INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

Base de données pour la gestion des lignes ferroviaires en Normandie

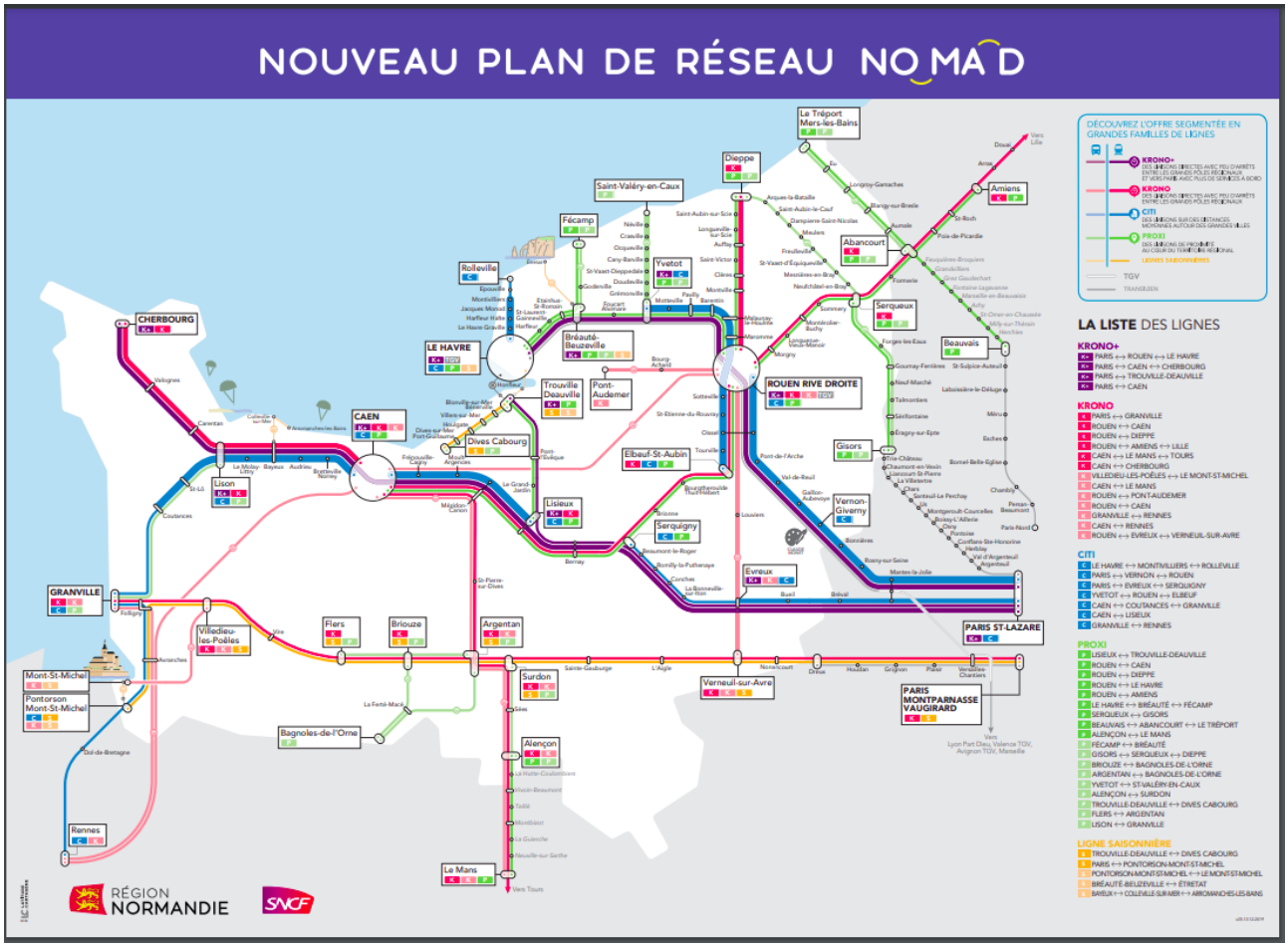


Figure 1: Plan issu du site de la SNCF

Réalisé par :
Ariane DEPONTHIEUX
Roxane LEDUC

Encadré par :
Mme. CHAIGNAUD

GM4 / Février 2023 - Mai 2023

Contents

1 Introduction

1.1 Contexte

Dans le cadre d'un cours sur les bases de données à l'INSA Rouen Normandie, nous avons travaillé à deux sur la conception et implantation d'une base de données relationnelle.

Nous avons choisis de travailler sur la gestion des lignes ferroviaires en Normandie.

1.2 But du projet

Le but de notre projet est d'élaborer une base de données de gestion de lignes ferroviaires régionales permettant de gérer les trajets de différents passagers. Cette base de données nous ait demandé par la SNCF Normandie.

Elle doit permettre aux agents ferroviaires d'avoir un accès simple des différents trajets, pour différents passagers. Un agent pourrait par exemple traquer un passager via cette base de données. Par ailleurs, elle doit aussi permettre de pouvoir répondre à une requête client qui souhaiterait notamment acheter un ticket.

Pour la partie requête client, cette base de données pourrait ainsi permettre une meilleure gestion de la page internet en charge de la gestion d'achat de ticket en Normandie.

1.3 Requêtes possibles

Voici une liste non exaustive des requêtes côté client et côté agent ferroviaire qui permettent d'illustrer :

Requêtes agents feroviaires

- Dans quel(s) train(s) va voyager le passager n°24 ?
- Quels sont les noms des personnes qui font le trajet Le Havre/Rouen Mardi 14/02/23 départ 8H03 ?
- Quel(s) sont les trains en circulation sur le Caen/Cherbourg le 08/03/23 ? (départ à Caen le 08/03, possibilité d'arriver le lendemain)

Requêtes clients

- Quels sont les trajets possibles Rouen/Le Havre avec un départ entre 8H et 12H le 18/03 ?
- A quelle heure arrive le train Cherbourg/Caen (du départ de 8H09 le 07/03/23) à l'arrêt Buyeux ?

2 Modèle Entité/Association

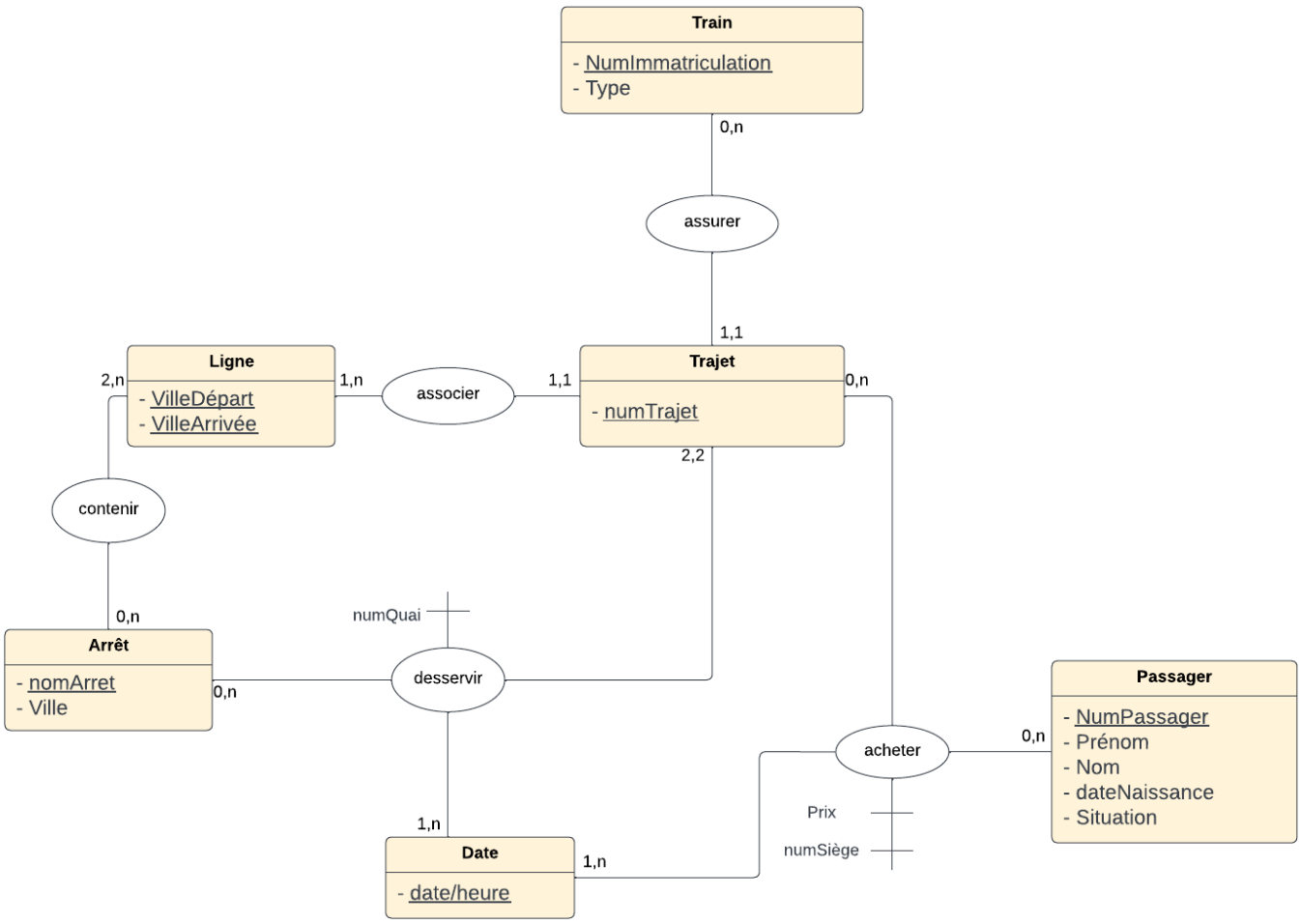


Figure 2: Modèle Entité/Association pour notre projet

Ce modèle s'articule principalement autour de l'entité `Trajet`. En effet, le but final de tout le travail réalisé pour le réseau normand est de réaliser des trajets pour les habitants. Le coeur de notre travail étant de permettre à n'importe qui de pouvoir se déplacer d'une gare A à une gare B le plus simplement possible. Il paraît ainsi assez naturel d'articuler notre modélisation autour de `Trajet`.

2.1 Entités

Trajet

Exemple de Trajet : 408

Un trajet correspond a plusieurs dates mais de façon répétitive est organisée. On parle par exemple du trajet Rouen/Breaute-Beauzeville du jeudi matin 8H07, chaque jeudi. Chaque `Trajet` s'identifie via un `numTrajet`. Il existe un nombre fini de trajets, et si des arrêts sont créés/supprimés, il y a la possibilité d'en ajouter/retirer dans la base de données. Nous avons pensé faire de `Trajet` une entité faible en associant `numTrajet` uniquement aux différentes

Ligne (autrement dit que chaque ligne a sa propre liste de trajet énuméré dans l'ordre), mais il semble ici nécessaire que les agents puissent avoir un accès facile à tous les trajets à l'aide d'uniquement un numéro.

De plus, chaque Trajet est assuré par une liste finie de Train, est associé à une unique Ligne, déssert exactement deux arrêts (arrêt de départ (intermédiaire) des passagers et arrêt d'arrivée (intermédiaire) des passagers) à des date/heure précises et à des numQuai donnés, et est acheté par des couples (Passager/Date) à un Prix et un numSiège attribué. Il est largement envisageable que l'entité Trajet ne soit associé à aucun couple (Passager/Date) car un Trajet est précisément un arrêt de départ (pas nécessairement l'arrêt de Départ de la ligne) et un arrêt d'arrivée (pas nécessairement le terminus).

Exemple : La ligne Caen/Cherbourg un Lundi midi : Le trajet Moult-Argences/Frénouville-Cagny (arrêts intermédiaires de la ligne) peut ne contenir aucun passager dans la base de données, bien que le train ne roule pas à vide.

Train

Exemple de Train : S.N.C.F 050.TQ.9; TGV

Chaque Train s'identifie via un NumImmatriculation qui lui ai propre auquel on précise le Type (TER, TGV, ...). Certains Train pouvants être hors service, il est possible qu'un Train n'assure aucun Trajet.

Ligne

Exemple de Ligne : Rouen/Le Havre

Chaque Ligne s'identifie via une VilleDépart et une VilleArrivée. Chaque Ligne est associé à au moins un Trajet (le trajet Arrêt de la ville de départ / Arrêt de la ville d'arrivée). Elle contient également plusieurs Arrêt, au minimum deux (arrêt de la ville de départ/ arrêt de la ville d'arrivée).

Arrêt

Exemple d'arrêt : Eu-la Mouillette; Eu

Chaque Arrêt s'identifie via un nomArret auquel on précise le nom de la Ville où se trouve cet arrêt. Le traffic ferroviaire normand restant relativement limité, on suppose ici qu'il n'existe pas deux arrêts de trains qui ait le même nom, afin que nomArret soit une clé primaire. Un Arret peut être associé à aucune Ligne, si celui-ci est en maintenance par exemple. Pour la même raison, il peut n'être associé à aucun couple (Date/Trajet).

Date

Exemple de Date : 08/03/23; 15H06

Chaque Date s'identifie via une date et une heure. L'heure étant celle où le train ferme ses portes pour quitter la gare pour tous les arrêts sauf le terminus; et l'heure d'ouverture de ses portes au terminus pour ce dernier. Chaque Date est associé à au moins un couple (Arret/Trajet). En effet, sinon celle-ci n'apparaîtrait pas dans la base de données. Pour la même raison, elle est associé à au moins un couple (Trajet/Passager) via la relation ternaire.

Passager

Exemple de Passager : 5066; Ariane; Deponthieux; 24/02/01; Etudiant

Chaque `Passager` s'identifie via un `numPassager`. Un passager décédé permet de libérer un `numPassager` pour un nouvel adhérent. Un `Passager` contient également `Prénom` et `Nom`, mais aussi `dateNaissance` et `Situation`, permettant ainsi le calcul du `Prix`.

2.2 Associations binaires

assurer

L'association `assurer` permet de relier les entités `Train` et `Trajet`. Ici, on considère que un trajet n'est assuré que par un et unique train pour simplifier notre modèle, bien que dans la vraie vie, le trajet du mercredi 10 Mai 2023 peut être assuré par un autre train que le même trajet mais au 17 Mai 2023. C'est un choix que nous faisons ici. Cela sous entendu une organisation des trains prédéfinies à l'avance est de manière périodique dans le temps. Cela n'empêche pas que l'on change le train référent d'un trajet mais implique une bonne initialisation du traffic par le réseau ferroviaire.

Par ailleurs, un train peut n'assurer aucun trajet si celui-ci est en cours de maintenance par exemple.

associer

Cette association permet de lier naturellement `Ligne` à tous ses `Trajet` associés. Pour chaque ligne, il faut la décomposer en ses différents arrêts et prendre toutes les combinaisons de 2 arrêts de cette ligne pour connaître le nombre de trajet de celle-ci. Ainsi, pour N arrêts (en comptant l'arrêt de départ et le terminus) d'une ligne, on compte $\binom{N}{2}$ trajets. On peut voir l'exemple dans la partie "Création de nos tables" qui permet d'avoir une idée du nombre de trajets associés à une ligne. A noter que la ligne "Le Havre - Paris" est différente de la ligne "Paris - Le Havre" selon notre définition.

contenir

Cette association lie de manière naturelle `Ligne` et `Arrêt`. Celle-ci permet à l'entité `Ligne` d'avoir l'information de ses arrêts de dessert. Une ligne contient au moins deux arrêts (arrêt de départ et terminus). Mais un arrêt peut n'appartenir à aucune ligne si celle-ci a par exemple été ajoutée à la base récemment car il est en train d'être construit et de s'inscrire dans la base de données mais n'est pas encore mis en service aux usagers et que aucun train ne s'y arrête pour le moment.

2.3 Association ternaire

acheter

Cette association permet d'exprimer quel passager a acheter quel trajet et à quelle date/heure, auquel on précise le prix et le numéro du siège qui dépend des trois entités.

Cette association est bien ternaire puisque toutes les branches sont de multiplicité `0,n` ou `1,n`.

2.4 Autre association

desservir

Cette association permet de distinguer un même trajet selon le jour et l'heure à laquelle il a lieu. En effet comme expliqué plus tôt, un trajet correspond par exemple à un "Yvetot-Bréauté-Beuzeville" des mercredis matin 10H, et `desservir` va permettre de préciser de quel mercredi il s'agit mais aussi à quel numéro de quai, car on permet ici une flexibilité de celui-ci en fonction du traffic actuel et des aléas du réseau.

Cette association n'est pas ternaire puisque l'une des branches est de multiplicité `2,2`.

On rappelle que un trajet se définit par un arrêt de départ et un arrêt d'arrivée, ceux notés sur le ticket du passager.

3 Modèle relationnel

3.1 Première étape

- Train(NumImmatriculation, Type)
- Trajet(numTrajet)
- Passager(NumPassager, Prénom, Nom, dateNaissance, Situation)
- Date(date/heure)
- Arrêt(nomArret, Ville)
- Ligne(VilleDépart, VilleArrivée)

3.2 Deuxième étape : clés étrangères

On ajoute des clés étrangères aux entités qui sont liées à d'autres entités par une multiplicité `0,1` ou `1,1`. Ici, seulement `Trajet` est concerné et on la modifie comme suit :

- Trajet(numTrajet, NumImmatriculation, VilleDépart, VilleArrivée)

3.3 Troisième étape

On créé des schémas de relations pour les relations binaires avec du `0,n`; `1,n` ou `2,n` d'un côté et de l'autre. Ici, `contenir` est concerné et on obtient :

- contenir(VilleDépart, VilleArrivée, nomArret)

Et pour simplifier la base de données, on introduit une clé `idContenir`. On obtient donc :

- contenir(idContenir, VilleDépart, VilleArrivée, nomArret)

Et pour

On créé des schémas de relations pour nos deux relations ternaires. Ici, `desservir` et `acheter` sont concernés et on obtient :

- desservir(numTrajet, date/heure, nomArret, numQuai)
- acheter(numTrajet, date/heure, NumPassager, Prix, numSiège)

De la même façon que pour la relation `contenir`, on introduit les clés `idDesservir` et `idAcheter` pour finalement avoir :

- desservir(idDesservir, numTrajet, date/heure, nomArret, numQuai)
- acheter(idAcheter, numTrajet, date/heure, NumPassager, Prix, numSiège)

3.4 Quatrième étape : sélection de clés

On décide de ne retirer aucune clé primaire car toutes seront utiles à nos requêtes.

On décide de modifier le nom de la clé primaire de `Train` : `NumImmatriculation` devient `idTrain`. En effet, cela ne change pas son utilisation mais le rend plus simple au niveau du code en `MariaDB`.

On ajoute également `idTrain` dans la base de données de `Train`, afin de permettre des jointures pour les requêtes.

3.5 Dénormalisation

On retire de la base de données la relation contenant uniquement des dates, qui est de toute évidence non pertinente. Le tableau des lignes n'apporte pas d'information utile non plus et ses clés primaires sont déjà dans `contenir`. On le supprime également.

Notre modèle devient finalement :

- Train(**idTrain**, Type)
- Trajet(**numTrajet**, idTrain, VilleDépart, VilleArrivée)
- Passager(**NumPassager**, Prénom, Nom, dateNaissance, Situation)
- Arrêt(**nomArret**, Ville)
- Ligne(**VilleDépart**, **VilleArrivée**)
- contenir(**idContenir**, VilleDépart, VilleArrivée, nomArret)
- desservir(**idDesservir**, numTrajet, date/heure, nomArret, numQuai)
- acheter(**idAcheter**, numTrajet, date/heure, NumPassager, Prix, numSiège)

4 Implémentation

L'implémentation de tables `SQL` est une étape clé dans la mise en place d'une base de données efficace. Dans ce contexte, `MariaDB` est un système de gestion de bases de données open source populaire qui offre des fonctionnalités avancées et une grande flexibilité. En créant nos tables sur `MariaDB`, nous pouvons stocker et organiser nos données de manière efficace, ce qui permet de faciliter la gestion et l'analyse de ces données.

4.1 Création de nos tables

Pour notre de base de données, on se focalise sur la ligne "Le Havre - Paris" pour tester nos requêtes. Cette ligne se constitue des arrêts suivants :

Figure 3: Ligne Le Havre - Paris

De la façon dont on a définit un trajet, on compte ici 4+3+2+1 = 10 trajets par trajet général (par exemple les passages de Le Havre - Paris des Mercredi matin). En supposant que tous les jours on ait 10 fois Le Havre - Paris assurés par la SNCF, et que cela se fait à heure fixe toutes les semaines, on comptera alors 10*7*10=700 trajets exisants dans la base de données pour recouvrir tous les cas pour cette ligne. La table desservir est donc la plus conséquente des tables et contient des informations essentielles à la gestion du traffic. Les informations se répètent mais c'est le meilleur moyen que nous avons trouvé pour faire apparaître une relation ternaire.

Pour tester nos requêtes, on entre par exemple les 10 trajets associées à la ligne "Le Havre - Paris" du Mercredi Matin. On entre alors quelques données à la main et on obtient les 8 tables suivantes grâce à `MariaDB` :

id_acheter	id_trajet	id_passager	date_heure	prix	num_siege
1	1	1	2023-04-12 23:00:00	5.5	92
2	1	2	2023-04-16 21:56:00	5.5	93
3	2	3	2023-04-16 21:56:00	7.5	120
4	5	4	2023-04-25 13:14:34	3.5	92

Figure 4: table `acheter`

nom_arret	ville
Breaute-Beuzeville	Breaute
Caen	Caen
Le Havre	Le Havre
Paris St-Lazare	Paris
Rouen Rive Droite	Rouen
Yvetot	Yvetot

Figure 5: table `Arret`

id_contenir	ville_depart	ville_arrivee	nom_arret
1	Le Havre	Paris	Paris St-Lazare
2	Le Havre	Paris	Rouen Rive Droite
3	Le Havre	Paris	Yvetot
4	Le Havre	Paris	Breaute-Beuzeville
5	Le Havre	Paris	Le Havre

Figure 6: table `contenir`

id_desservir	id_trajet	nom_arret	date_heure	num_quai
1	1	Le Havre	2023-04-26 13:17:00	4
2	1	Breaute-Beuzeville	2023-04-26 13:37:00	2
3	2	Le Havre	2023-04-26 13:17:00	4
4	2	Yvetot	2023-04-26 13:47:00	6
5	3	Le Havre	2023-04-26 13:17:00	4
6	3	Rouen Rive Droite	2023-04-26 14:04:00	2
7	4	Le Havre	2023-04-26 13:17:00	4
8	4	Paris St-Lazare	2023-04-26 15:19:00	18
9	5	Breaute-Beuzeville	2023-04-26 13:37:00	2
10	5	Yvetot	2023-04-26 13:47:00	6
11	6	Breaute-Beuzeville	2023-04-26 13:37:00	2
12	6	Rouen Rive Droite	2023-04-26 14:04:00	2
13	7	Breaute-Beuzeville	2023-04-26 13:37:00	2
14	7	Paris St-Lazare	2023-04-26 15:19:00	18
15	8	Yvetot	2023-04-26 13:47:00	6
16	8	Rouen Rive Droite	2023-04-26 14:04:00	2
17	9	Yvetot	2023-04-26 13:47:00	6
18	9	Paris St-Lazare	2023-04-26 15:19:00	18
19	10	Rouen Rive Droite	2023-04-26 14:04:00	2
20	10	Paris St-Lazare	2023-04-26 15:19:00	18

Figure 7: table `desservir`

ville_depart	ville_arrivee
Caen	Paris
Le Havre	Paris

Figure 8: table `Ligne`

id_passager	nom	prenom	date_naissance	situation
16	Leduc	Roxane	2001-07-11	Etudiant
17	Deponthieux	Ariane	2001-02-24	Etudiant
18	Chaignaud	Nathalie	1980-03-30	none
19	Costaud	Monsieur	1992-04-01	Militaire
20	Andre	Anouk	2001-09-29	Etudiant

Figure 9: table `Passager`

id_train	type
1	TER
2	TGV
3	TGV
4	TER

Figure 10: table `Train`

id_train	id_trajet	ville_depart	ville_arrivee
2	1	Le Havre	Paris
2	2	Le Havre	Paris
2	3	Le Havre	Paris
2	4	Le Havre	Paris
2	5	Le Havre	Paris
2	6	Le Havre	Paris
2	7	Le Havre	Paris
2	8	Le Havre	Paris
2	9	Le Havre	Paris
2	10	Le Havre	Paris
3	11	Caen	Paris

Figure 11: table `Trajet`

4.2 Création de "Views"

Une vue (ou vue virtuelle) est une requête SQL qui est enregistrée sous forme d'objet dans la base de données. Elle permet de créer une représentation virtuelle des données stockées dans une ou plusieurs tables de la base de données, en permettant aux utilisateurs d'accéder à ces données de manière simplifiée et structurée.

Afin de comprendre comment les manipuler sous SQL, nous avons décidé d'en créer une, permettant d'afficher les informations du(des) ticket(s) du passager n°1 (cf. Requête n°1).

SQL :

```
CREATE VIEW dans_quel_train_voyage_X AS
    SELECT DISTINCT A.prix, D.*, A.num_siege
    FROM acheter as A, desservir as D
    WHERE (A.id_passager = 1) AND (A.id_trajet = D.id_trajet);
```

4.3 12 requêtes

Requête n°1 : Afficher les informations du(des) ticket(s) du passager n°1.

SQL :

```
SELECT DISTINCT A.prix, D.*, A.num_siege
FROM acheter as A, desservir as D
WHERE (A.id_passager = 1)
    AND (A.id_trajet = D.id_trajet);
```

Algèbre relationnelle :

$$\Pi_{prix,\ ...,\ num\_siege}[\sigma_{id\_passager=1}(acheter) \infty_{id\_trajet} desservir]$$

prix	id_desservir	id_trajet	nom_arret	date_heure	num_quai	num_siege
5.5	1	1	Le Havre	2023-04-26 13:17:00	4	92
5.5	2	1	Breaute-Beuzeville	2023-04-26 13:37:00	2	92

Figure 12: résultat de la requête 1

Requête n°2 : Quels sont les trains en circulation le 26/04/2023 ?

SQL :

```
SELECT DISTINCT t.id_train
FROM Train t, desservir d, Trajet tr
WHERE d.date_heure LIKE '2023-04-26%'
    AND t.id_train=tr.id_train
    AND d.id_trajet=tr.id_trajet;
```

Algèbre relationnelle :

$$\Pi_{id\_train}[(\sigma_{date\_heure='2023-04-26\%'}(desservir) \infty_{id\_trajet} Trajet) \infty_{id\_train} Train]$$

id_train
2

Figure 13: résultat de la requête 2

Requête n°3 : Nombre de trajets réservés par client du réseau normandie, pour les clients ayant réservés plus de 2 tickets enregistrés dans la base des données (avec leur nom et prénom)

Pour la requête n°3, on introduit l'opérateur ${}_a\gamma_F()$ avec a étant l'attribut choisi pour regrouper entre eux les éléments ayant cet attribut d'identique et F une fonction de type $MIN, MAX, AVG, COUNT, ...$ que l'on applique à ses éléments regroupés et on ajoute cette information à la table.

SQL :

```
SELECT DISTINCT prenom, nom, COUNT(a.id_trajet) AS 'Nombre de trajets
reserves'
FROM Passager p, acheter a
WHERE a.id_passager=p.id_passager
GROUP BY a.id_passager
HAVING COUNT(a.id_trajet)>=2;
```

Algèbre relationnelle :

$\Pi_{prenom,nom,COUNT(*)}[\sigma_{COUNT(*)>1}(_{id\_passager}\gamma_{COUNT(*)}(acheter \infty_{id\_passager} Passager))]$

```
Empty set (0,001 sec)
```

Figure 14: résultat de la requête 3

Requête n°4 : Liste des passagers ayant au moins un trajet pour départ de 'Yvetot' ou d'arrivée 'Yvetot' au mois d'Avril 2023.

SQL :

```
SELECT DISTINCT prenom, nom
FROM Passager p
WHERE EXISTS (SELECT *
              FROM desservir d, acheter a
              WHERE d.nom_arret='Yvetot'
                  AND (d.date_heure LIKE '%04-26%')
                  AND d.id_trajet=a.id_trajet
                  AND a.id_passager=p.id_passager);
```

Algèbre relationnelle :

$\Pi_{prenom,\ nom}[(\sigma_{nom\_arret='Yvetot' \wedge date\_heure='\%04-26\%'}(desservir) \infty_{id\_trajet} acheter) \infty_{id\_passager} Passager]$

prenom	nom
Nathalie	Chaignaud
Monsieur	Costaud

Figure 15: résultat de la requête 4

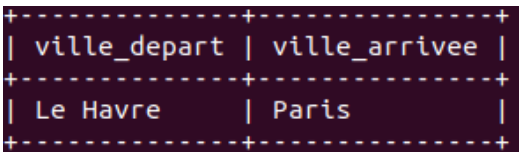
Requête n°5 : Quelles sont les lignes qui désservent l'arrêt 'Breaute-Beuzeville' OU 'Yvetot' ?

SQL :

```
SELECT DISTINCT ville_depart, ville_arrivee
FROM desservir d, Trajet t
WHERE d.id_trajet=t.id_trajet
    AND d.nom_arret IN ('Breaute-Beuzeville', 'Yvetot');
```

Algèbre relationnelle :

$\Pi_{ville\_depart,\ ville\_arrivee}[(\sigma_{nom\_arret='Breaute-Beuzeville'}(desservir) \infty_{id\_trajet} \cup \sigma_{nom\_arret='Breaute-Beuzeville'}(desservir)) \infty_{id\_trajet} Trajet]$



ville_depart	ville_arrivee
Le Havre	Paris

Figure 16: résultat de la requête 5

Requête n°6 : "A quelle heure et quel quai arrive le train à Yvetot (celui qui part du Havre et qui est censé arrivé vers 13H45 le 26/04/2023) ?", se demande un proche de Nathalie Chaignaud venu pour la récupérer. Cette information doit être affichée en gare sur un panneau d'affichage – Faire afficher toutes les informations utiles pour les panneaux d'affichages des trains qui arrivent entre 13H et 14H le 26/04/23 à Yvetot.

SQL :

```
SELECT DISTINCT ville_depart AS 'En provenance de',
                TIME(d.date_heure) AS 'arrivee prevue a',
                traj.id_train, type, num_quai
FROM desservir d, Train t, Trajet traj
WHERE traj.id_trajet=d.id_trajet
    AND t.id_train=traj.id_train
    AND d.nom_arret='Yvetot'
    AND d.date_heure BETWEEN TIMESTAMP('2023-04-26 13:00:00') AND
                             TIMESTAMP('2023-04-26 14:00:00');
```

Algèbre relationnelle :

$\Pi_{ville\_depart,...,num\_quai}[(\sigma_{nom\_arret='Yvetot',\ '2023-04-26\ 13:00:00'\leq date\_heure\leq'2023-04-26\ 14:00:00'}(dessservir)\infty_{id\_trajet}Trajet)\infty_{id\_train}Train]$

En provenance de	arrivée prévue à	id_train	type	num_quai
Le Havre	13:47:00	2	TGV	6

Figure 17: résultat de la requête 6

Requête n°7 : Liste des trains prévus pour parcourir au moins un trajet de chaque ligne du réseau Normandie sur la base des trajets enregistrés (la division). Le type du train est précisé pour les statistiques.

SQL :

On sélectionne tous les trains qui ne font pas partis de la liste des trains (t2) pour lesquels il existe une ligne pour laquelle il n'existe pas de trajet qui est assuré par ce train (t2).

Pour tester cette requête, on ajoute un trajet sur la ligne Caen-Paris parcouru par le train n°2, afin de vérifier notre résultat.

```
INSERT INTO Trajet(id_train, ville_depart, ville_arrivee) VALUES
                                                    (2,'Caen','Paris');
SELECT DISTINCT id_train, type
FROM Train
```

```
WHERE id_train NOT IN (
    SELECT DISTINCT id_train
    FROM Train
    WHERE EXISTS (
            SELECT ville_depart, ville_arrivee
            FROM Ligne l
            WHERE NOT EXISTS (
                    SELECT id_trajet
                    FROM Trajet t1
                    WHERE t1.ville_depart = l.ville_depart
                            AND t1.ville_arrivee = l.ville_arrivee
                            AND Train.id_train = t1.id_train
                    )
    )
);
```

Algèbre relationnelle :

$$\Pi_{id\_train}(Train) - \Pi_{id\_train}(\sigma_{ville\_depart,\ ville\_arrivee}(Ligne) \infty \Pi_{id\_train}(Trajet) (\sigma_{ville\_depart=l.ville\_depart \wedge ville\_arrivee=l.ville\_arrivee \wedge Train.id\_train=t1.id\_train}(Ligne \infty Trajet))))$$

id_train	type
2	TGV

Figure 18: résultat de la requête 7

Requête n°8 : Liste des passagers assis à la place 92 dans le train n°2 le 26 avril 2023? (ce siège est endommagé, a remarqué un personnel de nettoyage des trains en fin de journée)

SQL :

```
SELECT DISTINCT prenom, nom
FROM Passager p, acheter a, Trajet t
WHERE a.num_siege=92 AND t.id_train='2' AND t.id_trajet=a.id_trajet
AND p.id_passager=a.id_passager
AND EXISTS (SELECT *
            FROM desservir d
            WHERE d.date_heure LIKE '2023-04-26%'
                AND d.id_trajet=t.id_trajet);
```

Une autre façon de le coder :

```
SELECT DISTINCT prenom, nom
FROM Passager p, acheter a, Trajet t, desservir d
WHERE a.num_siege=92
    AND t.id_train='2'
    AND t.id_trajet=a.id_trajet
    AND p.id_passager=a.id_passager
```

```
    AND d.date_heure LIKE '2023-04-26%'
    AND d.id_trajet=t.id_trajet;
```

Algèbre relationnelle :

$\Pi_{prenom,\ nom}[((\sigma_{num\_siege=92}(acheter) \infty_{id\_trajet} \sigma_{id\_train='2'}(Trajet)) \infty_{id\_passager} Passager) \infty_{id\_trajet} \sigma_{date\_heure='2023-04-26\%'}(desservir)]$

prenom	nom
Roxane	Leduc
Monsieur	Costaud

Figure 19: résultat de la requête 8

Requête n°9 : Nombre de passagers par situation permettant une réduction (nombre de Etudiant, nombre de Militaire, nombre de Senior).

SQL :

```
SELECT situation, COUNT(situation) AS 'Nombre de personnes'
FROM Passager
WHERE situation!='none'
GROUP BY(situation);
```

Algèbre relationnelle :

$\Pi_{situation,\ COUNT(*)}[_{situation}\gamma_{COUNT(*)}(\sigma_{situation\neq'none'}(Passager))]$

situation	Nombre de personnes
Etudiant	3
Militaire	1

Figure 20: résultat de la requête 9

Requête n°10 : Quels sont les trajets possibles Le Havre/Rouen avec un départ entre 10H et 14H le 26/04/2023 ? (demande d'un client pour un achat de ticket sur le site web)

SQL :

```
SELECT d_depart.id_trajet, d_depart.date_heure, d_arrivee.date_heure
FROM desservir d_depart, desservir d_arrivee
WHERE d_depart.nom_arret='Le Havre'
    AND d_depart.date_heure BETWEEN TIMESTAMP('2023-04-26 10:00:00')
AND TIMESTAMP('2023-04-26 14:00:00')
    AND d_arrivee.nom_arret='Rouen Rive Droite'
    AND d_arrivee.id_trajet=d_depart.id_trajet
    AND d_depart.date_heure<d_arrivee.date_heure;
```

Algèbre relationnelle :

$\Pi_{id\_trajet,\ date\_heure}[\sigma_{nom\_arret='Le\ Havre','2023-04-26\ 10:00:00 \leq date\_heure \leq '2023-04-26\ 14:00:00'}(desservir\ d\_depart) \infty_{d\_depart.id\_trajet=d\_arrivee.id\_trajet, d\_depart.date\_heure<d\_arrivee.date\_heure} \sigma_{nom\_arret='RouenRiveDroite'}(desservir\ d\_arrivee)]$

id_trajet	date_heure	date_heure
3	2023-04-26 13:17:00	2023-04-26 14:04:00

Figure 21: résultat de la requête 10

Requête n°11 : Quel est le prix moyen du trajet Le Havre/ Breaute-Beuzeville n°1 payé par les passagers ?

SQL :

```
SELECT DISTINCT AVG(prix) AS 'Moyenne du prix paye par les passagers
                                pour le trajet numero 1 (en euros)'
FROM acheter
WHERE id_trajet=1;
```

Algèbre relationnelle :

$\gamma_{AVG(prix)}(\sigma_{id\_trajet=1}(acheter))$

Moyenne du prix payé par les passagers pour le trajet n°1 (€)
5.5

Figure 22: résultat de la requête 11

Requête n°12 : Tous les arrêts desservis par la ligne Le Havre/ Paris départ Le Havre le 04/26/2023 à 13H17 (avec les horaires de chaque arrêt)

SQL :

```
SELECT DISTINCT d_arret.nom_arret, d_arret.date_heure
FROM desservir d_depart, desservir d_arret
WHERE d_depart.id_trajet=d_arret.id_trajet
    AND d_depart.nom_arret='Le Havre'
    AND d_depart.date_heure LIKE '2023-04-26 13:17:00';
```

Algèbre relationnelle :

$\Pi_{nom\_arret,date\_heure}[(\sigma_{nom\_arret='Le\ Havre',\ date\_heure='2023-04-26\ 13:17:00'}(desservir\ d\_depart)) \infty_{d\_depart.id\_trajet=d\_arret.id\_trajet}(desservir\ d\_arret)]$

nom_arret	date_heure
Le Havre	2023-04-26 13:17:00
Breaute-Beuzeville	2023-04-26 13:37:00
Yvetot	2023-04-26 13:47:00
Rouen Rive Droite	2023-04-26 14:04:00
Paris St-Lazare	2023-04-26 15:19:00

Figure 23: résultat de la requête 12

5 Conclusion

Notre projet consistait à développer une base de données de gestion de lignes ferroviaires régionales pour répondre aux besoins de la SNCF Normandie. Cette base de données permet aux agents ferroviaires d'accéder facilement aux informations sur les différents trajets et aux passagers. Les agents peuvent ainsi suivre un passager spécifique en utilisant cette base de données. La base de données permet notamment de répondre aux requêtes des clients, notamment pour l'achat de tickets. Grâce à cette base de données, la gestion de la page internet chargée de la vente de tickets en Normandie peut être améliorée.

Ce projet nous a apporté une expérience précieuse en tant que futurs ingénieurs, nous permettant d'acquérir une compréhension approfondie des principes de conception et de gestion d'une base de données. Nous avons pu mettre en pratique nos connaissances en SQL et en particulier dans le langage spécifique à MariaDB.

En utilisant MariaDB, nous avons bénéficié d'un système de gestion de base de données robuste et performant. MariaDB est un choix judicieux pour ce projet en raison de sa compatibilité avec le langage SQL standard, sa grande stabilité et sa facilité d'utilisation.

En résumé, ce projet nous a permis de développer nos compétences en conception et gestion de bases de données, tout en mettant en évidence l'intérêt et les avantages de l'utilisation de MariaDB dans un contexte professionnel.

A noter que certaines tables et certains champs n'ont pas étaient utiles pour nos requêtes, en particulier les tables `Arret` et `contenir`. Celles-ci auraient pu tout de même servir si on voulait par exemple préciser la ville d'un arrêt qui n'a pas un nom explicite. Ici, on s'est concentré sur un ligne assez connue de la Normandie, les noms des arrêts étaient donc identiques ou quasiment identiques à la ville où ils se trouvent.

6 Bibliographie

- *Carte du réseau - Site de la SNCF*
  https://www.ter.sncf.com/normandie/se-deplacer/carte-du-reseau

- *Requêtes agrégats - COUNT*
  https://ressources.unisciel.fr/sillages/informatique/bdd/ch04_sql/co/14_agregats.html