NORMALISATION DE LA BDD

$ F N HD Q X HDJ U RDQS K D/SDCS L- R V p S K+LLQFHR/Q-J

PARTIE 1 : ANALYSE DES TABLES EXISTANTES

Classes classiques:

Propriétaire(#pseudo:string, photo: string, telephone: int, email: string, nom: string, prenom: string, age: int)

Dépendances fonctionnelles :
pseudo → photo
pseudo → telephone
pseudo → email
pseudo → nom
pseudo → prenom
pseudo → age

Cette table est en 1NF car il existe une clé, pseudo, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.
Cette table est en BCNF.

Locataire(#pseudo:string, photo: string, telephone: int, email: string, permis: string, validité: date, nom: string, prenom: string, age: int)
Dépendances fonctionnelles:
pseudo → photo
pseudo → telephone
pseudo → email
pseudo → permis
pseudo → validité
pseudo → nom
pseudo → prenom
pseudo → age

Cette table est en 1NF car il existe une clé, pseudo, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.

Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.
Cette table est en BCNF.

Entreprise(#id_entreprise:int, nom: string, adresse: string, ville: string, code_postal: int)

Dépendances fonctionnelles :
id_entreprise → nom
id_entreprise → adresse
id_etreprise → ville
id_entreprise → code_postal

Cette table est en 1NF car il existe une clé, id_entreprise, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.
Cette table est en BCNF.

Conducteur(#id_conducteur: int, #entreprise => Entreprise.id_entreprise, nom: string, prenom: string, age: int, photo: string, pseudo: string, telephone: int, email: string, permis : string, validité: date)

Dépendances fonctionnelles :
id_conducteur, entreprise → nom
id_conducteur, entreprise → prenom
id_conducteur, entreprise → age
id_conducteur, entreprise → photo
id_conducteur, entreprise → pseudo
id_conducteur, entreprise → telephone
id_conducteur, entreprise → email
id_conducteur, entreprise → permis
id_conducteur, entreprise → validité

Cette table est en 1NF car il existe une clé composée de id_conducteur et d' entreprise, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.

Cette table est en BCNF.

Contrat_location(#id_contrat:int, option_franchise: string, seuil_kilometrage: int, debut: date, fin: date, proprietaire => Propriétaire.pseudo, locataire => Locataire.pseudo, entreprise => Entreprise.id_entreprise)
Dépendances fonctionnelles :
id_contrat → option_franchise
id_contrat → seuil_kilometrage
id_contrat → debut
id_contrat → fin
id_contrat → proprietaire
id_contrat → locataire
id_contrat → entreprise

Cette table est en 1NF car il existe une clé, id_contrat, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.
Cette table est en BCNF.

Etat_des_lieux (#id_edl:int, #contrat=>Contrat_location.id_contrat, type(enum:{debut,fin}), photo: string, kilometrage: int, carburant: float, checklist: string, contrat_location => Contrat_location.id_contrat)

Dépendances fonctionnelles :
id_edl, contrat → type
id_edl, contrat → photo
id_edl, contrat → kilometrage
id_edl, contrat → carburant
id_edl, contrat → checklist
id_edl, contrat → contrat_location

Cette table est en 1NF car il existe une clé, composée de id_edl et contrat, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.
Cette table est en BCNF.

Facture (#id_facture:int, date: date, kilometrage: int, carburant: float, moyen_paiement: string, montant: float, contrat_location => Contrat_location.id_contrat)

Dépendances fonctionnelles :
id_facture → date
id_facture → kilometrage
id_facture → carburant
id_facture → moyen_paiement
id_facture → montant
id_facture → contrat_location

Cette table est en 1NF car il existe une clé, id_facture, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.
Cette table est en BCNF.

Commentaire (#id_commentaire:int, note: int, signaler: bool, description: string, contrat_location => Contrat_location.id_contrat)

Dépendances fonctionnelles :
id_commentaire → note
id_commentaire → signaler
id_commentaire → description
id_commentaire → contrat_location

Cette table est en 1NF car il existe une clé, id_commentaire, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.
Cette table est en BCNF.

Véhicule(#immatriculation: string, categorie: string, marque: string, modele: string, couleur: string, carburant: string, annee_mise_circulation : year, kilometrage: int, niveau_carburant: float, description: text, propriétaire => Propriétaire.pseudo)

Dépendances fonctionnelles :
immatriculation → categorie
immatriculation → marque
immatriculation → modele
immatriculation → couleur
immatriculation → carburant

immatriculation → annee_mise_circulation
immatriculation → kilometrage
immatriculation → niveau_carburant
immatriculation → description
immatriculation → proprietaire

Cette table est en 1NF car il existe une clé, immatriculation, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.
Cette table est en BCNF.

Pays(#nom: string)

Cette table est naturellement BCNF, car composée uniquement de sa clé.

Contrat_assurance (#id_assurance:int, nom_assurance: string, type: string, vehicule=> Véhicule.immatriculation)

Dépendances fonctionnelles :
id_assurance → nom_assurance
id_assurance → type
id_assurance → vehicule

Cette table est en 1NF car il existe une clé, id_assurance, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.
Cette table est en BCNF.

Annonce(#id_annonce: int, activite: bool, intitule: string, nombre_signalement: int, note: float, vehicule=> Véhicule.immatriculation)

Dépendances fonctionnelles :
id_annonce → activite
id_annonce → intitule
id_annonce → nombre_signalement
id_annonce → note
id_annonce → vehicule

Cette table est en 1NF car il existe une clé, id_annonce, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.
Cette table est en BCNF.

Option(#id_option:int, intitule: string)

Dépendances fonctionnelles :
id_option → intitule

Cette table est en 1NF car il existe une clé, id_option, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.
Cette table est en BCNF.

Periode(#id_periode: int, debut: date, fin: date)

Dépendances fonctionnelles :
id_periode → debut
id_periode → fin

Cette table est en 1NF car il existe une clé, id_periode, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.
Cette table est en 3NF car elle est en 2NF et de plus, aucun attribut ne dépend d'un autre attribut non clé.
Cette table est en BCNF.

Classes associations:

Est_disponible(#vehicule=>Vehicule.id_vehicule, #periode=> Periode.id_periode)

Peut_circuler(#vehicule=>Vehicule.id_vehicule, #pays=> Pays.nom)

Possede(#vehicule=>Vehicule.id_vehicule, #option => Option.id_option) avec Projection(Option,id_option)=Projection(Possède,option)

Les 3 classes associations sont naturellement en BCNF puisqu'elles ne contiennent que des attributs clés et aucunes ne dépendent des autres

PARTIE 2 : NORMALISATION DE 2 RELATIONS

Entreprise(#id_entreprise:int, nom: string, adresse: string, ville: string, code_postal: int)

On imagine les dépendances fonctionnelles suivantes :

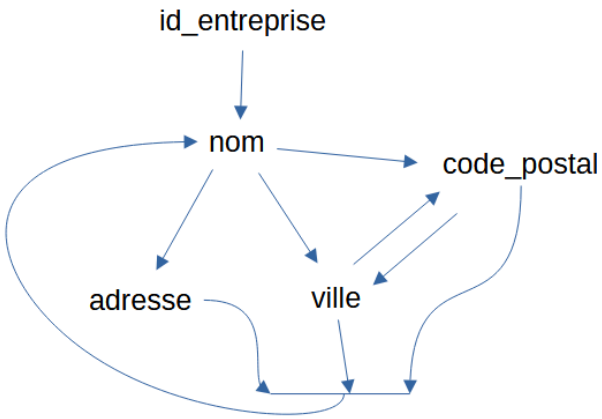
A partir d'un id d'entreprise, on peut déterminer le nom de l'entreprise
id_entreprise → nom

A partir du nom d'une entreprise, on peut déterminer son adresse, sa ville et son code postal
nom → adresse, ville, code_postal

Avec l'adresse d'une entreprise, on peut retrouver son nom :
adresse, ville, code_postal → nom

Avec le code_postal on peut trouver la ville et inversement :
code_postal → ville
ville → code_postal

Cette table est en 1NF car il existe une clé, pseudo, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.



Fermeture transitive :

Les dépendances fonctionnelles de base
id_entreprise → nom
nom → adresse
nom → ville
nom → code_postal
adresse, ville, code_postal → nom
code_postal → ville
ville → code_postal

Puis par transitivité on trouve :
id_entreprise → adresse (car id_entreprise → nom et nom → adresse)
id_entreprise → ville (car id_entreprise → nom et nom → ville)
id_entreprise → code_postal (car id_entreprise → nom et nom → code_postal)

Fermeture transitive:
{
id_entreprise → nom,
nom → adresse,
nom → ville,
nom → code_postal,
adresse, ville, code_postal → nom,
code_postal → ville,
ville → code_postal,
id_entreprise → adresse,

id_entreprise → ville,
id_entreprise → code_postal
}

Les trois dernières DFE sont trouvées par transitivité :
id_entreprise → adresse (car id_entreprise → nom et nom → adresse)
id_entreprise → ville (car id_entreprise → nom et nom → ville)
id_entreprise → code_postal (car id_entreprise → nom et nom → code_postal)

Couverture Minimale :
{
id_entreprise → nom
nom → adresse
nom → code_postal
adresse, ville, code_postal → nom
code_postal → ville
}

Cette couverture minimale permet de retrouver toutes les DF originelles.
Ici nous avons supprimé ville → code_postal car nous avons déjà code_postal → ville dans les DF, qui était bi-directionnelle donc nous nous sommes permis de supprimer une des deux DF.
Nous avons également supprimé nom → ville car par transitivité nous pouvons la retrouver avec nom → code_postal et code_postal → ville.

Clé candidate :

Ici la clé candidate est l'attribut id_entreprise qui permet de déduire l'ensemble des autres attributs.

Décomposition 2NF → 3NF:

Détail(#nom:string, adresse: string, ville: string, code_postal: int)

Entreprise(#id_entreprise:int, nom: string)

Ici, cette relation est bien en 3NF car elle est en 2NF et les attributs non clés sont déterminés à partir des attributs clés.

Locataire(#pseudo:string, photo: string, telephone: int, email: string, permis: string, validité: date, nom: string, prenom: string, age: int)

Dépendances fonctionnelles élémentaires :
pseudo → photo
pseudo → email
pseudo → permis
pseudo → telephone

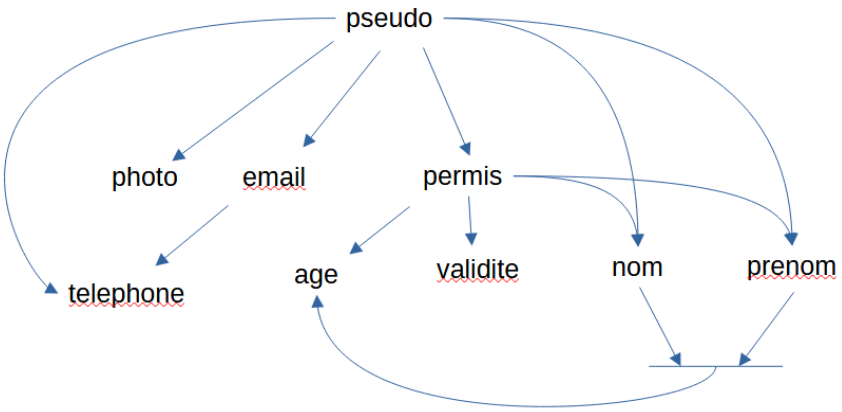
email → telephone
À partir d'une adresse mail, on peut retrouver le téléphone de la personne

telephone → email
À partir du numéro de la personne, on peut retrouver son email

nom, prenom → age
À partir du nom et du prénom de la personne, on peut retrouver son age

permis → validité, age, nom, prénom
À partir du permis du locataire, on peut retrouver la date de validité du permis, l'âge, le nom et le prénom du locataire

Cette table est en 1NF car il existe une clé, pseudo, qui détermine tous les attributs. De plus, tous les attributs sont atomiques.
Cette table est en 2NF car elle est en 1NF et de plus, aucun attribut ne dépend que d'une partie de la clé, car celle-ci n'est composée que d'un seul attribut.
Cette table n'est pas en 3NF car certains attributs dépendent d'un autre attribut non clé.



Normalisation:

Fermeture transitive
F+ = {
pseudo → photo,
pseudo → telephone,
telephone → email,

pseudo → email,
email → telephone,
pseudo → permis,
permis → validité, age, nom, prénom,
pseudo → validité,
pseudo → nom,
pseudo → prenom,
pseudo → age,
nom, prenom → age} = F

Couverture minimale
CM = {
pseudo → photo,
pseudo → telephone,
telephone → email,
pseudo → permis,
permis → validité, age, nom, prénom,
nom, prenom → age} = F

Décomposition 2NF → 3NF

Contact(#telephone:int, email:string)
Conducteur(#nom: string, #prenom: string, age: int)
Permis(#permis: string, validité: date, conducteur_nom=>Conducteur.nom, conducteur_prenom=>Conducteur.prenom)
Locataire(#pseudo:string, photo: string, telephone=>Contact, permis=>Permis)