Introduction

Le but de ce projet était de créer une application complète d’une organisation en se basant simplement sur une liste de requis donnés. Nous avons dû concevoir une base de données SQL qui respectait les demandes de l’organisation. Nous avons également dû créer une série de requête SQL qui permettait d’accomplir diverses actions importantes. Le projet devait être divisé en un front-end en utilisant Angular et un back-end en NodeJS. L’application permet à des clients d’entré et modifier de l’information présent dans la base de données.

**Dépendance fonctionnelle**

* **Clinique** (noClinique, nom, rue, ville, province, codePostale, numTéléphone, numTélécopieur)

DF noClinique -> nom, rue, ville, province, codePostale, numTéléphone, numTélécopieur

DF codePostale -> rue, ville, province

* **ProprietaireAnimal** (noProprietaire, noClinique, nom, prenom, rue, ville, province, codePostale, numTéléphone)

DF noProprietaire,noClinique -> , nom, prenom, rue, ville, province, codePostale, numTéléphone

DF codePostale -> rue, ville, province

* **Animal** (noAnimal, noClinique, noProprietaire, nom, type, espece, taille, poids, description, dateNaissance, dateInscription, etatActuel)

DF noAnimal, noClinique -> , noProprietaire, nom, type, espece, taille, poids, description, dateNaissance, dateInscription, etatActuel  
DF espece->type

* **Employe** (noEmploye, noClinique, nom, prenom, rue, ville, province, codePostale, numTelephone, dateNaissance, sexe, NAS, estGestionnaire)

DF noEmploye, noClinique-> , nom, prenom, adresse, numTelephone, dateNaissance, sexe, NAS, estGestionnaire

DF NAS -> noEmploye, noClinique, nom, prenom, adresse, numTelephone, dateNaissance, sexe, estGestionnaire

DF codePostale -> rue, ville, province

* **Veterinaire** (noEmploye, estEnService)

DF noEmploye->estEnService

* **Examen** (noExamen, noClinique, noAnimal, dateExamen, heureExamen, noVeterinaire, description)

DF noExamen, noClinique -> noAnimal, dateExamen, heureExamen, noVeterinaire, description

* **Traitement** (noTraitement, description, cout)

DF noTraitement ->description, cout

* **Facture** (noFacture, estPaye, noExamen, noClinique, moyenPayement, date, coutTotal)

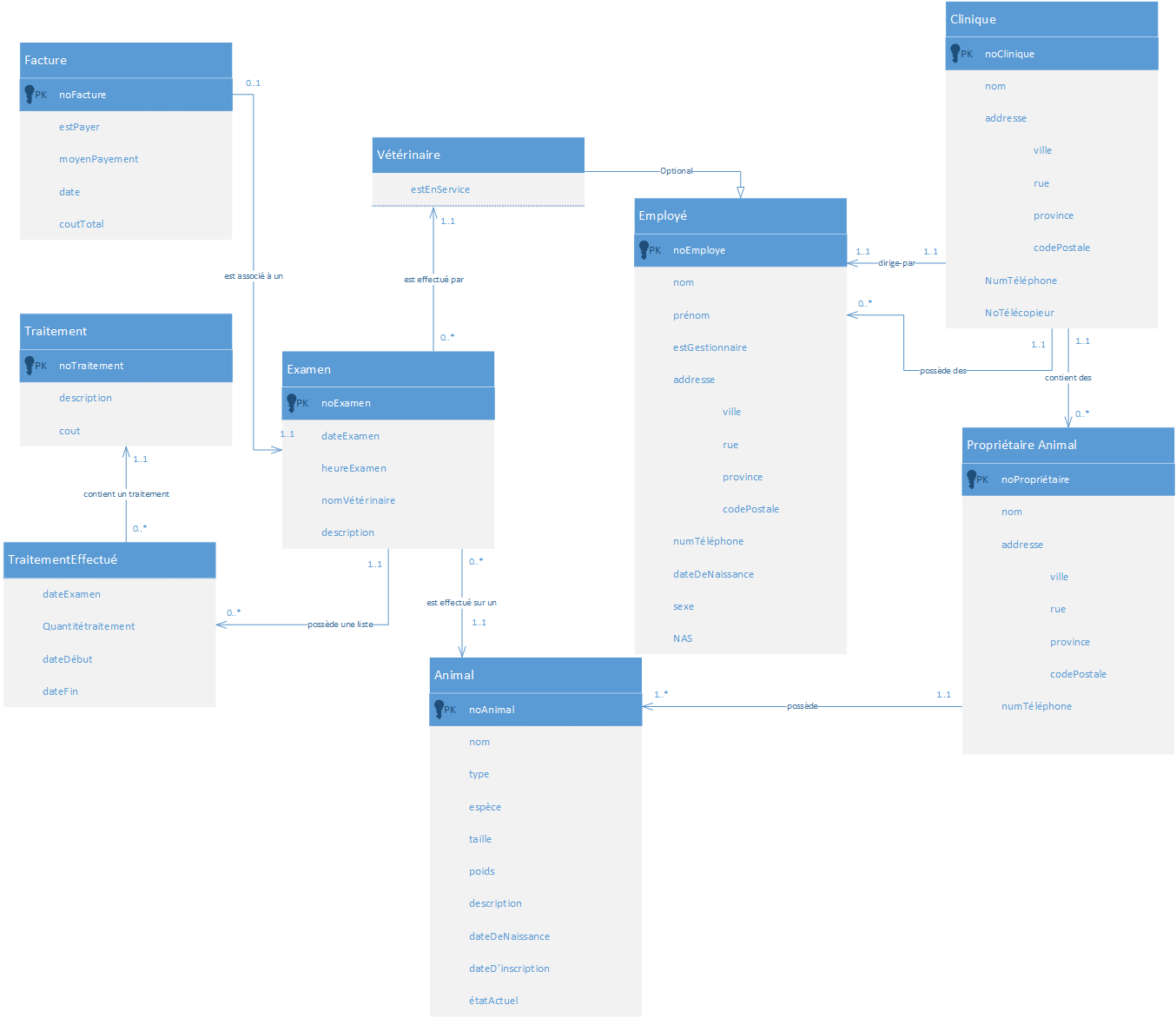
DF Facture noFacture -> estPaye, noExamen, noClinique, moyenPayement, date, coutTotal

DF noExamen,noClinique-> noFacture , estPaye, moyenPayement, date, coutTotal

* **TraitementEffectue**( noExamen, noClinique, noTraitement, quantiteTraitement, dateDebut, dateFin)

DF noExamen, noClinique -> noTraitement, quantiteTraitement, dateDebut, dateFin

Notre base de données est en 2NF. La première raison est qu’elle est considérée comme une base de données en 1NF car tous les attributs de toutes les tables sont atomique. C’est-à-dire qu’il y a aucun attribut qui est composé de plusieurs éléments. La deuxième raison explique pourquoi notre base de donné est une normale de deuxième forme et non de troisième forme est la présence des dépendances partielles dans la table clinique, propriétaire animal, animal et Employé. Les dépendances transitives sont la relation entre codePostal et le reste des informations de l’adresse. Le code postal permet de déduire le reste des informations par rapport à l’adresse même si le codePostale n’est pas la clé primaire de sa table.   
  
Une autre dépendance transitive qui empêche la base de données d’être une 3NF est la relation entre type et espèce dans la table Animal parce que pour chaque espèce il y a un type et l’espèce n’est pas la clé primaire de la table. La base de données est donc de deuxième forme normale.



**Modele Relationel**

* **Clinique**(noClinique, nom, rue, ville, province, codePostale, numTéléphone, numTélécopieur)
  + Primary Key : noClinique
* **ProprietaireAnimal** (noProprietaire, noClinique, nom, prenom, rue, ville, province, codePostale, numTéléphone)
  + Primary Key : (noProprietaire, noClinique)
  + Foreign Key : noClinique REFERENCES Clinique(noClinique)

Proprietaire Animal détient la clé principale de Clinique comme Foreign Key puisqu’il s’agit d’une relation de 0..\* par rapport à une 1..1.

* **Animal** (noAnimal, noClinique, noProprietaire, nom, type, espece, taille, poids, description, dateNaissance, dateInscription, etatActuel)
  + Primary Key : (noAnimal, noClinique)
  + Foreign Key : (noProprietaire, noClinique) REFERENCES ProprietaireAnimal (noProprietaire, noClinique)

Animal détient la clé principale de Propriétaire Animal comme Foreign Key  
puisqu’un propriétaire peut posséder plusieurs animaux. C’est donc une relation de 1..\* par rapport à 1..1.

* **Employe** (noEmploye, noClinique, nom, prenom, rue, ville, province, codePostale , numTelephone, dateNaissance, sexe, NAS, estGestionnaire)
  + Primary Key : noEmploye
  + Foreign Key : noClinique REFERENCES Clinique(noClinique)

Employe détient la clé primaire de Clinique comme foreign key puisqu’une clinique emploi plusieurs employés donc c’est une 1..\* et un 1..1.

* **Veterinaire** (noEmploye, estEnService)
  + Primary Key Primary Key : noEmploye
  + Foreign Key : noEmploye REFERENCES Employe (noEmploye)

Vétérinaire détient la clé primaire d’employé comme foreign key puisqu’un vétérinaire est un employé et il y a une relation d’héritage.

* **Examen** (noExamen, noClinique, noAnimal, dateExamen, heureExamen, noVeterinaire, description)
  + Primary Key : (noExamen, noClinique)
  + Foreign Key : noVeterinaire REFERENCES Veterinaire(noEmploye)
  + Foreign Key: noAnimal,noClinique REFERENCES Animal(noAnimal, noClinique)

Examen détient les clé primaires d’animal et vétérinaire  
puisque chaque examen est associé à un vétérinaire et un Animal.  
Ce qui cause une relation de 0..\* à 1..1 dans les deux cas.

* **Traitement (noTraitement, description, cout)**
  + Primary Key: noTraitement
* **Facture** (noFacture, estPaye, noExamen, noClinique, moyenPayement, date, coutTotal)
  + Primary Key : noFacture
  + Foreign Key: noExamen, noClinique REFERENCES Examen(noExamen, noClinique)

Facture détient la clé primaire d’Examen puisque chaque Facture est associé à un examen et il y a 0..1 facture pour 1..1 examen.

* **TraitementEffectue**( noExamen, noClinique, noTraitement, quantiteTraitement, dateDebut, dateFin)
* Primary Key : (noExamen, noTraitement)
* Foreign Key : noExamen REFERENCES Examen(noExamen)
* Foreign Key : noTraitement REFERENCES Traitement(noTraitement)  
    
  TraitementEffectue détient les clé primaire de noExamen car chaque traitement est  
  associé à un examen et il peut y avoir plusieurs traitements par examen.

La table traitementEffectue possède la clé primaire de la table traitement car chaque traitementEffectué est associé à un traitement.