NF18-P23

Projet NF18 – Gestion des trains
Rapport final

EL ATIA Nadia - HU Xuanyao - MIGUEL Laura - ROBEAU Elisa

Sommaire

I – Contexte du projet

Dans le cadre de l'UV NF18, nous avons créé une base de données, par groupe de quatre étudiants, afin de mettre en pratique notre apprentissage des différentes notions vues en cours.

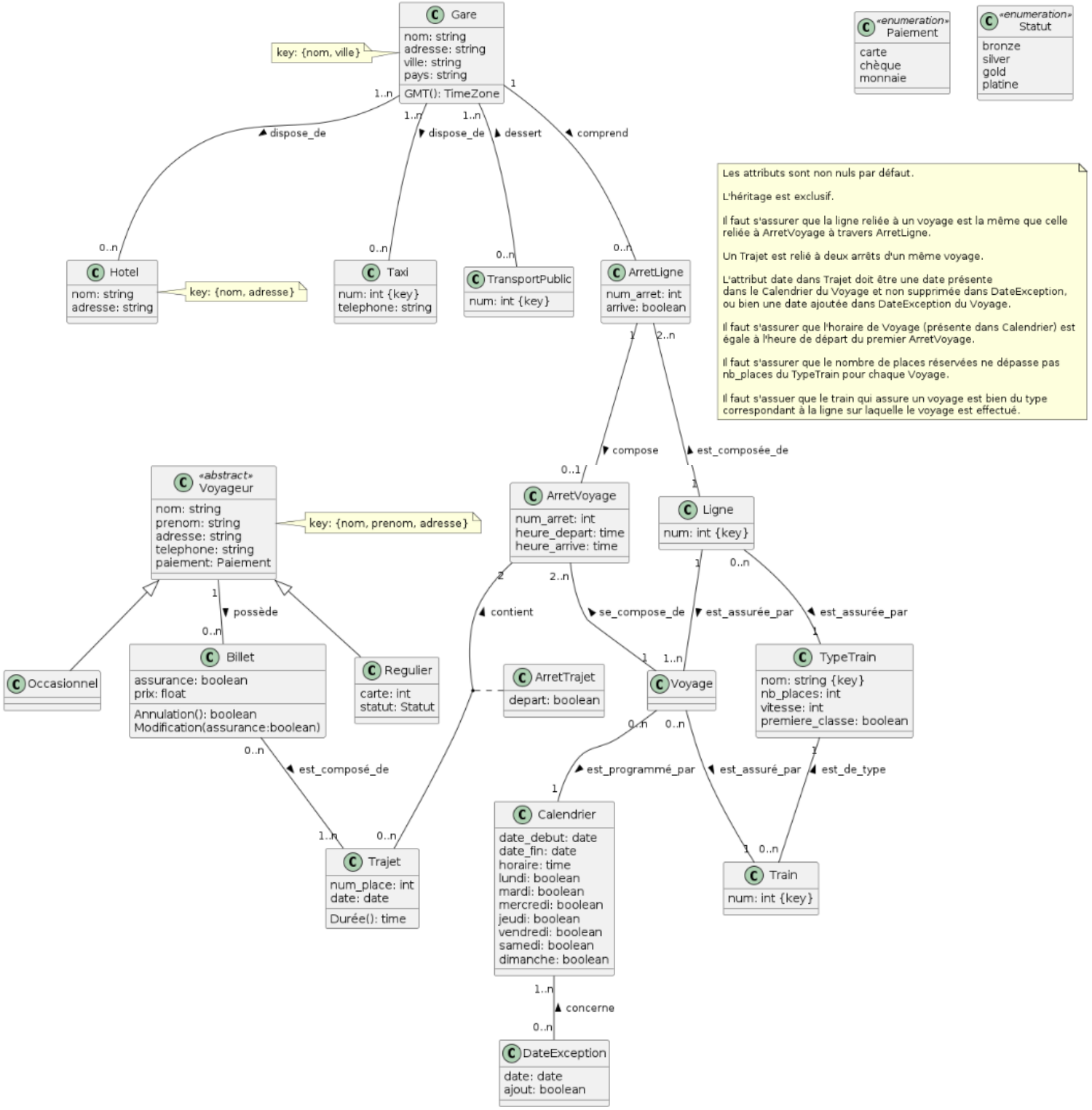
Notre projet est sur la gestion des trains. Nous devions implémenter un système qui permettait :

- De gérer les gares, les trains et les lignes ;
- Du côté des clients, de rechercher des trajets, consulter les horaires des trains et acheter les billets ;
- D'obtenir des statistiques sur la société.

Ce rapport reprend les différentes étapes de la création de cette base de données où, dans un premier temps, nous avons réalisé une base de données relationnelle, puis, dans un second temps, nous l'avons modifiée en base de données non relationnelle.

II – Base de données relationnelle

1 - Modélisation avec UML



Nous avons tout d'abord réalisé le MCD ci-dessus pour répondre aux attentes de la société de chemin de fer. La société de chemin de fer a demandé des fonctionnalités concernant deux types d'utilisateurs : les employés d'une part, et les clients de la société de l'autre.

Du côté des employés, la société souhaitait gérer (ajouter, supprimer, modifier) les gares, les lignes, les trains (et types de trains), les voyages et leurs calendriers. Nous avons fait le choix d'ajouter des structures intermédiaires telles que les classes « ArretLigne » et « ArretVoyage », afin de pouvoir modéliser au mieux le système (notamment respecter les contraintes de la société comme « avoir accès à la gare de départ d'une ligne », ou encore « un voyage ne s'arrête pas forcément dans toutes les gares de la ligne mais tous ses arrêts appartiennent à la même ligne »).

Du côté des clients, nous avons créé la classe « Voyageur », classe mère de deux classes filles : les voyageurs occasionnels et les voyageurs réguliers (pour lesquels on souhaite enregistrer notamment une carte de voyageurs et un statut).

Ce sont les classes « Billet » et « Trajet » qui permettent de faire communiquer les parties employés et clients, étant donné que ce sont deux tables auxquelles à la fois les employés et les clients doivent avoir accès (les employés pour obtenir des statistiques sur la société comme les bénéfices, et les clients pour pouvoir utiliser les services de la société).

Enfin, la société de chemin de fer souhaite donner à ses clients des informations sur les hôtels, transports publics et taxis à proximité des gares : nous avons donc ajouté 3 classes pour les gérer.

Une des difficultés que nous avons rencontrées était sur la différence entre une ligne, un voyage et un trajet. Nous avons fini par trancher qu'une ligne correspondait aux arrêts physiques possibles pour un train (donc les gares par lesquelles le chemin de fer passe). Mais un train n'est pas forcé de s'arrêter à tous les arrêts de la ligne : un voyage est ainsi constitué des arrêts auxquels le train s'arrête réellement. Enfin, un trajet est un voyage du point de vue d'un voyageur : un voyageur peut monter au milieu du voyage d'un train, et sortir du train avant le terminus.

Une autre difficulté a été la modélisation des calendriers hebdomadaires associés à des voyages, et en particulier des dates où le calendrier change exceptionnellement : nous avons réglé le problème en choisissant de créer une classe dédiée au calendrier hebdomadaire (la classe « Calendrier ») avec un attribut par jour de la semaine pour enregistrer sur quels jours le calendrier est valable, et une seconde table (« DateException ») qui concerne directement la classe « Calendrier » pour indiquer les changements de calendrier.

Enfin, de nombreuses contraintes n'ont pas pu être directement exprimées dans les classes et associations : par exemple, le fait que la date d'un trajet doit être présente dans le calendrier du voyage associé et qu'il faut s'assurer que ce n'est pas une date où le voyage est exceptionnellement annulé.

En même temps que le MCD, nous avons rédigé une note de clarification, expliquant en détail l'UML (s'y référer pour plus de détails concernant l'UML).

2 - Passage du MCD au MLD

Nous avons ensuite traduit notre MCD en MLD afin d'implémenter nos tables en SQL par la suite.

Sur cette étape, nous avons décidé que la différenciation entre voyageurs occasionnels et réguliers se ferait grâce à un attribut « occasionnel » au sein de la table « Voyageur » (choix d'un héritage par classe mère, car la classe mère possède une association avec la table « Billet », et que les classes filles n'en possèdent pas). Cela a entraîné des contraintes à vérifier sur les attributs comme la présence d'un numéro de carte et d'un statut dans le cas où l'attribut « occasionnel » est à False.

Pour respecter les contraintes comme « une gare peut être proche de plusieurs hôtels et un hôtel peut être proche de plusieurs gares » (association *-*), nous avons dû créer des tables telles que « DisposeHôtel », reliant donc gares et hôtels (idem pour les transports publics, les taxis, et cas similaires pour les tables « Billets » et « Trajet », et « Calendrier » et « DateException »).

Il a également fallu ajouter des identifiants uniques pour les voyages, billets, trajets et calendriers.

La principale difficulté de cette étape a été de bien exprimer toutes les contraintes de cardinalité comme « un hôtel est proche d'au moins une gare » (traduit par des projections dans le MLD). Nous avons aussi pu exprimer des contraintes venant de l'UML qui, jusqu'à lors, n'avait pas pu être respectées, comme s'assurer que la ligne reliée à un voyage est la même que celle reliée à « ArretVoyage » à travers « ArretLigne ».

3 - Implémentation de la base en SQL

Nous avons ensuite implémenté notre base de données avec SQL et l'avons remplie. Lors de cette étape, nous nous sommes aussi occupées des vues pour préparer l'application Python (notamment les vues qui nous permettront de nous assurer que les contraintes de projection seront respectées, et d'autres pour faciliter l'accès à des informations). Voici quelques captures d'écran des tables et vues (toutes les tables et vues ne sont pas présentées).

-- Gare

nom	ville	adresse	pays
Gare ferroviaire	Compiegne	rue Ferdinand Bacs	France
Gare du Nord	Paris	18 rue de Dunkerque	France
Gare de Lyon	Paris	place Louis-Armand	France
Gare Montparnasse	Paris	17 boulevard de Vaugirard	France
Gare ferroviaire	Creil	rue Despinas	France
Gare ferroviaire	Pont-Sainte-Maxence	rue de la Paix	France
Gare ferroviaire	Amiens	47 place Alphonse Fiquet,	France
Gare Bruxelles-Midi	Bruxelles	47B avenue Fonsny	Belgique

-- ArretLigne

num_arret	ligne	arrive	nom_gare	ville_gare
1	27	f	Gare du Nord	Paris
2	27	f	Gare ferroviaire	Creil
3	27	f	Gare ferroviaire	Pont-Sainte-Maxence
4	27	f	Gare ferroviaire	Compiegne
5	27	t	Gare ferroviaire	Amiens
1	156	f	Gare du Nord	Paris
2	156	t	Gare Bruxelles-Midi	Bruxelles

-- Calendrier (hebdomadaire des voyages)

id_calendrier	date_debut	date_fin	horaire	lundi	mardi	mercredi	jeudi	vendredi	samedi	dimanche
1	2020-01-01	2022-12-31	12:00:00	t	t	t	t	t	t	t
2	1999-11-24	2016-02-14	18:00:00	t	t	t	t	t	t	f
3	2017-05-01	2025-09-30	14:00:00	f	t	t	t	t	f	f

-- Trajet

id_trajet	num_place	date_
1	23	2021-03-27
2	57	2020-07-04
3	567	2020-07-04

-- Voyageur

nom	prenom	adresse	telephone	paiement	carte	statut	occasionnel
Beauchamp	Elisabeth	45 rue de la République	0765438729	carte			t
Smith	Henry	288 avanue du Général de Gaulle	0614263798	monnaie	563	bronze	f

-- Billet

id_billet	assurance	prix	voyageur_nom	voyageur_prenom	voyageur_adresse
1	f	34.7	Smith	Henry	288 avanue du Général de Gaulle
2	t	9.45	Smith	Henry	288 avanue du Général de Gaulle

-- DisposeHotel (lien entre Hotel et Gare)

nom_gare	ville_gare	nom_hotel	adresse_hotel
Gare ferroviaire	Compiegne	B&B	10 avenue Marcellin Berthelot
Gare du Nord	Paris	Marriott	70 avenue des Champs-Elysees
Gare de Lyon	Paris	Marriott	70 avenue des Champs-Elysees
Gare Montparnasse	Paris	Marriott	70 avenue des Champs-Elysees
Gare du Nord	Paris	Ritz	15 place Vendôme
Gare de Lyon	Paris	Ritz	15 place Vendôme
Gare Montparnasse	Paris	Ritz	15 place Vendôme

-- v_checkdate (vue utile pour vérifier que la date d'un trajet est présente dans le calendrier et que ce n'est pas une date avec suppression exceptionnelle, ou bien que la date n'est pas dans le calendrier mais que c'est un ajout exceptionnel)

id_trajet	trajet_date	id_calendrier	date_debut	date_fin	lundi	mardi	mercredi	jeudi	vendredi	samedi	dimanche	date_exception	ajout
1	2021-03-27	1	2020-01-01	2022-12-31	t	t	t	t	t	t	t		
1	2021-03-27	1	2020-01-01	2022-12-31	t	t	t	t	t	t	t		
2	2020-07-04	1	2020-01-01	2022-12-31	t	t	t	t	t	t	t		
2	2020-07-04	1	2020-01-01	2022-12-31	t	t	t	t	t	t	t		
3	2020-07-04	2	1999-11-24	2016-02-14	t	t	t	t	t	t	f		
3	2020-07-04	2	1999-11-24	2016-02-14	t	t	t	t	t	t	f		

-- v_checkplace (vue utile pour vérifier que le nombre de places réservées ne dépasse pas le nombre de places dans le train pour chaque voyage)

nb_places	date_	id_voyage	nb_billets
204	2021-03-27	234	2
204	2020-07-04	234	1

– v_villevoyage (vue utile pour la recherche d'un voyage en fonction de la ville de départ et celle d'arrivée)

num_arret_voyage	voyage	heure_depart	heure_arrivee	arret_ligne	ligne	nom_gare	ville_gare
1	234	12:00:00	11:55:00	1	27	Gare du Nord	Paris
2	234	12:30:00	12:28:00	3	27	Gare ferroviaire	Pont-Sainte-Maxence
3	234	13:05:00	13:00:00	4	27	Gare ferroviaire	Compiegne
4	234	13:22:00	13:20:00	5	27	Gare ferroviaire	Amiens
1	27	18:00:00	17:45:00	1	156	Gare du Nord	Paris
2	27	20:36:00	20:27:00	2	156	Gare Bruxelles-Midi	Bruxelles

4 - Application Python

Dans la partie de l'application en Python, nous abordons les étapes de base des opérations de base de données à l'aide des fonctions Python, notamment la connexion à la base de données, la récupération des données, l'insertion et la mise à jour.

Les opérations sont divisées en deux perspectives fonctionnelles : celle du client et celle du membre de la société. Grâce à ces opérations, nous avons réalisé des fonctionnalités telles que la recherche de places, l'achat de billets et la consultation de l'état des trains.

Avant l'exécution, le fonctionnement sera précédé par l'appel de la fonction *check_bdd()* afin de vérifier les contraintes sur les projections (view).

Voici les étapes à suivre:

A. Choix du profil : voyageur

a. Créer un compte voyageur

La fonction demande à l'utilisateur de fournir les informations nécessaires. Les données saisies sont ensuite vérifiées pour garantir leur exhaustivité et leur validité. Par la suite, la fonction vérifie si le voyageur est déjà présent dans la base de données. Si ce n'est pas le cas, le compte voyageur est ajouté à la base de données.

```
----- Création d'un compte voyageur -----
Nom : Martin
Prénom : Lucie
Adresse : 2 R.Bouvines
Téléphone : 8888888888
Méthode de paiement (carte/cheque/monnaie) : carte
Voyageur occasionnel ? oui/non (entrez autre chose que oui) : oui

Le compte voyageur a été créé avec succès.
```

b. Consulter la liste des voyages

```
Ligne, voyage, numéro d'arrêt, gare de départ, ville de départ, heure de depart, heure d'arrivée
-----------------------------------------------------------------------------------------------
27, 234, 1, Gare du Nord, Paris, 12:00:00, 11:55:00
27, 234, 2, Gare ferroviaire, Pont-Sainte-Maxence, 12:30:00, 12:28:00
27, 234, 3, Gare ferroviaire, Compiegne, 13:05:00, 13:00:00
27, 234, 4, Gare ferroviaire, Amiens, 13:22:00, 13:20:00
156, 27, 1, Gare du Nord, Paris, 18:00:00, 17:45:00
156, 27, 2, Gare Bruxelles-Midi, Bruxelles, 20:36:00, 20:27:00
```

c. Consulter les horaires de trains en fonction de la gare de départ et d'arrivée

```
Votre choix : 4

Entrez la gare de ville de départ : Paris
Entrez la gare de ville d'arrivée : Compiegne

Horaires des trains de Paris à Compiegne :
Voyage ID : 234, Ligne : 27, Départ : 12:00:00, Gare : Gare du Nord, Ville : Par
is, Arrivé : 13:00:00, Gare : Gare ferroviaire, Ville : Compiegne
```

d. Chercher un voyage aller simple en fonction de la date/gare donnée

```
Entrer la date (YYYY-MM-DD) : 2021-03-27
Entrer la ville de depart : Paris
Entrer la ville d'arrivee : Compiegne
Voyages le 2021-03-27 de Paris à Compiegne:
Date : 2021-03-27, Voyage ID : 234, Ligne : 27, Départ : 12:00:00, Gare : Gare d
u Nord, Ville : Paris, Arret : 1, Arrivé : 13:00:00, Gare : Gare ferroviaire, Vi
lle : Compiegne, Arret : 3
```

e. Acheter un billet

* Avec une date acceptée :

```
Votre choix : 2
Achat d'un billet
Entrez votre nom : Martin
Entrez votre prénom : Lucie
Entrez votre adresse : 2 R.Bouvines
Entrez le numéro du voyage : 234
Entrez le numéro de l'arrêt : 1
Entrez le numéro de l'arrêt d'arrivée : 3
Entrer la date (YYYY-MM-DD) : 2021-03-27
Billet acheté avec succès ! (création du trajet associé)
Prix de votre billet: 168.7255446
Numéro (id) du billet: 1687255446
```

* Avec une date refusée :

```
Votre choix : 2
Achat d'un billet
Entrez votre nom : Martin
Entrez votre prénom : Lucie
Entrez votre adresse : 2 R.Bouvines
Entrez le numéro du voyage : 234
Entrez le numéro de l'arrêt : 1
Entrez le numéro de l'arrêt d'arrivée : 3
Entrer la date (YYYY-MM-DD) : 2021-07-14

ERREUR : aucun voyage pour cette date.
```

f. Annuler un billet

```
Votre choix : 6

Veuillez saisir le numéro de billet : 1687775937
Le billet a été annulé dans CompositionBillet avec succès.
Le billet a été annulé dans Billet avec succès.
```

B. Choix du profil : membre de la société

a. Ajouter une gare

```
Votre choix : 1

Gares dans la base de données :
Nom : Gare ferroviaire  Ville : Compiegne
Nom : Gare du Nord      Ville : Paris
Nom : Gare de Lyon      Ville : Paris
Nom : Gare Montparnasse Ville : Paris
Nom : Gare ferroviaire  Ville : Creil
Nom : Gare ferroviaire  Ville : Pont-Sainte-Maxence
Nom : Gare ferroviaire  Ville : Amiens
Nom : Gare Bruxelles-Midi       Ville : Bruxelles
Nom de la gare : Nice-Ville
Ville de la gare : Nice
Adresse : Av.Thiers
Pays : France
Gare ajoutée.
```

b. Ajouter un train

```
Votre choix : 2

Trains dans la base de données :
Numéro : 1      Type de trains : metro
Numéro : 2      Type de trains : metro
Numéro : 3      Type de trains : metro
Numéro : 4      Type de trains : metro
Numéro : 5      Type de trains : metro
Numéro : 9      Type de trains : TGV
Numéro : 34     Type de trains : TGV
Numéro : 45     Type de trains : TGV
Numéro : 345    Type de trains : TER
Numéro : 745    Type de trains : TER
Numéro : 675    Type de trains : TER
Numéro : 7      Type de trains : RER
Numéro du train : 10
Types de train dans la base de données :
TER
TGV
RER
metro
Type de train : TER
Train ajouté.
```

c. Supprimer un train

```
Numéro de train : 10
Train supprimé.
```

d. Modifier le type d'un train

```
Numéro de train : 10
Types de train dans la base de données :
TER
TGV
RER
metro
Type de train : TER
Type du train modifié.
```

e. Statistiques sur la société

```
Votre choix : 5

Voici le nombre de trajets par date :
        Date : 2021-03-27       Nombre de trajets : 1
        Date : 2020-07-04       Nombre de trajets : 2
        Date : 2022-01-01       Nombre de trajets : 1

Nombre de voyages par ligne de train :
        Ligne : 156     Nombre de voyages : 1
        Ligne : 27      Nombre de voyages : 1

Nombre de voyages par jour de la semaine :
        Jour : Lundi    Nombre de voyages : 2

Argent gagné par la société : 550.48

Somme des prix des billets par voyageur :
        Nom : Martin    Prénom : Lucie  Adresse : 2 R.b Argent dépensé :  337.56
        Nom : Smith     Prénom : Henry  Adresse : 288 avanue du G,n,ral de Gaulle   Argent dépensé :  212.93

Voyageurs ayant le statut bronze :
        Nom : Smith     Prénom : Henry  Adresse : 288 avanue du G,n,ral de Gaulle

Taux de remplissage des trains :
        Numéro de voyage : 234  Date : 2022-01-01       Taux de remplissage : 0.98
        Numéro de voyage : 234  Date : 2021-03-27       Taux de remplissage : 1.96
        Numéro de voyage : 234  Date : 2020-07-04       Taux de remplissage : 0.49
```

III – Base de données non relationnelle

1 - SQL

Dans la partie non relationnelle de notre travail, nous avons utilisé PostgreSQL pour effectuer des manipulations relationnelles avec JSON. Nous avons pris la décision de convertir certains attributs en type JSON, ce qui nous a permis d'insérer des données au format JSON dans une table. Ci-dessous, nous présentons les attributs que nous avons identifiés comme pouvant bénéficier de cette modification :

- Attributs composés :
  - L'attribut « **adresse** » a été transformé en un attribut composé pour « Hotel », « Taxi », « Transport_public » et « Gare ».

- Attributs multivalués :
  - Nous avons supprimé les tables « Hotel », « Taxi » et « Transport_public » afin de regrouper les informations dans la table « Gare ». Plus précisément, nous avons introduit trois attributs multivalués dans la table « Gare » pour représenter les données relatives aux hôtels, aux taxis et aux transports publics.
  - Exemple d'insertion :

```
INSERT INTO Gare VALUES ('Gare ferroviaire', 'Compiegne',
    '{
        "numero": 1,
        "rue": "avenue des papillons",
        "cp": "60200",
        "pays": "France"
    }',
    '[
        {
            "nom":"B&B",
            "adresse":"10 avenue Marcellin Berthelot"
            },
        {
        "nom":"Ritz", "adresse":"15 place Vendôme"
        }
    ]',
    '[
        {
            "num":1096,
            "tel":"0654782945"
        }
    ]',
    NULL
);
```

- « **Calendrier** » :
  - Dans la table « **Calendrier** », nous avons procédé à des transformations concernant les jours de la semaine. Nous avons remplacé les attributs

BOOLEAN « lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche » par « jours ».

➢ Exemple d'insertion :

```
630 INSERT INTO Billet VALUES (1, FALSE, 34.70,
631     '{
632         "nom" : "Smith",
633         "prenom" : "Henry",
634         "adresse" : {"num" : 288, "rue" : "avenue du Général de Gaulle"},
635         "telephone" : "0614263798",
636         "paiement" : "monnaie",
637         "carte" : 563,
638         "statut" : "bronze"
639     }'
640 );
```

Dans le cadre de ces modifications, nous avons également adapté les requêtes SELECT correspondantes. Voici les résultats obtenus :

--Affiche la somme des prix des billets par voyageur (SELECT SUM)

```
postgres-# GROUP BY voyageur_nom, voyageur_prenom, voyageur_adresse;
 voyageur_nom | voyageur_prenom |                  voyageur_adresse                  | somme_prix
--------------+-----------------+----------------------------------------------------+------------
 Smith        | Henry           | {"num" : 288, "rue" : "avenue du Général de Gaulle"} |      44.15
(1 ligne)
```

--Afficher le nombre de voyages par jour de la semaine (SELECT CASE)

```
 jour_semaine | nombre_voyages
--------------+----------------
 dimanche     |              1
 lundi        |              2
 samedi       |              2
 vendredi     |              2
 mercredi     |              2
 jeudi        |              2
 mardi        |              2
(7 lignes)
```

--Affiche le nom/prenom/adresse des voyageurs ayant le statut bronze (SELECT WHERE)

```
 voyageur_nom | voyageur_prenom |                  voyageur_adresse
--------------+-----------------+----------------------------------------------------
 Smith        | Henry           | {"num" : 288, "rue" : "avenue du Général de Gaulle"}
(1 ligne)
```

2 - Application Python

Par la suite, nous avons remplacé les anciennes fonctions de Python par celles qui ont été modifiées en JSON.

A. Fonctions *creer_compte_voyageur()* et *achat_billet()*

Ces fonctions permettent de créer un billet pour un voyageur. La fonction *creer_compte_voyageur()* ne va plus créer une nouvelle relation dans la table Voyageur puisqu'elle n'existe plus, mais est appelée pat *achat_billet()* pour créer le JSON correspondant aux informations du voyageur.

Étapes pour créer un billet :

1. Choix du profil : n°1 pour l'interface client
2. Choix de l'action: n°1 pour acheter un billet
3. Le client choisit ses options en fonction des informations qui suivent :

* Numéro de l'arrêt :

num_arret_voyage	voyage	heure_depart	heure_arrivee	arret_ligne	ligne	nom_gare	ville_gare
1	234	12:00:00	11:55:00	1	27	Gare du Nord	Paris
2	234	12:30:00	12:28:00	3	27	Gare ferroviaire	Pont-Sainte-Maxence
3	234	13:05:00	13:00:00	4	27	Gare ferroviaire	Compiegne
4	234	13:22:00	13:20:00	5	27	Gare ferroviaire	Amiens
1	27	18:00:00	17:45:00	1	156	Gare du Nord	Paris
2	27	20:36:00	20:27:00	2	156	Gare Bruxelles-Midi	Bruxelles

* Date du trajet :

```
postgres=# select * from v_CheckDate;
 id_trajet | trajet_date | id_calendrier | date_debut |  date_fin  |                                  jours                                  |
-----------+-------------+---------------+------------+------------+-------------------------------------------------------------------------+-
         1 | 2021-03-27  |             1 | 2020-01-01 | 2022-12-31 | ["lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche"] |
         1 | 2021-03-27  |             1 | 2020-01-01 | 2022-12-31 | ["lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche"] |
         2 | 2020-07-04  |             1 | 2020-01-01 | 2022-12-31 | ["lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche"] |
         2 | 2020-07-04  |             1 | 2020-01-01 | 2022-12-31 | ["lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche"] |
         3 | 2020-07-04  |             2 | 1999-11-24 | 2016-02-14 | ["lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi"]           |
         3 | 2020-07-04  |             2 | 1999-11-24 | 2016-02-14 | ["lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi"]           |
(6 lignes)
```

* Exemple d'entrée :

```
----- Achat d'un billet -----
Entrez le numéro du voyage : 234
Entrez le numéro de l'arrêt : 1
Entrez le numéro de l'arrêt d'arrivée : 2
Entrer la date (YYYY-MM-DD) : 2021-03-27
Voulez-vous souscrire à l'assurance (1 pour oui/0 pour non) :1
Nom : Martin
Prénom : Lucie
Adresse (pays) : France
Adresse (ville) : Compiegne
Adresse (rue) : R.Bouvines
Adresse (numéro) : 2
Téléphone : 8888888888
Méthode de paiement (carte/cheque/monnaie) : carte
Voyageur occasionnel ? oui/non (entrez autre chose que oui) : oui
Billet acheté avec succès ! (création du trajet associé)
Prix de votre billet: 168.7253454
Numéro (id) du billet: 1687253454
```

B. Fonction *ajouter_gare()*

Cette fonction permet d'ajouter une gare.

Étapes pour ajouter une gare :

1. Choix du profil : n°2 pour l'interface d'un membre de la société
2. Choix de l'action : n°1 pour ajouter une gare
3. L'employé ajoute ensuite la nouvelle gare (exemple avec Nice-Ville) :

```
Gares dans la base de données :
Nom : Gare ferroviaire  Ville : Compiegne
Nom : Gare du Nord      Ville : Paris
Nom : Gare de Lyon      Ville : Paris
Nom : Gare Montparnasse Ville : Paris
Nom : Gare ferroviaire  Ville : Creil
Nom : Gare ferroviaire  Ville : Pont-Sainte-Maxence
Nom : Gare ferroviaire  Ville : Amiens
Nom : Gare Bruxelles-Midi       Ville : Bruxelles
Nom de la gare : Nice-Ville
Ville de la gare : Nice
Adresse (rue): Av.Thiers
Adresse (numéro): 1
Gare ajoutée.
```

C. Fonctions dans les statistiques sur la société

Fonctions modifiées :

- *argent_par_voyageur()* : Affiche la somme des prix des billets par voyageur.
- *voyageur_bronze()* : Affiche le nom, le prénom et l'adresse des voyageurs ayant le statut bronze.
- *nb_voyages_par_jour()* : Affiche le nombre de voyages effectués par jour de la semaine.

Étapes pour consulter les statistiques :

1. Choix du profil : n°2 pour l'interface d'un membre de la société
2. Choix de l'action : n°5 pour consulter les statistiques sur la société
3. Résultats obtenus :

```
Voici le nombre de trajets par date :
        Date : 2021-03-27       Nombre de trajets : 2
        Date : 2020-07-04       Nombre de trajets : 2

Nombre de voyages par ligne de train :
        Ligne : 156     Nombre de voyages : 1
        Ligne : 27      Nombre de voyages : 1

Nombre de voyages par jour de la semaine :
        Jour : dimanche Nombre de voyages : 1
        Jour : lundi    Nombre de voyages : 2
        Jour : samedi   Nombre de voyages : 2
        Jour : vendredi Nombre de voyages : 2
        Jour : mercredi Nombre de voyages : 2
        Jour : jeudi    Nombre de voyages : 2
        Jour : mardi    Nombre de voyages : 2

Argent gagné par la société : 212.87534540000001

Somme des prix des billets par voyageur :
        Nom : Smith     Prénom : Henry  Adresse : {"num" : 288, "rue" : "avenue du G,n,ral de Gaulle"}  Argent dépensé : 44.150000000000006
        Nom : Martin    Prénom : Lucie  Adresse : {"pays": "France", "ville": "Compiegne", "rue": "R.Bouvines", "num": "2"}     Argent dépensé : 168.7253454

Voyageurs ayant le statut bronze :
        Nom : Smith     Prénom : Henry  Adresse : {"num" : 288, "rue" : "avenue du G,n,ral de Gaulle"}

Taux de remplissage des trains :
        Numéro de voyage : 234  Date : 2021-03-27       Taux de remplissage : 1.96
        Numéro de voyage : 234  Date : 2020-07-04       Taux de remplissage : 0.49
```

IV – Conclusion

Ce projet nous a permis de compléter nos compétences théoriques par des compétences pratiques sur un sujet complexe. Nous avons pu réaliser une base de données, d'abord relationnelle puis non relationnelle, de sa conception, à son implémentation et son utilisation à travers une application.

Nous avons conscience qu'elle est encore perfectible : certaines choses évoquées en cours n'ont pas été appliquées (notamment les triggers et l'incrémentation automatique des clés artificielles), et des mécanismes pour harmoniser les données entrées dans la base (par exemple, données uniquement en minuscules) pourraient être ajoutés.

De plus, pour ce qui est de la seconde partie du projet (modélisation de la base de données en NoSQL), les changements effectués sont avant tout pour s'exercer au non relationnel. Il convient que dans la réalité, une base de données en non relationnel ne serait pas adaptée pour la gestion des trains où il faut être précis pour éviter les erreurs et les incohérences. Par exemple, le fait de supprimer la table Voyageur pour la mettre dans la table Billet génère des imprécisions et empêche, à l'avenir, de manipuler les voyageurs sans passer par leurs billets. De plus, l'attribut qui regroupe les informations d'un voyageur est en format JSON. Si l'application est mal programmée, il est possible de se retrouver avec un voyageur qui n'a ni nom, ni prénom, là où avec des clés, il serait impossible de créer un billet sans ces données.

Par ailleurs, en tant qu'ingénieur en informatique, il est nécessaire de vérifier la fiabilité et la sécurité de ses applications. Pour notre part, nous avons conscience qu'il aurait été intéressant d'optimiser notre travail et de le rendre plus fiable et plus sécurisé (par exemple contre les attaques par injection).

Enfin, même si notre base de données doit être améliorée, nous sommes malgré tout satisfaites de ce que nous avons pu effectuer en peu de temps. Nous avons réussi, par exemple, à modéliser au mieux ce qu'attendait le client, notamment au niveau de l'UML avec l'implémentation des tables intermédiaires telles que « ArretLigne » et « ArretVoyage » qui permettent plus de précision et de justesse.