

Gestion de la base de données d'une agence immobilière

Systèmes de gestion de bases de données
Groupe 2

ABDELKHALEK RACHED BENMBARKA MOEZ

28 Novembre 2005

Table des matières

1	Cahier des charges	3
1.1	Analyse du sujet	3
1.2	Fonctionnalités	4
2	Conception	5
2.1	Modèle conceptuel	5
2.1.1	Entités	5
2.1.2	Associations	7
2.1.3	Contraintes	7
2.1.4	Schéma conceptuel	9
2.2	Modèle relationnel	9
2.2.1	Du conceptuel au relationnel	9
2.2.2	Normalisation	14
3	Réalisation	15
3.1	Création des tables	15
3.2	Requêtes SQL	15
4	Interface	20
4.1	Choix de l'environnement de développement	20
4.2	Manuel d'utilisation	20
4.2.1	Compilation et exécution	20
4.2.2	Menus	20
5	Exemples et tests	22
5.1	Ajout d'un immeuble	22
5.2	Affichage des loyers en retard	24
5.3	Classement des quartiers selon leurs revenus locatifs	24
A	Requêtes	27
A.1	Create.sql	27
A.2	Sequences.sql	29
B	Tests	30

Introduction

La conception de bases de données est un exercice délicat qui demande beaucoup de rigueur pour éviter de tomber dans des redondances et autres défauts d'intégrité.

Dans le cadre de ce projet, nous avons été invités à réfléchir sur la création d'une base de données d'une agence immobilière. C'est une application concrète qui nous confronte immédiatement aux différentes difficultés et interrogations qui peuvent se poser lors de la création de toute base de données.

Nous avons adopté au cours de ce projet une démarche en plusieurs étapes : l'analyse (spécification du cahier des charges : chapitre 1), la conception (modèle conceptuel et modèle relationnel : chapitre 2), la réalisation (chapitre 3) la mise en place de l'interface (chapitre 4) et d'un jeu de tests permettant de réaliser des requêtes aussi diversifiées que possible.

Chapitre 1

Cahier des charges

1.1 Analyse du sujet

Le projet tourne autour de la création de la base de données d'une agence immobilière permettant de faciliter la gestion du parc locatif de cette agence. Cette base devrait permettre d'organiser toutes les informations relatives aux clients de l'agence, qu'ils soient propriétaires ou simples locataires, ainsi qu'aux logements gérés par celle-ci (état, adresse, catégorie...). La base devra d'autre part permettre de faire le suivi quotidien des paiements (charges et loyers) perçus par l'agence ainsi que différents tri des données.

La base devra donc s'articuler autour de plusieurs axes :

- **Les immeubles** : sont caractérisés par un numéro unique, un nom, une date de construction, une date de dernière réfection, une adresse et un nombre total de logements.
- **Les logements** peuvent être occupés ou libres lors de la consultation. Dans le premier cas, le logement fait l'objet d'un bail. Chaque logement est par ailleurs caractérisé par une surface, un type, un état, un propriétaire, une adresse, une date de réfection, un loyer et des charges.
- **Les baux** : Définissent les conditions de location d'un logement, peuvent avoir plusieurs signataires et indiquent notamment : la durée du bail [3, 6 ou 9 ans], la caution versée [2 mois de loyer en général], le loyer mensuel, les charges mensuelles, la date de signature. Les baux établissent aussi le lien entre les logements d'une part et les occupants et les signataires d'autre part.
- **Les clients de l'agence** : Un client est identifié grâce à son numéro de sécurité social. Les clients sont classés en 3 catégories.
 - **Les occupants** : On dispose sur les occupants d'informations bien précises : leurs noms, prénoms, civilités, adresses, dates de naissance, professions, situations (salarié CDI, salarié CDD, demandeur d'emploi, retraité, indépendant, etc.), revenus et prestations.
 - **Les locataires** : Ce sont les signataires du bail qui prennent en charge un ou plusieurs occupants. Ils peuvent être eux mêmes occupants. On dispose, les concernant, des renseignements d'identités usuels ainsi que des spécifications des liens qui les lient aux personnes dont ils sont en charge.
 - **Les propriétaires** : On dispose, les concernant, des renseignements d'identités usuels (nom, adresse).
- **Les villes - Les quartiers** : Chaque ville se distingue par un nom et un code postal unique. Chaque quartier est désigné par son nom et se trouve dans une seule ville.

1.2 Fonctionnalités

L'application devra assurer plusieurs fonctionnalités passant par les opérations standard de consultation et insertion aux opérations de recherche d'informations particulières. Notamment,

1. Loyers et charges annuelles perçues par un immeuble.
2. Reconstitution d'une adresse à partir d'un numéro de logement.
3. Liste des logements occupés à une date donnée.
4. Liste des logements non occupés à la date courante.
5. Revenus totaux des occupants d'un logement.
6. Historique des paiements des loyers et des charges entre deux dates données.
7. La liste des logements et les payants correspondants ayant des retards de paiement de charges ou de loyers.
8. Liste des retards de paiement pour un immeuble donnée.
9. Le total des Loyers et charges payés mensuellement par un payant.
10. Liste de tous les quartiers classés par revenus locatifs décroissants.
11. Total des loyers collectés par un propriétaire annuellement.
12. Liste de tous les baux en cour dans une ville.

Chapitre 2

Conception

2.1 Modèle conceptuel

Dans cette partie on va présenter le modèle entités-association utilisé.

2.1.1 Entités

Les entités utilisées sont les suivantes :

1. **ville** :
Attributs :
 - code_postal :*NUMBER*, clé primaire
 - nom_ville :*CHAR(20)*
2. **quartier** :
Attributs :
 - num_quartier :*NUMBER*, clé primaire
 - code_postal :*NUMBER*
 - nom_quartier :*NUMBER*
3. **immeuble** :
Attributs :
 - num_immeuble :*NUMBER*, clé primaire
 - num_quartier :*NUMBER*
 - nom_immeuble :*CHAR(20)*
 - date_construction :*DATE*
 - date_refection :*DATE*
 - adresse :*CHAR(50)*
4. **logement** :
Attributs :
 - num_logement :*NUMBER*, clé primaire
 - num_immeuble :*NUMBER*
 - nss_proprietaire :*NUMBER*
 - categorie :*NUMBER*
 - etat :*NUMBER*
 - etage :*NUMBER*
 - porte :*NUMBER*
 - escalier :*NUMBER*
 - surface :*NUMBER*
 - loyer :*NUMBER*
 - charges :*NUMBER*

5. **personne :**

L'entité personne rassemble tous les comptes clients : propriétaire, occupant et payant. Bien qu'on n'ait pas besoin des mêmes informations pour ces trois types de clients, une seule entité nous permet de minimiser le risque de duplication de données si un seul client s'avère en même temps propriétaire, payant ou occupant.

Attributs :

- nss :*NUMBER*, clé primaire
- nom :*CHAR(20)*
- prenom :*CHAR(20)*
- civilite :*CHAR(3)*
- adresse :*CHAR(200)*
- naissance :*DATE*
- profession :*CHAR(50)*
- situation :*CHAR(50)*
- revenu :*NUMBER*
- prestations :*NUMBER*
- charges :*NUMBER*

6. **bail : Pourquoi une entité et pas une association ?** : Dans une première écriture du modèle conceptuel, le bail était représenté par deux associations : **occupé par** et **signé par** entre logement et personne. On s'est aperçu qu'avec ce schéma on sépare des informations sur un même objet (le bail) entre deux associations. Par exemple : la date de début de l'occupation du logement est identique à la date de signature du bail. Les occupants et les payants ont une relation avec le logement mais qui doit être définie par une seule entité qui est le bail.

Attributs :

- num_bail :*NUMBER*, clé primaire
- num_logement :*NUMBER*
- date_signature :*DATE*
- date_fin :*DATE*
- loyer :*NUMBER*
- charges :*NUMBER*
- caution :*NUMBER*

7. **loyer :**

Pourquoi une entité et pas simplement un attribut ? : Cela nous permet de reconstituer un historique des paiements de loyers. D'autre part, il faut noter la différence entre cette entité et l'attribut loyer de l'entité immeuble. Le dernier étant modifiable à tout moment, alors que l'entité loyer définit le paiement d'une somme fixée à la signature d'un bail. En effet, on a fait l'hypothèse que le loyer est fixé pour un logement donné dès la signature du bail correspondant.

Attributs :

- num_loyer :*NUMBER*, clé primaire
- num_bail :*NUMBER*
- date_paiement :*DATE*

8. **charge :**

Attributs :

- num_charge :*NUMBER*, clé primaire
- num_bail :*NUMBER*
- date_paiement :*DATE*

9. **etat :**

Représente les états possibles d'un logement : Bon, à rénover,...

Attributs :

- num_etat : *NUMBER*, clé primaire
- nom_etat : *CHAR(100)*

10. **catégorie :**

Représente les catégories possibles pour un logement :T1, T2,...

Attributs :

- num_categorie : *NUMBER*, clé primaire
- nom_categorie : *CHAR(10)*

2.1.2 Associations

Ces entités sont liées entre elles par des associations :

1. Un quartier **est situé** dans une ville.
2. Un immeuble **est localisé** dans un quartier.
3. Un logement **est contenu** dans un immeuble,
4. un logement **est dans l'état** etat et est **de catégorie** categorie.
5. Un logement **appartient** à une personne.
6. Un logement **est régi par** un bail.
7. Un bail est **signé par** une ou plusieurs personne(s).
8. Un bail **défini** un ou plusieurs occupant(s).
9. Un bail **fixe** un loyer et **précise** une charge.
10. une personne peut **avoir un lien familial** avec une ou plusieurs autres personnes.

Le fait d'avoir tous les comptes clients dans une seule entité nous permet de gérer les liens de parenté non seulement entre locataires et payants mais indifféremment entre toutes les personnes.

2.1.3 Contraintes

Cardinalités

Nous avons été amené à imposer certaines règles entre les entités :

- Une ville contient **0 ou N** quartier. A noter qu'une ville ne peut être ajoutée à la base que lors de l'ajout d'un nouveau immeuble dans un quartier situé dans la ville. Alors que la suppression de tous les immeubles d'une ville ne supprime pas automatiquement celle-ci. D'autre part un quartier est évidemment contenu dans **1** unique ville.
- Un quartier contient **0 ou N** immeubles, Un immeuble est localisé dans **1** seul quartier.
- Un immeuble contient **0 ou N** logements, un logement est contenu dans **1** seul immeuble.
- Un immeuble a un **1** seul type et **1** seule catégorie, alors qu'un type ou une catégorie peuvent être associés à **0 ou N** logement.
- Un logement appartient à une **1** seule personne, alors qu'une personne peut posséder **0 ou N** logements.
- Un logement peut être régi par **0 ou N** baux. Ceci nous permet d'avoir un historique des baux. Deux baux concernant le même logement ont nécessairement deux périodes de validité(entre date_signature et date_fin) disjointes. Un bail concerne **1** seul logement.
- Un bail peut être signé par **0 ou N** personnes. Une personne peut signer **0 ou N** baux. En même temps le même bail définit **1 ou N** occupants. Une même personne occupe **0 ou 1** logements.

- Un loyer ou charge enregistré concerne **1** seul bail. Un bail peut voir **0 ou N** loyers ou charges enregistrés avec son numéro.
- Enfin, une personne peut avoir **0,N** liens de parenté.

Contraintes de typage

Les types des attributs sont rapportés dans le schéma du modèle conceptuel. (voir figure 2.1)

Contraintes intra-relations

On ne citera ici que les contraintes concernant les attributs hors clés. Plus de précisions concernant les clés seront traitées ultérieurement après le passage au modèle relationnel.

1. Ville
 - nom_ville est **NOT NULL**
2. Quartier
 - nom_quartier est **NOT NULL**
3. Immeuble
 - nom_immeuble, date_construction, date_refection et adresse sont **NOT NULL**.
 - la date_refection est supérieur à la date_construction.
4. Personne
 - nom, prenom, naissance sont **NOT NULL**.
 - la civilité est dans 'Mr', 'Mll', 'Mme'.
 - La valeur par défaut pour profession et situation est 'Non communiquée'.
et 0 pour revenu et prestations
5. logement
 - categorie, etat, date_refection, porte, etage, surface, loyer et charges sont **NOT NULL**
 - Le triplet (porte, escalier, num_immeuble) est **UNIQUE**. Il permet de définir unique logement.
 - La valeur par défaut pour charges est 0.
6. bail
 - date_signature, date_fin, porte, loyer sont **NOT NULL**
 - La valeur par défaut pour charges et caution est 0.
 - date_signature est inférieur à date_fin.
7. loyer et charge
 - date_paiement est **NOT NULL**
8. etat
 - nom_etat est **NOT NULL**
9. categorie
 - nom_categorie est **NOT NULL**
10. famille
 - lien est **NOT NULL**

Contraintes inter-relations

Pour assurer la cohérence des données après insertions, modifications ou suppression il faut exiger certaines règles :

1. Un quartier ne peut exister que dans une ville. Donc la suppression d'une ville doit impliquer la suppression de tous les quartiers associés. La même règle s'applique pour les immeubles par rapport aux quartiers et les logements par rapport aux immeubles.

2. Pour pouvoir garder un historique des paiements, un bail n'est pas supprimé si le logement correspondant est supprimé.
3. De la même manière les traces des paiements de loyers et charges ne sont pas supprimées même si le bail est supprimé.
4. Par contre, la trace d'une signature ou occupation est supprimée si le bail correspondant l'est (la signature ou l'occupation n'a plus aucun sens puisqu'elle n'est plus liée à aucun logement ou paiement).

2.1.4 Schéma conceptuel

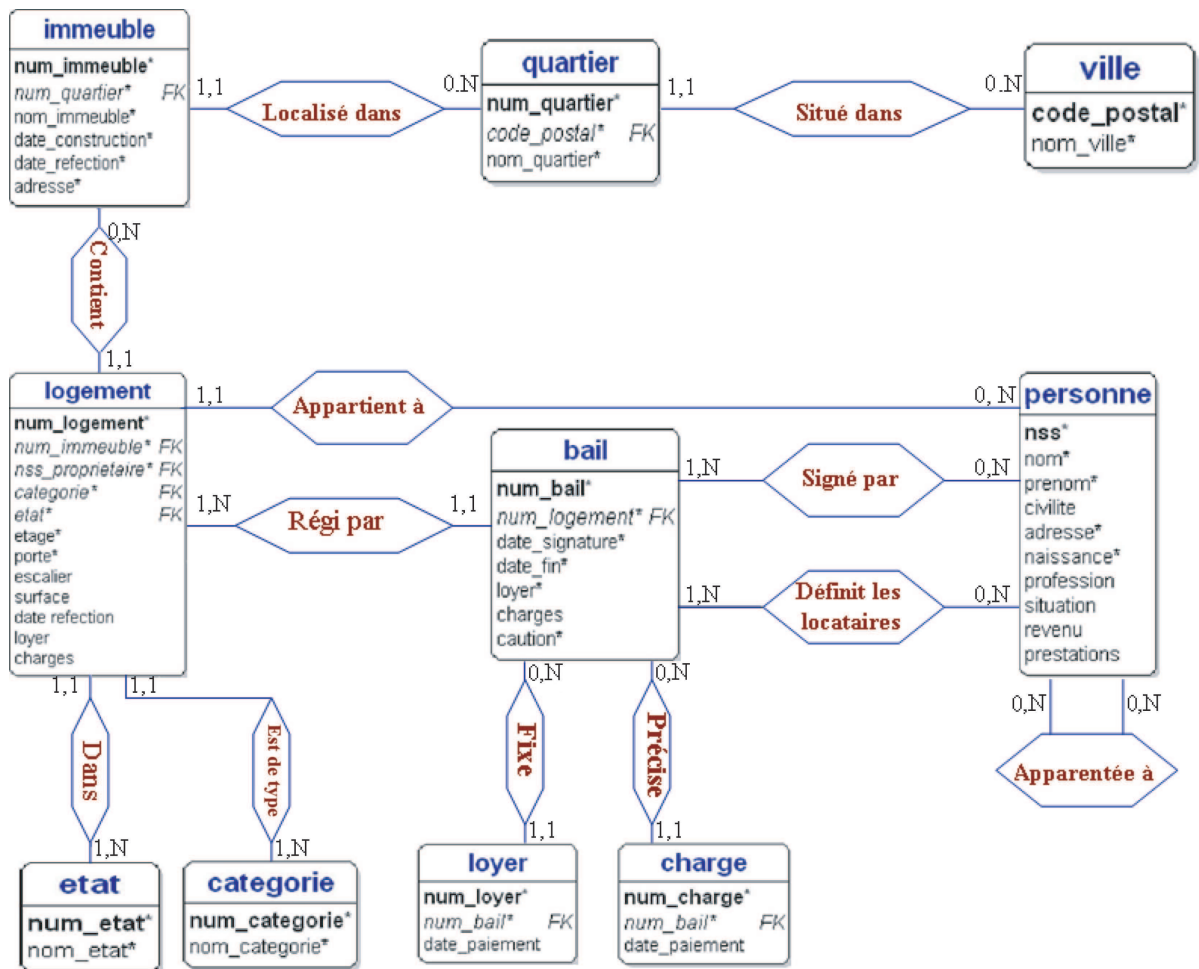


FIG. 2.1 – Schéma conceptuel

2.2 Modèle relationnel

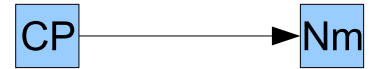
Le modèle conceptuel étant établie, le passage au modèle relationnel se fait en appliquant un ensemble de règles dépendant des cardinalités.

2.2.1 Du conceptuel au relationnel

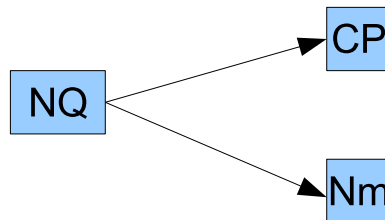
Les relations retenues ainsi que les dépendances fonctionnelles sont représentées sur les schémas suivants. Les clés primaires sont en gras, les clés étrangères en italique.

FIG. 2.2 – Schéma relationnel

Ville (CP: **code_postal**, Nm : nom_ville) :

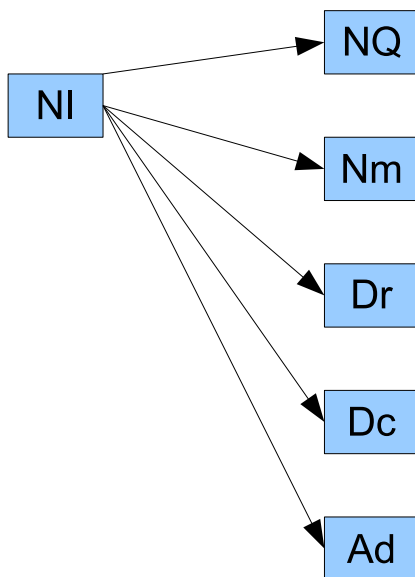


Quartier (NQ: **num_quartier**, CP : *code_postal*, Nm : nom_quartier) :



CP : clé étrangère qui référence la relation ville.
Il faut assurer que la suppression d'une ville supprime automatiquement les quartiers associés.

Immeuble (NI: **num_immeuble**, NQ : *num_quartier*, Nm : nom_immeuble, Dc: date_construction, Dr: date_refection, Ad: adresse) :



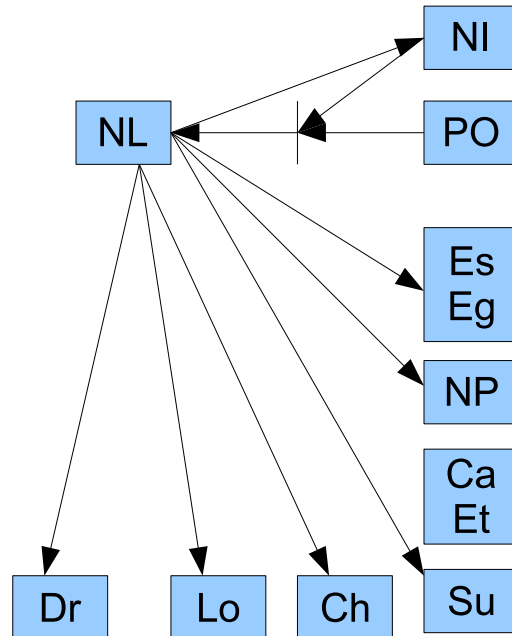
Adresse: complément pour l'adresse (ne contient pas le numéro de l'immeuble ni nom du quartier ni de la ville.)

Exemple: 1, Rue 'Avenue du Docteur SCHWEITZER'

NQ : clé étrangère qui référence la relation quartier.
Il faut assurer que la suppression d'un quartier supprime automatiquement les immeubles associés.

FIG. 2.3 – Schéma relationnel

logement (NL: num_logement, NI : num_immeuble, NP : nss_propriétaire, Ca: categorie, Et: etat, Eg: etagePo: porte, Es: escalier, Su: surface, Dr: date_refection,Lo: loyer, Ch: charges) :



CI : clé étrangère qui référence la relation immeuble. Il faut assurer que la suppression d'un immeuble supprime automatiquement les logements associés.

NP : clé étrangère qui référence la relation personne.

(NI, Po) : c'est une clé candidate. Elle ne peut pas être la clé primaire car sinon, on aura pas une deuxième forme normale puisque : PO -> ES.

personne (NP: nss, Nm : nom, Pr : prenom, Ci: civilite, Ad: adresse, Na: naissance, Po: profession, Si: situation, Re: revenu, Pe: prestations) :

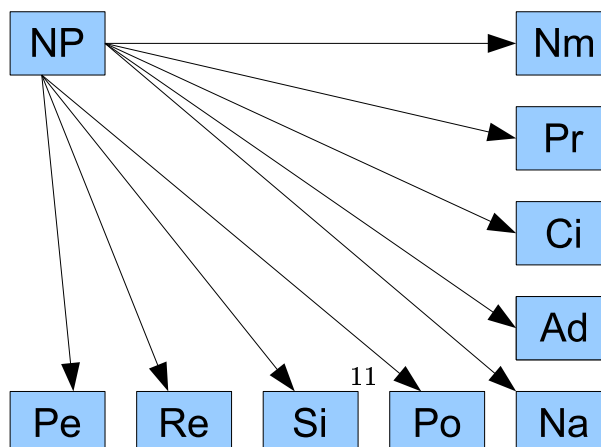
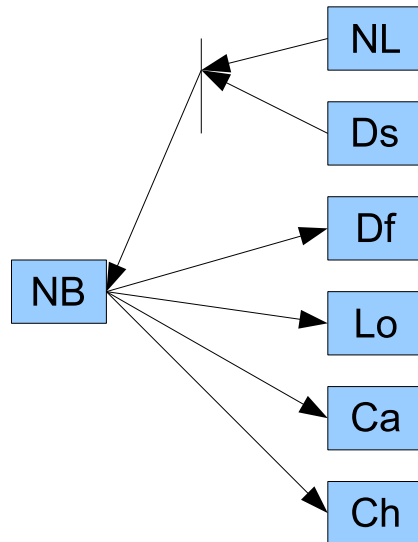


FIG. 2.4 – Schéma relationnel

bail (NB: num_bail, NL : num_logement, Ds : date_signature, Df: date_fin, Lo: loyer, Ch: charge, Ca: caution)



NL : Clé étrangère qui référence la relation logement.
Pour but de garder un historique des baux, la suppression d'un logement ne doit pas impliquer la suppression des baux correspondants.
(NL,Ds) : Clé candidate.
Pour des raisons de simplicité on a choisit de prendre NB comme la clé primaire.

categorie (NC: num_categorie, Nm : nom_categorie)

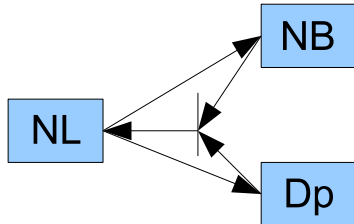


etat (NE: num_etat, Nm : nom_etat)



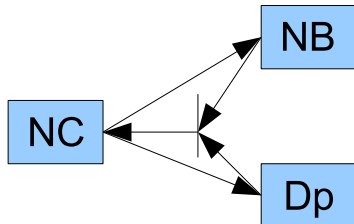
FIG. 2.5 – Schéma relationnel

loyer (NL: num_loyer, NB : num_bail, Dp: date_paiement)



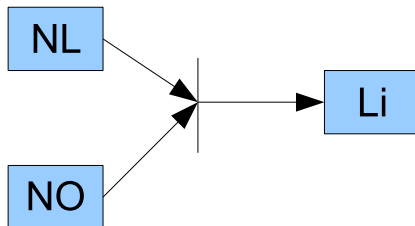
NB : Clé étrangère qui référence la relation bail.
 Pour but de garder un historique des paiements, la suppression d'un bail ne doit pas impliquer la suppression des loyers correspondants.
(NB,Dp): Clé candidate.
 Pour des raisons de simplicité on a choisit de prendre NL comme la clé primaire.

charge (NC: num_charge, NB : num_bail, Dp: date_paiement)



NC : Clé étrangère qui référence la relation bail.
(NB,Dp): Clé candidate.

famille (NL: nss1, NO: nss2, Li: lien)



occupation (NB: num_bail, NP: nss)

signature (NB: num_bail, NP: nss)

2.2.2 Normalisation

On va prouver ici que les relations retenus vérifient les hypothèses de la troisième forme normale.

Première forme normale :

- On voit dans les schémas relationnels que tous les attributs de toutes les relations sont atomiques(NUMBER, DATE, CHAR).
- De plus, chaque relation possède au moins une clé.

On en conclut : **Toutes les relation sont en première forme normale.**

Deuxième forme normale :

- On vient de prouver que toutes les relations sont en première forme normale.
- A part pour les relations : famille, occupation, signature, toutes les clés primaires sont atomiques, donc aucun attribut hors clé ne peut dépendre d'une partie stricte d'une clé.

famille : Le seul attribut hors clé : lien dépend de la totalité de la clé :(nss1,nss2)

occupation et signature : Elles ne possèdent pas d'attributs hors clé.

D'où : **Toutes les relations sont en deuxième forme normale.**

Troisième forme normale :

- On vient de montrer que toutes les relations sont en deuxième forme normale.
- D'après les schémas des dépendances fonctionnelles(figures précédentes), aucun attribut hors clé ne dépend d'un attribut ou groupe d'attributs hors clé.

D'où finalement : **Toutes les relations sont en troisième forme normale.**

Chapitre 3

Réalisation

3.1 Création des tables

La normalisation étant vérifiée on passe à la création des tables dans la base. Le fichier Create.sql crée toutes les tables.(Voir annexe :A.1) .

Une séquence a été créée pour chaque table possédant un entier comme identifiant pour assurer l'incrémement automatique lors de l'insertion. Ceci à l'exception de la table personne dont l'identifiant est le numéro de sécurité social qui doit donc être entré par l'utilisateur et la table ville dont l'identifiant est le code postal. Le fichier Sequences.sql est mis également en annexe(A.2).

3.2 Requêtes SQL

Commençons par les requêtes des opérations élémentaires : insertion, modification et suppression.

1. Insertion d'une ville

```
INSERT INTO ville VALUES ( 33000, 'Talence');
```

2. Suppression d'une ville

```
DELETE FROM ville WHERE code_postal=33000
```

3. Modification du nom d'une ville

```
UPDATE ville SET nom_ville='Bordeaux' WHERE code_postal=33000;
```

On va présenter ici les requêtes exécutants les opérations listées dans le paragraphe 1.2

1. Loyers et charges annuelles perçues par un immeuble :

L'aggrégateur SUM appliqué à la colonne bail de la relation loyer calcule la sommes des loyers des baux sélectionnés, ceux signés avant la date courante et dont la date_fin n'est pas encore atteinte. Une jointure avec la relation logement permet d'avoir les numéros des logements correspondants aux baux sélectionnés.

```
-- immeuble numéro 1
Select SUM (B.loyer)*12 AS loyers_annuel FROM bail B , logement L
WHERE (B.num_logement = L.num_logement)
AND (L.num_immeuble = 1)
AND (B.date_signature < sysdate)
```

```
AND (B.date_fin > sysdate);
```

```
Select SUM (B.charges)*12 FROM bail B , logement L
      WHERE (B.num_logement = L.num_logement)
      AND (L.num_immeuble = 1)
      AND (B.date_signature < sysdate)
      AND (B.date_fin > sysdate);
```

2. Reconstitution d'une adresse à partir d'un numéro de logement

L'adresse se constitue d'informations lues à partir des tables : logement, immeuble, quartier et ville. Des jointure successives sur ces différents relations puis une selection(num_logement=2) permettent de lire toutes les informations nécessaires. On projettent enfin sur les attributs utiles : etage, escalier, porte, nom_immeuble, nom_quartier, code_postal et nom_ville.

```
-- logement numéro 2
SELECT L.etage, L.escalier, L.porte, I.nom_immeuble, Q.nom_quartier,
Q.code_postal, V.nom_ville
FROM immeuble I, logement L, quartier Q, ville V
WHERE (L.num_logement=2)
AND (I.num_immeuble=L.num_immeuble)
AND (I.num_quartier=Q.num_quartier)
AND (V.code_postal=Q.code_postal);
```

3. Liste des logements occupés à une date donnée

Un logement est considéré occupé s'il ya un bail le concernant dans la table bail.

La liste est récupérée simplement par une selection sur la relation logement puis une projection sur num_logement.

```
-- date: 27-nov-05
SELECT B.num_logement FROM bail B
WHERE ( B.date_fin > '27-nov-05' )
AND (B.date_signature < '27-nov-05' );
```

4. Liste des logements non occupés à la date courante

Il suffit de prendre le complémentaire de la sélection précédente(avec date=la date courante).

```
SELECT B.num_logement, I.num_immeuble, I.nom_immeuble
FROM bail B, Logement L, immeuble I
WHERE (L.num_logement=B.num_logement)
AND (L.num_immeuble=I.num_immeuble)
AND (B.num_bail NOT IN (SELECT B.num_bail
                        FROM bail B
                        WHERE ( B.date_fin > sysdate )
                        AND (B.date_signature < sysdate ))));
```

5. revenus totaux des occupants d'un logement

```
Select sum (P.revenu) as "Revenus" , sum (P.prestations) AS "Prestations " , sum(p.prestatio
      FROM personne P, occupation O, bail B
      WHERE (P.nss = O.nss)
      AND (O.num_bail = B.num_bail)
      AND (B.num_logement = 2);
```

6. Historique des paiements des loyers et des charges entre deux dates

Les traces des paiements des loyers sont enregistrés dans la relation loyer. Une sélection sur cette relation(date1<date_paiement<date2), puis des jointures successives sur les tables : logement, immeuble et bail permettent d'avoir la liste avec les information utiles en projetant sur date_paiement, num_logement, nom_immeuble et loyer.

```
-- date1=01-jan-96, date2=01-jan-05

SELECT L0.date_paiement, L.num_logement, I.nom_immeuble, B.loyer
FROM loyer L0, logement L, immeuble I, bail B
WHERE (L0.date_paiement>'01-jan-96')
AND (L0.date_paiement<'01-jan-05')
AND (L0.num_bail = B.num_bail)
AND (B.num_logement = L.num_logement)
AND (L.num_immeuble=I.num_immeuble);
```

```
-- Historique payement charge entre date1 et date2
```

```
SELECT C.date_paiement, L.num_logement, I.nom_immeuble, B.loyer
FROM charge C, logement L, immeuble I, bail B
WHERE (C.date_paiement>'01-jan-96')
AND (C.date_paiement<'01-jan-05')
AND (C.num_bail = B.num_bail)
AND (B.num_logement = L.num_logement)
AND (L.num_immeuble=I.num_immeuble);
```

7. La liste des logements et les payants correspondants ayant des retards de paiement de charges ou de loyers

Un payant a un retard de paiement de loyer si aucun loyer avec son numéro de logement n'est enregistré dans la table loyer avec une date de paiement supérieure au premier du mois de la date courante.

Cette date est récupéré grâce à la selection suivante :

```
select round(sysdate,'Mon') from dual
```

La requête se compose de deux parties. Une première selection :

```
SELECT L0.num_bail
FROM charge L0, bail B
WHERE (B.num_bail=L0.num_bail)
AND (L0.date_paiement>= (select trunc(sysdate,'Mon') from dual));
```

qui permet de selectionner les numéros de baux qui ont enregistré un paiement après le premier du mois courant.

La deuxième selection permet avec l'opérateur ensembliste **NOT IN** de récupérer les numéros de baux qui ne sont pas dans le résultat de la première sélection. Des jointures et des projections permettent ensuite de lire les colonnes utiles.

```
SELECT B.num_bail, L.num_logement, I.nom_immeuble, S.nss, P.nom AS payant
FROM bail B, logement L, immeuble I, signature S, personne P
WHERE (B.num_logement=L.num_logement)
AND (I.num_immeuble=L.num_immeuble)
AND (S.num_bail=B.num_bail)
AND (P.nss=S.nss)
AND B.num_bail NOT IN
```

```
(SELECT LO.num_bail
FROM charge LO, bail B
WHERE (B.num_bail=LO.num_bail)
AND (LO.date_paiement >= (select trunc(sysdate,'Mon') from dual)));
```

```
-- Payants/occupants ayant retards de loyer pour un immeuble(ici num_immeuble=1) r
SELECT L.num_logement, I.nom_immeuble, P1.nom AS payant, P2.nom AS occupant
FROM bail B, logement L, immeuble I, signature S, personne P1, personne P2, occupation O
WHERE (B.num_logement=L.num_logement)
AND (I.num_immeuble=L.num_immeuble)
AND (I.num_immeuble=1)
AND (S.num_bail=B.num_bail)
AND (P1.nss=S.nss)
AND (O.num_bail=B.num_bail)
AND (O.nss=P2.nss)
AND B.num_bail NOT IN
(SELECT LO.num_bail
FROM loyer LO, bail B
WHERE (B.num_bail=LO.num_bail)
AND (LO.date_paiement >= (select trunc(sysdate,'Mon') from dual)));
```

8. Liste des retards de paiement pour un immeuble donnée

```
-- num_immeuble=1

SELECT L.num_logement, I.nom_immeuble, P1.nom AS payant, P2.nom AS occupant
FROM bail B, logement L, immeuble I, signature S, personne P1, personne P2, occupation O
WHERE (B.num_logement=L.num_logement)
AND (I.num_immeuble=L.num_immeuble)
AND (I.num_immeuble=1)
AND (S.num_bail=B.num_bail)
AND (P1.nss=S.nss)
AND (O.num_bail=B.num_bail)
AND (O.nss=P2.nss)
AND B.num_bail NOT IN
(SELECT LO.num_bail
FROM loyer LO, bail B
WHERE (B.num_bail=LO.num_bail)
AND (LO.date_paiement >= (select trunc(sysdate,'Mon') from dual)));
```

9. Le total des loyers et charges payés mensuellement par un payant

```
Select SUM (B.loyer) AS loyers, SUM(B.charges) AS charges, SUM(B.loyer) + SUM(B.charges) AS total
FROM bail B, signature S
WHERE (S.nss=30071983)
AND (S.num_bail=B.num_bail)
AND (B.date_fin>sysdate);
```

10. Liste de tous les quartiers classés par revenus locatifs décroissants

Le calcul se fait en groupant les logements par leurs quartiers(avec **GROUP BY Q.nom_quartier, Q.num_quartier**) et en calculant pour chaque groupe

le total des loyers correspondants aux baux en cour. L'ordre décroissant est assuré par :**ORDER BY SUM (B.loyer) DESC** ;

```
Select Q.nom_quartier ,Q.num_quartier, SUM (B.loyer) as revenus_locatifs_mensuels FROM bail B, logement L, immeuble I
WHERE (B.num_logement = L.num_logement)
AND (L.num_immeuble = I.num_immeuble)
AND (I.num_quartier = Q.num_quartier)
AND (B.date_signature < sysdate)
AND (B.date_fin > sysdate)
GROUP BY Q.nom_quartier, Q.num_quartier
ORDER BY SUM (B.loyer) DESC ;
```

11. Total des loyers collectés par un propriétaire annuellement

```
SELECT SUM(B.loyer)*12 AS total_loyers_annuel
FROM bail B, logement L
WHERE (B.num_logement = L.num_logement)
AND (L.nss_proprietaire = 26121981)
AND (B.date_fin > sysdate);
```

12. Liste de tous les baux en cour dans une ville

```
SELECT B.num_bail, B.num_logement, I.nom_immeuble, Q.nom_quartier, V.nom_ville
FROM ville V, quartier Q, immeuble I, logement L, bail B
WHERE (V.code_postal=33000)
AND (V.code_postal=Q.code_postal)
AND (Q.num_quartier=I.num_quartier)
AND (I.num_immeuble=L.num_immeuble)
AND (B.num_logement=L.num_logement)
AND (B.date_fin > sysdate);
```

Chapitre 4

Interface

4.1 Choix de l'environnement de développement

- Langage de programmation : JDBC
- SQL : Oracle

4.2 Manuel d'utilisation

4.2.1 Compilation et exécution

Les codes sources fournis sont composés d'un package **agence** et d'un programme principal **Menu**.

Avant la compilation, il est nécessaire d'initialiser l'environnement. Un script **init_env** est fourni avec les codes sources. L'initialisation se fait avec :

```
source init_env
```

Ensuite, la compilation se fait avec :

```
javac Menu.java
```

puis l'exécution :

```
java Menu
```

4.2.2 Menus

A l'exécution le menu principal apparaît :

```
Menu Principal
*****
1-Gestion villes
2-Gestion quartiers
3-Gestion immeubles
4-Gestion logements
5-Gestion clients
6-Gestion contrats
7-Gestion paiements
8-Désactiver le mode verbose
9-Quitter
```

Les fonctionnalités offertes par l'interface sont groupées par objet.
Le choix se fait en tapant le numéro correspondant à la fonction voulue.
Par exemple, en tapant 7, le sous menu suivant s'affiche :

Menu gestion Paiements

- 1-Ajouter un paiement de loyer
- 2-Ajouter un paiement de charge
- 3-Afficher historique des paiements des loyers
- 4-Afficher historique des paiements des charges
- 5-Afficher les retards de loyers
- 6-Afficher les retards de charges
- 7-Retour

Pour revenir au menu principal il suffit de taper le numéro précédent 'Retour'.

A travers les fonctions des sous menu, il est possible d'ajouter, supprimer un enregistrement(ville, quartier,...) ou simplement afficher les enregistrements. Il est également possible d'exécuter les différentes requêtes vues précédemment.
Des exemples d'utilisation sont donnés dans la partie 5.

Chapitre 5

Exemples et tests

Nous détaillons dans cette partie le fonctionnement du programme sur un exemple. L'exemple de départ utilisé pour les différentes requêtes qui vont suivre est détaillé en annexe (B. Il se compose de deux villes (Bordeaux, Toulouse), trois quartiers (St Michel, La Bastide, Bacalan), un immeuble dans chaque quartier et six logements (tous occupés au départ) répartis sur les trois immeubles avec différentes configurations familiales. Pour charger ce jeux de tests il suffit d'exécuter les fichiers ¹ init.sql (effacer toutes les tables et les s'quences et les recréer à nouveau) puis test.sql (ajout des données dans les tables).

Nous faisons le choix de détailler les opérations suivantes :

- Ajout d'un immeuble : Représentative de toutes les opérations d'ajout.
- Requêtes particulières ;
 - Affichage de loyers en retard à la date courante.
 - Classement des quartiers selon leurs revenus locatifs.

5.1 Ajout d'un immeuble

La base initiale contient 3 immeubles. Nous allons ajouter un quatrième immeuble dans un nouveau quartier de Toulouse. Pour cela, à partir du menu principal on entre 3.

Menu Principal

```
1-Gestion villes
2-Gestion quartiers
3-Gestion immeubles
4-Gestion logements
5-Gestion clients
6-Gestion contrats
7-Gestion paiements
8-Activer mode verbose.
9-Quitter
```

apparaît alors le menu suivant :

Menu gestion immeubles

```
1-Ajouter un immeuble
2-Supprimer un immeuble
3-Afficher la liste des immeubles
```

¹Ces fichiers sont dans le répertoire Test

- 4-Total charges perçus pour un immeuble
- 5-Total loyers perçus pour un immeuble
- 6-Retards de charges dans un immeuble
- 7-Retards de loyer dans un immeuble
- 8-Retour

En entrant 1 à partir du menu puis les informations suivantes ;

Sélectionnez le quartier(Entrer un nouveau numéro pour créer un nouveau quartier):

NUM_QUARTIER	NOM_QUARTIER	CODE_POSTAL
1	St Michel	33000
2	La Bastide	33000
3	Bacalan	33000
4		

Le quartier 4 n'existe pas. Voulez vous créer un nouveau quartier? (0:non/1:oui)

1

Ajout d'un nouveau quartier.

Nom du quartier:

St Sernin

Choix de la ville :(tapez un nouveau code postal pour ajouter une nouvelle ville)

CODE_POSTAL	NOM_VILLE
33000	Bordeaux
31000	Toulouse

Code postal de la ville:

31000

Quartier St Sernin créé !

Nom de l'immeuble:

ADA

Date de construction :

01-dec-1996

date dernière réfection :

12-mar-2003

Complément adresse:

12 Rue 20 Mars

immeuble ajouté!

5.2 Affichage des loyers en retard

Pour afficher les loyers en retard à la date courante, on suit à partir du menu principal les étapes suivantes :

Menu Principal

- 1-Gestion villes
- 2-Gestion quartiers
- 3-Gestion immeubles
- 4-Gestion logements
- 5-Gestion clients
- 6-Gestion contrats
- 7-Gestion paiements
- 8-Activer mode verbose
- 9-Quitter

7

Menu gestion Paiements

- 1-Ajouter un paiement de loyer
- 2-Ajouter un paiement de charge
- 3-Afficher historique des paiements des loyers
- 4-Afficher historique des paiements des charges
- 5-Afficher les retards de loyers
- 6-Afficher les retards de charges
- 7-Retour

5

Tous les retards de paiement de loyer:

NUM_BAIL-----NUM_LOGEMENT-----NOM_IMMEUBLE-----NSS-----NOM

5 -----5 -----Pascal -----12031945-----Gérard

5.3 Classement des quartiers selon leurs revenus locatifs

Permet d'avoir un classement par ordre décroissant des quartiers de la base selon leurs revenus locatifs. Les quartiers listés sont ceux qui ont effectivement des revenus mensuels (des baux en cours).

Menu Principal

- 1-Gestion villes
- 2-Gestion quartiers
- 3-Gestion immeubles
- 4-Gestion logements
- 5-Gestion clients
- 6-Gestion contrats
- 7-Gestion paiements
- 8-Activer mode verbose

9-Quitter
2

Menu gestion quartier

1-Supprimer un quartier
2-Afficher la liste des quartiers
3-Classement selon revenus décroissant
4-Retour
3

NOM_QUARTIER-----NUM_QUARTIER-----REVENUS_LOCATIFS_MENSUELS

St Michel -----1 ----- 1450-----

Bacalan -----3 ----- 620-----

La Bastide -----2 ----- 450-----

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.

Outre la mise en pratique des connaissances en SGBD, ce projet nous a permis de découvrir la manière dont pouvait se faire le lien entre le SGBD (Oracle) et le langage de programmation (JDBC) ainsi que de nous familiariser avec ce langage.

Annexe A

Requêtes

A.1 Create.sql

```
CREATE TABLE ville(  
    code_postal INT NOT NULL PRIMARY KEY ,  
    nom_ville CHAR(20) NOT NULL);  
  
CREATE TABLE quartier (  
    num_quartier INT NOT NULL PRIMARY KEY ,  
    code_postal INT NOT NULL REFERENCES ville (code_postal) ON DELETE CASCADE,  
    nom_quartier CHAR(20) NOT NULL);  
  
CREATE TABLE immeuble (  
    num_immeuble INT NOT NULL PRIMARY KEY,  
    num_quartier INT NOT NULL REFERENCES quartier (num_quartier) ON DELETE CASCADE,  
    nom_immeuble CHAR(20) NOT NULL,  
    date_construction DATE NOT NULL,  
    date_refection DATE NOT NULL,  
    adresse char(50) NOT NULL,  
    check (date_refection > date_construction));  
  
CREATE TABLE personne(  
    nss INT NOT NULL PRIMARY KEY ,  
    nom CHAR(20) NOT NULL,  
    prenom CHAR(20) NOT NULL,  
    civilite CHAR(3) CHECK (civilite in ('Mr', 'Mll', 'Mme')),  
    adresse CHAR(200),  
    naissance DATE NOT NULL,  
    profession CHAR(50) DEFAULT 'Non communiquée',  
    situation CHAR(50) DEFAULT 'Non communiquée',  
    revenu FLOAT DEFAULT 0,  
    prestations FLOAT DEFAULT 0);  
  
CREATE TABLE categorie(  
    num_categorie INT NOT NULL PRIMARY KEY,  
    nom_categorie CHAR(10) NOT NULL);  
  
CREATE TABLE etat(  
    num_etat INT NOT NULL PRIMARY KEY,
```

```
nom_etat CHAR(100) NOT NULL);
```

```
CREATE TABLE logement(  
num_logement INT NOT NULL PRIMARY KEY,  
num_immeuble INT NOT NULL REFERENCES immeuble (num_immeuble) ON DELETE CASCADE,  
nss_proprietaire INT NOT NULL REFERENCES personne (nss) ON DELETE CASCADE,  
categorie INT NOT NULL REFERENCES categorie (num_categorie),  
etat INT NOT NULL REFERENCES etat (num_etat),  
    etage INT NOT NULL,  
porte INT NOT NULL,  
escalier CHAR,  
surface FLOAT NOT NULL,  
date_refection DATE NOT NULL,  
loyer FLOAT NOT NULL,  
charges FLOAT NOT NULL DEFAULT 0,  
UNIQUE (porte, escalier,num_immeuble));
```

```
CREATE TABLE bail(  
num_bail INT NOT NULL PRIMARY KEY,  
num_logement INT NOT NULL REFERENCES logement (num_logement),  
date_signature DATE NOT NULL,  
date_fin DATE NOT NULL ,  
    loyer FLOAT NOT NULL,  
charges FLOAT,  
caution FLOAT DEFAULT 0,  
check ( date_signature < date_fin));
```

```
CREATE TABLE occupation(  
nss INT NOT NULL REFERENCES personne (nss),  
num_bail INT NOT NULL REFERENCES bail (num_bail) ON DELETE CASCADE,  
PRIMARY KEY(nss,num_bail));
```

```
CREATE TABLE signature(  
num_bail INT NOT NULL REFERENCES bail (num_bail) ON DELETE CASCADE,  
nss INT NOT NULL REFERENCES personne (nss) ,  
PRIMARY KEY(num_bail,nss));
```

```
CREATE TABLE loyer(  
num_loyer INT NOT NULL PRIMARY KEY,  
num_bail INT NOT NULL REFERENCES bail (num_bail),  
date_paiement DATE NOT NULL);
```

```
CREATE TABLE charge(  
num_charge INT NOT NULL PRIMARY KEY,  
num_bail INT NOT NULL REFERENCES bail (num_bail),  
date_paiement DATE NOT NULL);
```

```
CREATE TABLE famille(  
nss_loc INT NOT NULL REFERENCES personne (nss) ON DELETE CASCADE,  
nss_occ INT NOT NULL REFERENCES personne (nss) ON DELETE CASCADE,  
lien CHAR(20) NOT NULL,
```

```
PRIMARY KEY(nss_loc,nss_occ));
```

A.2 Sequences.sql

```
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_logement START WITH 1 INCREMENT BY 1;  
CREATE SEQUENCE seq_bail START WITH 1 INCREMENT BY 1;  
CREATE SEQUENCE seq_loyer START WITH 1 INCREMENT BY 1;  
CREATE SEQUENCE seq_charge START WITH 1 INCREMENT BY 1;
```

Annexe B

Tests

Nous détaillons dans les tableaux suivants, les données utilisées pour les tests effectués dans la partie Exemples et tests. Ces tableaux sont générés automatiquement et mis en ligne à partir de la base par un scripte php.

CODE_POSTAL	NOM_VILLE
33000	Bordeaux
31000	Toulouse

NUM_QUARTIER	CODE_POSTAL	NOM_QUARTIER
1	33000	St Michel
2	33000	La Bastide
3	33000	Bacalan

FIG. B.1 – Villes et quartiers

NSS	NOM	PRENOM	CIVILITE	ADRESSE	NAISSANCE	PROFESSION	SITUATION	REVENU	PRESTATIONS
30071983	Ben Slimane	Mehdi	Mr	34 Cours Gambetta Bordeaux	30-SEP-54			0	0
21041944	Dupont	Julien	Mr	110 rue ste Catherine Bordeaux	14-MAY-44			0	0
24445535	Laporte	Frédéric	Mr	32 rue Lamartine	30-SEP-62			0	0
40071666	Duval	Pierre	Mr	23 Rue Mouffard Paris 75014	30-DEC-44			0	0
45121985	Traoré	Mamadou	Mr	14 Quai des Chartrons Bdx 33000	21-JAN-45			0	0
54/68904	Duval	Laurent	Mr		30-SEP-83	Etudiant		0	133
45112385	Traore	Naiba	Mll		08-JAN-81	Etudiant		0	233
2312/1983	Lee	Kim	Mll		30-OCT-77	Professeur	CDI	1200	133
44551983	Lee	Xuan	Mme		07-OCT-39	sans emploi		0	390
26121981	Slim	Gnira	Mr		15-JAN-82	Etudiant		0	133
12031945	Gérard	Philippe	Mr		30-SEP-69	Ingenieur	CDI	2200	500
26121221	Gérard	Odile	Mme		10-JAN-71	Sans emploi		0	233
12091993	Gérard	Thomas	Mr		12-SEP-93	Eleve		0	0
14324567	Martin	Laurent	Mr		30-SEP-84	Etudiant		0	233

FIG. B.2 – Personnes

NUM_IMMEUBLE	NUM_QUARTIER	NOM_IMMEUBLE	DATE_CONSTRUCTION	DATE_REFECTION	ADRESSE
1	1	Java	12-OCT-93	12-OCT-03	13 Rue du peuple
2	2	Cobol	14-SEP-01	12-OCT-04	Rue du bourget
3	3	Pascal	14-SEP-01	12-OCT-03	Rue des lacs

NUM_LOGEMENT	NUM_IMMEUBLE	NSS_PROPRIETAIRE	CATEGORIE	ETA	TAGE	PORTIE	ESCALIER	SURFACH	DATE_REFECTION	LOYER	CHARGES
1	1	30071983	3	1	2	204	A	60	30-SEP-99	450	140
2	1	30071983	1	2	1	103	A	25	12-OCT-98	250	80
3	1	30071983	1	1	4	106	C	22	01-JAN-99	270	90
4	2	21041944	3	1	3	312		64	30-MAR-02	450	130
5	3	24445535	2	1	0	6		45	15-SEP-01	320	110
6	3	24445535	1	1	3	301		27	10-NOV-04	300	60

NUM_BAIL	NUM_LOGEMENT	DATE_SIGNATURE	DATE_FIN	LOYER	CHARGES	CAUTION
1	1	01-JAN-00	01-JAN-09	450	140	900
2	2	01-SEP-04	30-AUG-07	500	160	500
3	3	01-NOV-04	30-OCT-07	500	160	500
4	4	01-SEP-04	30-AUG-07	450	130	900
5	5	01-NOV-04	30-OCT-07	320	110	640
6	6	01-JUN-04	31-MAY-07	300	60	600

FIG. B.3 – Immeubles,logements et baux