme forme normale puisque notre clé est simple (non composite) et que l'attribut nom_ville dépend de la clé donc on a bien dépendance de la totalite de la clé (FN2).Enfin on a bien FN3 (comme l'attribut dépend directement de la clé).
FN3 respectée

QUARTIER (num_quartier, nom_quartier, code_postal)

FN3 respectée (évident)

IMMEUBLE (num_immeuble, num_quartier, nom_immeuble, date_cons, date_last_refection, complement_adress, etat)

Tous les attributs sont atomiques donc FN1 est validée. Ensuite notre clé étant simple non composite donc on a bien dépendance de la totalité de la clé et non d'une partie stricte pour tous les attributs. Donc FN2. On a FN3 car aucun attribut ne dépend de attributs hors clés.

CATÉGORIE (num_categorie, nom_categorie, surface, loyer_min, loyer_moy, loyer_max)

FN3 respectée (évident).

ETAT (num_etat, nom_etat)

FN3 respectée (évident).

PERSONNE (nss, nom, prenom, date_naissance, profession, revenu, situation_prof, prestation_sociale)

nss est bien une clé car aucune combinaison des autres attributs ne fournit une clé meme ayant le nom , le prenom et la date de naissance on ne peut pas retrouver les valeurs des autres attributs . Enfin comme tous les attributs sont atomiques :FN1 respectée. Comme la clé est atomique, ainsi on a bien dépendance de la totalité de la clé pour tout attribut :FN2 respectée. Aucun attribut hors cle ne depend d'un attribut ou groupe d'attributs hors clés , par exemple la profession et la situation professionnelle ne suffisent pas pour avoir le revenu :FN3 respectée.

LOGEMENT (num_logement, num_immeuble, num_categorie, num_etat, nss, adress_prop, date_last_refection, disponibilite)

num_Logement est bien une clé car le num_immeuble et nss ne renseignent rien sur l'appartement consideré, les attributs sont tous atomiques : FN1 respectée.
Encore une fois la cle est simple ce qui garantit la FN2.
FN3 : aucun attribut hors clé ne dépend d'un attribut ou groupe d'attributs hors clés. Par exemple le num_immeuble ne donne rien car un immeuble peut contenir differents types de logements de differentes surfaces, de meme la connaissance de num_immeuble et nss ne donne aucun attribut car un proprietaire peut avoir plusieurs différents appartements dans le meme immeuble.

BAIL (num_bail, num_logement, date_signature, caution, date_fin)

num_Bail est la clé , en fait tous les attributs dépendent fonctionnellement du num_Bail . Tous les attributs sont atomiques :FN1 respectée.

La clé est simple non composite donc tous les attributs dependent bien de la totalité de la clé et non d'une partie stricte :FN2 respectée.

Meme si on dispose du numéro du logement et de la date de signature on ne peut pas retrouver ni la caution ni la date_fin car notre agence fait des prix speciaux pour les étudiants ou autres personnes (par exemple des partenariats avec des entreprises) : Ainsi aucun attribut hors clé ne depend d'un attribut ou d'un groupe d'attributs hors clés :FN3 respectée.

FACTURE (num_facture, num_bail, date_payement, charge_mens, loyer)

num_Facture est bien une clé qui permet de retrouver tous les autres attributs qui sont atomiques :FN1 respectée.

La clé étant non composite donc :FN2 respectée.

Enfin la date de payement et le numéro du logement ne permettent pas de retrouver le loyer et les charges : FN3 respectée.

SIGNE_PAR (nss,num_bail, type_sign, adress_sign)

Les attributs sont atomiques et elle posséde une clé :FN1.

Les 2 attributs hors clés type_sign et adress_sign ne dependent pas d'une partie de la clé.En effet si on a le nss d'une personnee on peut pas avoir son adresse car on a pas d'attribut adresse dans la table personne.Aussi le type signataire ne depend pas du nun_bail seulement : FN2 respectée.

Les 2 attributs hors clés sont independants : FN3 respectée.

DEFINIT_LOCATAIRE (nss,num_bail)

Les 2 clés sont atomiques.

FN3 respectée (évident).

EST_A_CHARGE (nss1,nss2)

FN3 respectée (évident).

Chapitre 4

Réalisation : oracle9i

4.1 Création de la base

Aprés la conception et la modélisation de la structure de la base, on s'est intéressé á la création des tables de données. Lors de la création des tables, il a fallu suivre un certain ordre respectant les contraintes d'intégrité inter-tables de type référentiel. Ainsi chaque clé étrangère doit référencier une table qui existe déjà. En fait, pour la mise en place de la base on a créé 3 fichiers : creation.sql, drop.sql et remplissage.sql. Le premier permettant de créer les séquences, les tables et les vues, le deuxième de les effacer , et le troisième pour remplir les tables avec quelques lignes significatives qui vont nous servir pour tester les requêtes. Le fichier lancer.sql permet de faire appel aux trois fichiers précédents : c'est un script de mise en place de la base.

4.1.1 Création des tables, des séquences et des vues

Dans le fichier creation.sql, on a créé les séquences, elles correspondent aux clés qui s'auto-incrémentent, puis les tables tout en respectant les contraintes référentielles inter-tables, et enfin les vues. Pour ces dernières, on a envisagé leur création à une étape assez avancée de la réalisation du projet, plus précisemment en s'interessant aux mises à jour de la base et en particulier en traitant les suppressions, la suppression physique étant non autorisée vu qu'on a besoin des historiques, il a fallu donc introduire de nouvaux attributs dans les tables immeuble et logement et de réaliser une suppression logique et ceci grâce aux vues.

Les séquences

- CREATE SEQUENCE seq_quartier START WITH 1 INCREMENT BY 1;
- CREATE SEQUENCE seq_immeuble START WITH 1 INCREMENT BY 1;
- CREATE SEQUENCE seq_categorie START WITH 1 INCREMENT BY 1;
- CREATE SEQUENCE seq_etat START WITH 1 INCREMENT BY 1;
- CREATE SEQUENCE seq_personne START WITH 1 INCREMENT BY 1; Pour la table personne bienque la clé primaire soit comme mentionné précédamment, le numéro de sécurté sociale, on a choisit, pour des raisons de simplicité, de le présenter par un numéro.
- CREATE SEQUENCE seq_logement START WITH 100 INCREMENT BY 1;
- CREATE SEQUENCE seq_bail START WITH 1 INCREMENT BY 1;
- CREATE SEQUENCE seq_facture START WITH 5000 INCREMENT BY 1;

Les Tables

```
CREATE TABLE ville (code_postal INT NOT NULL PRIMARY KEY,
              nom_ville CHAR(20) NOT NULL);
```

⇒ création de la table **ville**.

```
CREATE TABLE quartier (num_quartier INT NOT NULL PRIMARY KEY,
                nom_quartier CHAR(20) NOT NULL,
               code_postal INT NOT NULL,
          FOREIGN KEY (code_postal) REFERENCES ville) ;
```

⇒ création de la table **quartier** avec respect de la contrainte d'intègrité pour la clé étrangère.

```
CREATE TABLE immeuble (num_immeuble INT NOT NULL PRIMARY KEY,
                num_quartier INT NOT NULL,
                nom_immeuble CHAR(20) DEFAULT 'immeuble',
          date_cons DATE,
          date_last_refection DATE,
          complement_adresse CHAR(40) DEFAULT 'complement adresse',
          charges INT NOT NULL ;
          etat CHAR(10)  DEFAULT 'dispo',
          CHECK (date_last_refection >= date_cons),
          FOREIGN KEY (num_quartier) REFERENCES quartier);
```

⇒ création de la table **immeuble** avec respect de tout type de contraintes.

```
CREATE TABLE categorie (num_categorie INT NOT NULL PRIMARY KEY,
                 nom_categorie CHAR(10) NOT NULL,
              surface INT DEFAULT 0,
           loyer_min INT DEFAULT 0,
           loyer_moy INT DEFAULT 0,
           loyer_max INT DEFAULT 0);
```

⇒ création de la table **categorie** .

```
CREATE TABLE etat (num_etat INT NOT NULL PRIMARY KEY,
              nom_etat CHAR(10) NOT NULL);
```

⇒ création de la table **etat**.

```
CREATE TABLE personne (nss INT NOT NULL PRIMARY KEY,
                nom CHAR(15) NOT NULL,
          prenom CHAR(15) NOT NULL,
          date_naissance DATE,
          profession CHAR(30) DEFAULT 'NC',
          revenu INT DEFAULT 0,
          situation_prof CHAR(20) DEFAULT 'NC',
          prestations_sociale INT DEFAULT 0);
```

⇒ création de la table **personne**, cette table ne référence aucune table.

```
CREATE TABLE logement (num_logement INT NOT NULL PRIMARY KEY,
                num_immeuble INT NOT NULL,
          num_categorie INT NOT NULL,
          num_etat INT NOT NULL,
          nss INT NOT NULL,
          adresse_prop CHAR(80) DEFAULT 'NC',
             date_last_refection DATE,
          disponibilite CHAR(10) DEFAULT 'dispo',
          FOREIGN KEY  (num_immeuble) references immeuble,
          FOREIGN KEY (num_categorie) references categorie,
```

```
        FOREIGN KEY (num_etat) references etat,
        FOREIGN KEY (nss) references personne);
```

⇒ création de la table **logement**, l'attribut disponibilite sert à préciser si le logement est toujours déstiné á la location ou non.

```
CREATE TABLE bail (num_bail INT NOT NULL PRIMARY KEY,
            num_logement INT NOT NULL,
            date_signature DATE
                  caution INT DEFAULT 0,
    date_fin DATE NOT NULL,
    FOREIGN KEY (num_logement) REFERENCES logement);
```

⇒ création de la table **bail**.

```
CREATE TABLE facture (num_facture INT NOT NULL PRIMARY KEY,
               num_bail INT NOT NULL,
               date_payement DATE DEFAULT "DD-MMM-YYYY",
        charges_mens INT DEFAULT 0,
         loyer INT NOT NULL,
     FOREIGN KEY (num_bail) REFERENCES bail);
```

⇒ création de la table **facture**, dans un premier temps, il y avait pas un attribut loyer pour cette entité, en effet on pourrait le récupérer dans la catégorie du logement en passant par la durée du bail et le numéro du logement concerné par ce bail, mais vu qu'il y a assez de requêtes qui s'intéressent aux payements, on a décidé de l'ajouter, en plus par définition une fature doit contenir les sommes à payer.

```
CREATE TABLE signe_par (nss INT NOT NULL,
num_bail INT NOT NULL,
          type_sign CHAR(25) DEFAULT 'NC',
        adresse_sign CHAR(60) DEFAULT 'AS',
PRIMARY KEY (nss, num_bail),
FOREIGN KEY (nss) REFERENCES personne,
FOREIGN KEY (num_bail) REFERENCES bail);
```

⇒ création de la table **signe_par**.

```
CREATE TABLE definit_locataire (nss INT NOT NULL,
                  num_bail INT NOT NULL,
        PRIMARY KEY (nss, num_bail),
            FOREIGN KEY (nss) REFERENCES personne,
            FOREIGN KEY (num_bail) REFERENCES bail);
```

⇒ création de la table **definit_locataire**.

```
CREATE TABLE est_a_charge (nss1 INT NOT NULL,
              nss2 INT NOT NULL,
        PRIMARY KEY (nss1),
            FOREIGN KEY (nss1) REFERENCES personne);
```

⇒ création de la table **est_a_charge**.

Les vues

```
CREATE VIEW immeuble_reel(num_immeuble,num_quartier,nom_immeuble,
date_cons,date_last_refection,complement_adresse,charges)
AS (SELECT num_immeuble, num_quartier, nom_immeuble, date_cons,
date_last_refection, complement_adresse, charges
        FROM immeuble
    WHERE (etat = 'dispo'));
```

⇒ création de la table **immeuble_reel**.

```
CREATE VIEW logement_reel(num_logement,num_immeuble,num_categorie,
num_etat,nss,adresse_prop,date_last_refection)
AS (SELECT num_logement, num_immeuble, num_categorie, num_etat, nss,
adresse_prop, date_last_refection
          FROM logement
      WHERE (disponibilite = 'dispo'));
```

⇒ création de la vue **logement_reel**.

La mise à jour des tables immeuble et logement influe sur les vues sans que l'utilisateur se rend compte de l'origine des modifications.

4.1.2 Suppression

Le fichier drop.sql permet la suppression des tables, des vues et des séquences.
Il s'organise comme suit :

- DROP VIEW logement_reel ;
- DROP VIEW immeuble_reel ;
- DROP TABLE signe_par ;
- DROP TABLE definit_locataire ;
- DROP TABLE est_a_charge ;
- DROP TABLE facture ;
- DROP TABLE bail ;
- DROP TABLE logement ;
- DROP TABLE personne ;
- DROP TABLE categorie ;
- DROP TABLE etat ;
- DROP TABLE immeuble ;
- DROP TABLE quartier ;

- DROP TABLE ville ;
- DROP SEQUENCE seq_quartier ;
- DROP SEQUENCE seq_immeuble ;
- DROP SEQUENCE seq_categorie ;
- DROP SEQUENCE seq_etat ;
- DROP SEQUENCE seq_personne ;
- DROP SEQUENCE seq_logement ;
- DROP SEQUENCE seq_bail ;
- DROP SEQUENCE seq_facture ;

L'ordre de la suppression des tables, correspond á l'ordre inverse de leur création.

4.2 Commandes SQL

Cette partie du travail nous a permis de tout tester au début : consultation, mises à jour, requêtes... avant d'intégrer le code SQL dans le code permettant de gérer l'interface.
Chaque commande SQL correspond bien à un besoin réel, soit pour la gestion interne de l'agence, soit pour répondre aux besoins de ses clients.

4.2.1 Consultation et mise à jour

Cette partie sera mieux gérer au niveau de l'interface. En effet, on se limitera dans cette partie à des commandes sql assez simples.

Exemples de consultation

```
PROMPT Affichage de la table donnee:
ACCEPT TABLE PROMPT 'entrez le nom de la table en MAJUSCULES: '
SELECT * FROM &TABLE;
```

⇒permet d'afficher une table donnée.

```
PROMPT Les attributs de la table:
ACCEPT TABLE PROMPT 'entrez le nom de la table en MAJUSCULES: '
select COLUMN_NAME from USER_TAB_COLUMNS where TABLE_NAME= '&TABLE';
```

⇒permet dáfficher les noms des colonnes d'une table donnée.

Exemples de mise à jour

- Mises à jour simples : elles figurent dans le fichier mise_a_jour_simple.sql .
  - Suppression d'un logement qui n'est plus disponible à la location :

```
PROMPT Ce logement est non destine a la location:
ACCEPT numlog PROMPT ' entrez le numero du logement: '
UPDATE Logement
SET disponibilite='supprime'
WHERE num_log='&numlog'
```

- Changement de l'adresse d'un propriétaire :

```
PROMPT Changer adresse propretaire:
ACCEPT nonprop PROMPT ' entrez le nom du proprietaire: '
ACCEPT prenomprop PROMPT ' entrez le prenom du proprietaire: '
ACCEPT adresse PROMPT ' entrez la nouvelle adresse du proprietaire: '
UPDATE Logement
SET adresse_prop='&adresse'
WHERE nss IN
(SELECT nss
FROM personne P
WHERE (P.nom = '&nomprop')
AND (P.prenom = '&prenomprop'));
```

- Mises à jour avec des triggers : elles figurent dans le fichier mise_a_jour_trigger.sql .

  Les triggers qu'on a mis en place se créent avec des erreurs de compilations, donc ils ne sont pas fonctionnels et on est pas parvenu à corriger les erreurs, enfin on ne voit pas trop l'erreur.
  - Le premier trigger trig_refection doit être déclenché suite au changement de la date_last_refection d'un logement, en effet suite à une tellee action, il faut opérer un changement au niveau de l'état du logement en question.
  - Le deuxième trigger trig_immeuble se déclenche suite à la modification de l'état de l'immeuble, en effet si un immeuble nést plus destiné à la location, tous les logements qui sont dedans ne sont plus disponibles aussi.

4.2.2 Requêtes

Requêtes concernant les villes

- Cette première requête permet de déterminer les quartiers d'une ville donnée.

```
PROMPT Les quartiers dans une ville donne:
ACCEPT nomville PROMPT 'entrez le nom de la ville: '
SELECT Q.nom_quartier
FROM quartier Q, ville V
WHERE (Q.code_postal=V.code_postal)
AND (V.nom_ville= '&nomville');
```

- Cette requête fournit le nombre de baux en cours pour une ville donnée. Pour réaliser cette requête, on se sert des vues logement_reel et immeuble_reel, en effet on ne s'intéresse qu'aux logements qui sont effectivement destinés à la location, car se sont ces derniers qui ont éventuellement des baux en cours. D'aileurs cette remarque concernera la quasi totalité des requêtes, en exceptant celles qui concernent l'historique. Dans un premier temps on a aboutit à la version suivante :

```
PROMPT Liste des baux en cours dans une ville donnee:
ACCEPT nomville PROMPT 'entrez le nom de la ville: '
ACCEPT date PROMPT 'entrez la date de recherche: '
Select num_bail
       FROM bail B, logement-reel L, immeuble-reel I, quartier Q, ville V
```

```
WHERE (B.num_logement = L.num_logement)
AND (B.date_fin > '&date')
AND (L.num_immeuble = I.num_immeuble)
AND (I.num_quartier = Q.num_quartier)
AND (Q.code_postal = V.code_postal)
AND (V.nom_ville = '&nomville');
```

Ensuite on a essayé de l'optimiser en terme de temps d'exécution et ceci en minimisant les jointures pour aboutir à une nouvelle version :

```
PROMPT Liste des baux en cours dans une ville donnee:
ACCEPT nomville PROMPT 'entrez le nom de la ville: '
ACCEPT date PROMPT 'entrez la date de recherche: '
Select num_bail
FROM bail B, logement_reel L
WHERE (B.date_fin > '&date')
AND (B.num_logement = L.num_logement)
AND L.num_logement IN
          (Select num_logement
         FROM logement_reel L1, immeuble_reel I
       WHERE (I.num_immeuble = L1.num_immeuble)
        AND I.num_immeuble IN
            (Select num_immeuble
        FROM immeuble_reel I1, quartier Q
        WHERE (I1.num_quartier = Q.num_quartier)
        AND Q.num_quartier IN
                 (Select num_quartier
               FROM quartier Q1, ville V
              WHERE (Q1.code_postal = V.code_postal)
AND (V.nom_ville = '&nomville'))));
```

Comparaison des deux requêtes en terme d'optimisation du temps d'exécution :

Operation	Selection	Jointure	Clonage	Correlation
Version1	6	4	0	0
Version2	9	1	3	0

⇒ La version2 est plus optimale.

Requêtes concernant les quartiers

Cette requête permet de déterminer les immeubles d'un quartier donné.

```
PROMPT Les immeubles dans un quartier donne:
ACCEPT nomquartier PROMPT 'entrez le nom du quartier: '
SELECT I.nom_immeuble
FROM immeuble_reel I, quartier Q
WHERE (Q.nom_quartier='&nomquartier')
AND (I.num_quartier=Q.num_quartier);
```

Requêtes concernant les immeubles

- Le nom dún immeuble étant unique comme mentionné au niveau des hypothèses prises, les logeements trouvés ont bien la même adresse et ils sont à louer.

```
PROMPT Liste des logements vides dans un immeuble donne:
ACCEPT nomimmeuble PROMPT'entrez le nom immeuble: '
Select num_logement
       FROM logement_reel L, immeuble_reel I
       WHERE (I.nom_immeuble = '&nomimmeuble')
       AND (I.num_immeuble = L.num_immeuble)
       AND (L.num_logement NOT IN
                  (select num_logement
     FROM bail B
     WHERE (B.date_fin > sysdate)));
```

– Ce genre de requêtes permet d'etablir des statistiques sur les immeubles...Pour effectuer cette requête on a considéré l'an 2007, ce qui nous a semblé assez rigoureux, puisqu' en réalité les charges ne peuvent quáugmenter et dans ce cas regarder tout l'historique de l'immeuble concerné et faire une moyenne fournira une valeur qui ne correspond pas à la réalité.

```
PROMPT Charges annuelles percues par un immeuble donne:
ACCEPT nomimmeuble PROMPT'entrez le nom immeuble: '
select sum(charges_mens) + charges AS total_charges
       FROM facture F , immeuble_reel
       where (nom_immeuble = '&nomimmeuble')
       AND ('01-jan-2007'<= F.date_payement)
       AND ( F.date_payement < '31-dec-2007')
       AND F.num_bail NOT IN
                (select num_bail
         FROM bail B
         where (B.date_fin < '01-jan-2007')
    AND (B.num_logement IN
              (Select num_logement
 FROM immeuble_reel I, logement_reel L
 where (I.nom_immeuble = '&nomimmeuble')
 AND (L.num_immeuble = I.num_immeuble))));
```

Requêtes concernant les logements

– L'adresse est constituée d'informations lues dans les tables logement, immeuble, quartier et ville. Des jointures successives sur ces différentes relations puis une séléction permettent de lire toutes les informations nécessaires. On projette enfin sur les attributs utiles. Pour cette requête, on a considéré les tables immeuble et logement et non pas les vues associées car on aura peut être besoin de l'adresse d'un logement même s'il n'est plus destiné à la location.

```
PROMPT Adresse du logement donne:
ACCEPT numlogement PROMPT 'entrez le numero du logement: '
SELECT L.num_logement, I.nom_immeuble, I.complement_adresse, Q.nom_quartier,
Q.code_postal, V.nom_ville
FROM immeuble I, logement L, quartier Q, ville V
WHERE (L.num_logement='&numlogement')
AND (I.num_immeuble=L.num_immeuble)
  AND (I.num_quartier=Q.num_quartier)
AND (V.code_postal=Q.code_postal);
```

– Un logement est considéré occupé sí l y a un bail en cours qui le concerne. On a effectué cette requête avec deux versions.

- Version 1 :

```
PROMPT Liste des logements occupes a une date donnee:
ACCEPT date PROMPT 'entrez la date de recherche: '
Select B.num_logement
        FROM bail B
WHERE (B.date_fin > '&date')
AND (B.date_signature < '&date');
```

- Version 2 :

```
PROMPT Liste des logements occupes a une date donnee:
ACCEPT date PROMPT 'entrez la date de recherche: '
Select num_logement
       FROM logement_reel L
       WHERE (L.num_logement NOT IN
                   (select B.num_logement
    FROM bail B

    WHERE (B.date_fin > '&date')));
```

- Comparaison des deux versions :

Operation	Selection	Jointure	Clonage	Correlation
Version1	2	0	0	0
Version2	2	0	1	0

⇒ La première requête est meilleure.

- On sintéresse aux logements non occupés et qui répondent à la condition voulue. On considère la vue pour effectuer cette requête, car elle doit crrespondre à la demande d'un client qui cherche un logement à louer. On a besoin de toutes les informations qui concernent ces logements.

```
PROMPT Liste des logements vides dont le loyer_moy est inferieur a un prix donne:
ACCEPT prix PROMPT 'Entrez le prix maximum que vous pourriez payer:'
SELECT * FROM logement_reel L
        WHERE num_logement NOT IN
(SELECT num_logement
FROM bail
WHERE date_fin > sysdate)
AND L.num_categorie IN
(SELECT C.num_categorie
FROM categorie C
WHERE loyer_moy < '&prix');
```

Requêtes concernant les personnes

- On suppose en fait que tout logement est occupé par les membres de la même famille, en effet notre agence s'intéresse au revenu total des occupants du logement sans distinguer les relations de parenté : par exemple si Mme MARTIN habite un appartement avec un ami et son fils Mr MARTIN pour des raisons de gestion l'agence s'intéresse aux revenus de tous les locataires du logement et non seulement à celui de Mme MARTIN et son fils. Ainsi cette requête sera plutôt effectuée relativement au logement concerné et non la famille et dans ce cas on supprimera dans la requête effectuée la selection :

```
P.Nom = '&nomfamille'

PROMPT Revenus totaux de famille X occupant un logement dans un immeuble Y:
```

```
ACCEPT nomfamille PROMPT 'entrez le nom de la famille: '
ACCEPT nomimmeuble PROMPT 'entrez le nom immeuble: '
Select sum(P.revenu)
       from personne P , bail B, definit_locataire D
       WHERE (B.num_bail = D.num_bail)
       AND (D.nss = P.nss)
       AND (P.Nom = '&nomfamille')
       AND (B.num_bail IN
                   (Select B.num_bail
          from immeuble I , logement L, bail B
          WHERE (I.nom_immeuble = '&nomimmeuble')
          AND (I.num_immeuble = L.num_immeuble)
          AND (L.num_logement = B.num_logement)
          AND (B.date_fin > SYSDATE)));
```

– Cette requête a des limites dues aux limites du modèle , en fait elle ne permet le calcul des loyers payés mensuellement par un locataire que si celui-ci a bien réglé les loyers du mois courant.

```
PROMPT Loyer paye mensuellement par un locataire donne:
ACCEPT nomloca PROMPT 'entrez le nom du locataire: '
ACCEPT prenomloca PROMPT 'entrez le prenom du locataire: '
Select avg(F.loyer + F.charges_mens) AS total_paye_mensuellement
FROM facture F, signe_par S, bail B, personne P
WHERE (S.nss = P.nss)
AND  (P.nom = '&nomloca')
    AND (P.prenom = '&prenomloca')
    AND (S.num_bail=B.num_bail)
AND(B.date_fin>sysdate)
AND (F.num_bail=B.num_bail);
```

– Cette requête calcule en moyenne le total mensuel collecté par Mr X qui possède Y logements. On s'intéresse à toutes les factures qui concernent ses logements et on effectue une moyenne.

```
PROMPT Total mensuel moyen des loyers collectes par un proprietaire donne:
ACCEPT nompro PROMPT 'entrez le nom du proprietaire: '
ACCEPT prenompro PROMPT 'entrez le prenom du proprietaire: '
Select (avg( loyer)*0.95) as Total_moyen
       FROM facture F, bail B
       WHERE (F.num_bail = b.num_bail)
       AND B.num_bail IN
                (Select num_bail
       FROM bail B1, logement_reel L
WHERE (B1.num_logement = L.num_logement)
AND L.num_logement IN
     (Select num_logement
      FROM logement_reel L1, personne P
      WHERE (L1.nss = p.nss)
      AND (p.nom = '&nompro')
      AND (P.prenom = '&prenompro')));
```

– On suppose que l'agence n'effectue de telles reqêtes qu'en fin du mois ce qui justifiera l'utilisation du sysdate.On considère qu'un payant a un retard de payement de loyer pour le mois courant si la date de payement de sa facture est supérieure au 10 du mois. La requête se compose de deux parties :

- Une première sélection qui permet de déterminer les numéros de factures qui ont été payées aprés les dix premiers jours du mois.
- puis, on récupère avec lópérateur ensembliste IN les numéros de baux qui sont dans le résultat de la première sélection. Des jointures et des projections permettent ensuite de lire les colonnes utiles.

Ainsi on a la requête suivante :

```
PROMPT Liste des occupants ayant un retard de loyer pour un immeuble donne:
ACCEPT nomimmeuble PROMPT'entrez le nom immeuble: '
SELECT L.num_logement, I.nom_immeuble, P.nom AS nom_locataire, P.prenom AS prenom_loctaire
FROM bail B, logement_reel L, immeuble_reel I, personne P, signe_par S
WHERE (B.num_logement=L.num_logement)
AND (I.num_immeuble=L.num_immeuble)
AND (I.nom_immeuble= '&nomimmeuble')
AND (S.num_bail=B.num_bail)
AND (S.nss = P.nss)
AND    B.num_bail  IN
(SELECT F.num_bail
                FROM facture F , bail B
                WHERE (B.num_bail=F.num_bail)
                AND   (F.date_payement >= (select trunc( sysdate , 'mon') + 9 FROM dual) ));
```

Requêtes concernant l'historique

- Cette requête est assez importante pour l'agence, en effet elle lui permet d'avoir une bonne idée sur la fidélisation de ses clients.

```
PROMPT Liste des baux renouves avec ou sans maintien des locataires
select DISTINCT B1.num_logement
       FROM bail B1, bail B2, signe_par S1 , signe_par S2
       WHERE (B1.num_logement = B2.num_logement)
       AND (B1.date_signature != B2.date_signature)
       AND (S1.num_bail = B1.num_bail)
       AND (S2.num_bail = B2.num_bail)
       AND (S1.type_sign != 'garant')
       AND (S2.type_sign != 'garant')
       AND (S1.nss= S2.nss);
```

- Cette requête permet de déterminer la trace des payements pour un logement donné.

```
PROMPT Historique des factures pour un logement donne entre 2 dates donnees:
ACCEPT date1 PROMPT 'entrez la premiere date de recherche: '
ACCEPT date2 PROMPT 'entrez la deuxieme date de recherche: '
ACCEPT numlogement PROMPT 'entrez le numero du logement: '
select num_facture, (charges_mens + loyer) AS Total_mensuel
       FROM facture F
       where ('&date1'<= F.date_payement)
       AND ( F.date_payement < '&date2')
       AND F.num_bail NOT IN
                (select num_bail
          FROM bail B
          where (B.date_fin < '&date1')
         AND (B.num_logement ='&numlogement'));
```

Chapitre 5

Interfaces

On a utilisé deux interfaces différentes à savoir l'interface *oracle-JDBC* et l'interface *MySQL-PHP*.

Ces deux interfaces ne sont pas complètes mais contiennent les techniques nécessaires pour lancer les requêtes qui manquent.

Le mode d'emploi de ces interfaces est détaillé dans le fichier *Lisez-moi* situé dans l'archive contenant les sources du projet.

5.1 Interface JDBC

Voir les figures 5.1 ; 5.2 et 5.3

5.2 Interface PHP

Voir les figures 5.4 et 5.5

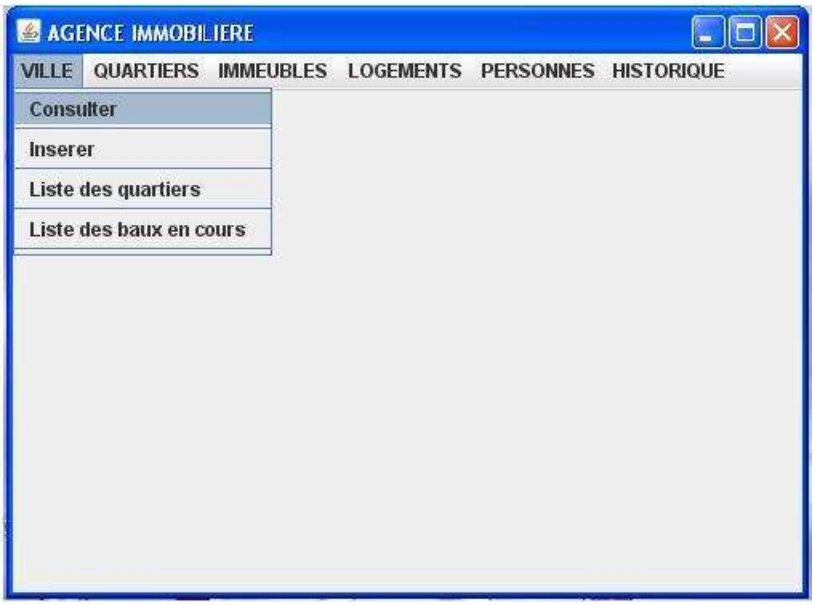

FIG. 5.1 – Fenetre de communication

FIG. 5.2 – Consultation des villes

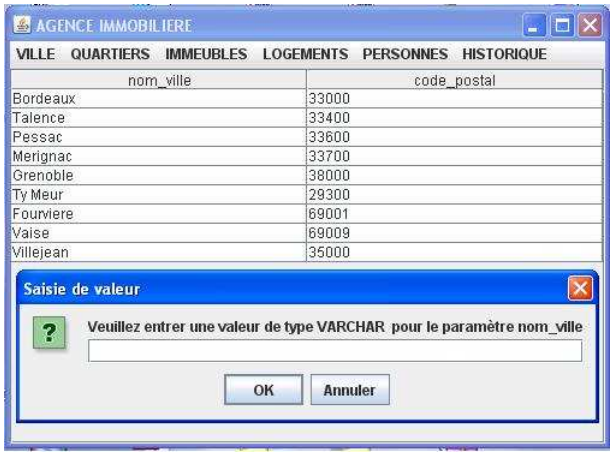

FIG. 5.3 – Requete

FIG. 5.4 – Page d'entrée

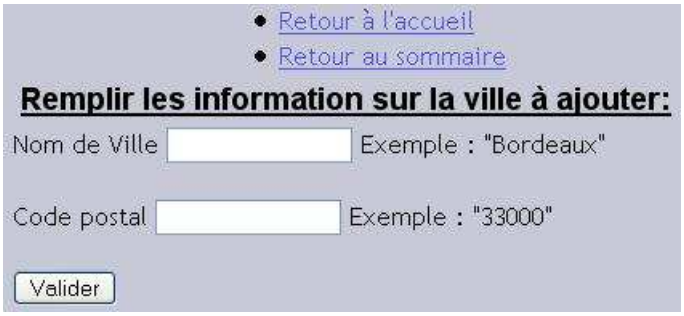

FIG. 5.5 – Ajouter une ville

Chapitre 6

Conclusion

Au cours de ce projet nous avons pu mettre en oeuvre les connaissances acquises au cours des séances de cours et de TD en vue de la création d'une base de données se voulant aussi proche de la réalité que possible.

Ce sujet, qui à première vue semblait simple,s'est avéré plus difficile qu'il n'y paraissait. En effet, les liens entre associations et entités qui semblaient être évidents, sont apparus plus compliqués, car il fallait tenir compte des redondances dans la base de données, qu'il faut éviter le plus souvent possible.

La majeure partie du temps que nous avons consacré au projet a été dans le but d'établir la modélisation des données.
Outre la mise en pratique des connaissances en SGBD, ce projet nous a permis de dcouvrir la manire dont pouvait se faire le lien entre le SGBD (Oracle) et le langage de programmation (*JDBC* et *php*) ainsi que de nous familiariser avec ces langages.

En somme, ce projet nous a permis de mieux comprendre la complexité qu'engendre la gestion des bases de données : ses buts, ses méthodes, ses problèmes et ses difficultés.