



Dorian CHENET
Hugo DUCLY
Groupe projet A8

Rapport de projet

BDD/Réseau

TRAKS

Version 4

29 Novembre 2019

Table des matières

Présentation	2
Architecture contextualisée	3
Diagrammes applicatifs	4
Dictionnaire de données	6
Schéma E/A	8
Schéma relationnel	9
Extrait représentatif du jeu de données	10
Requêtes sur la base de données	16

Présentation

Le but de notre projet était de créer une plateforme permettant de visualiser les trajets disponibles d'un réseau ferré et de pouvoir réserver un trajet. Nous avons fait le choix de cette modélisation afin de

La base de donnée contiendra les informations concernant les trajets prévus ou terminés entre différentes gares, les informations des voyageurs et leur réservations. Nous avons choisi de supposer que tous les trajets sont stockés dans la base et que de ce fait on connaît à l'avance toutes les informations les concernant. Un trajet stocké est un consommable qui correspond à un horaire donné, un train donné, un départ et une arrivée donnés. La base de données contiendra aussi les incidents ayant lieu sur un trajet. Chaque client qui souhaite réserver doit détenir un compte qui renseigne différentes informations et doit opérer un paiement, qui est enregistré aussi. Il doit aussi rester de la place pour le trajet sélectionné.

Compte tenu de cette modélisation, la page internet sera une page de consultation des trajets existants selon le critère de recherche que l'on souhaite utiliser, c'est à dire soit juste le départ et l'arrivée, ou bien aussi la date et l'heure du trajet. Il sera aussi possible de le réserver et d'opérer un paiement (factice).

Pour la partie réseau, on décide qu'un personnel utilise un logiciel client qui communique avec un serveur, lui même client de la base de données afin d'y insérer les différents signalements d'incidents qu'on lui a transmis et de communiquer sa résolution dans un second temps.

Architecture contextualisée

Nous avons utilisé l'architecture conseillée par l'énoncé du sujet, à savoir un serveur TCP en langage C, lui-même client de notre SGBD, un client TCP en langage Java, et un site web en PHP. Ci-après un schéma la présentant.

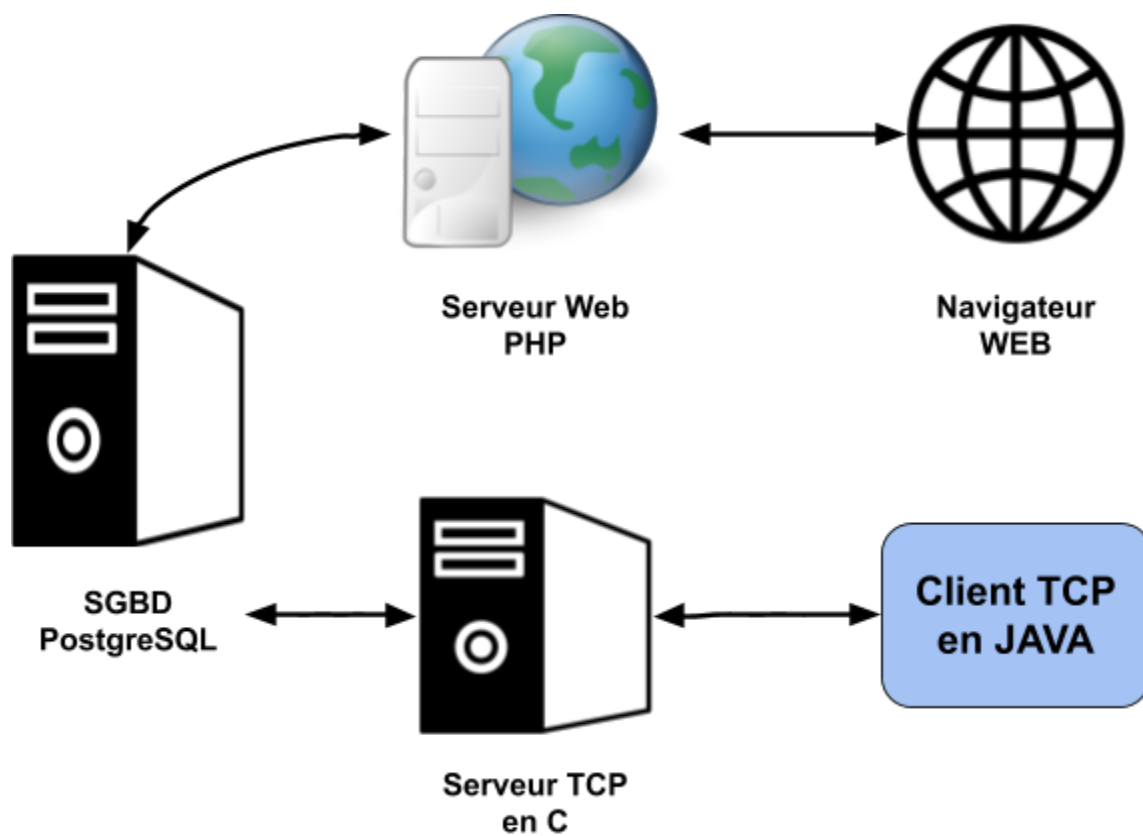
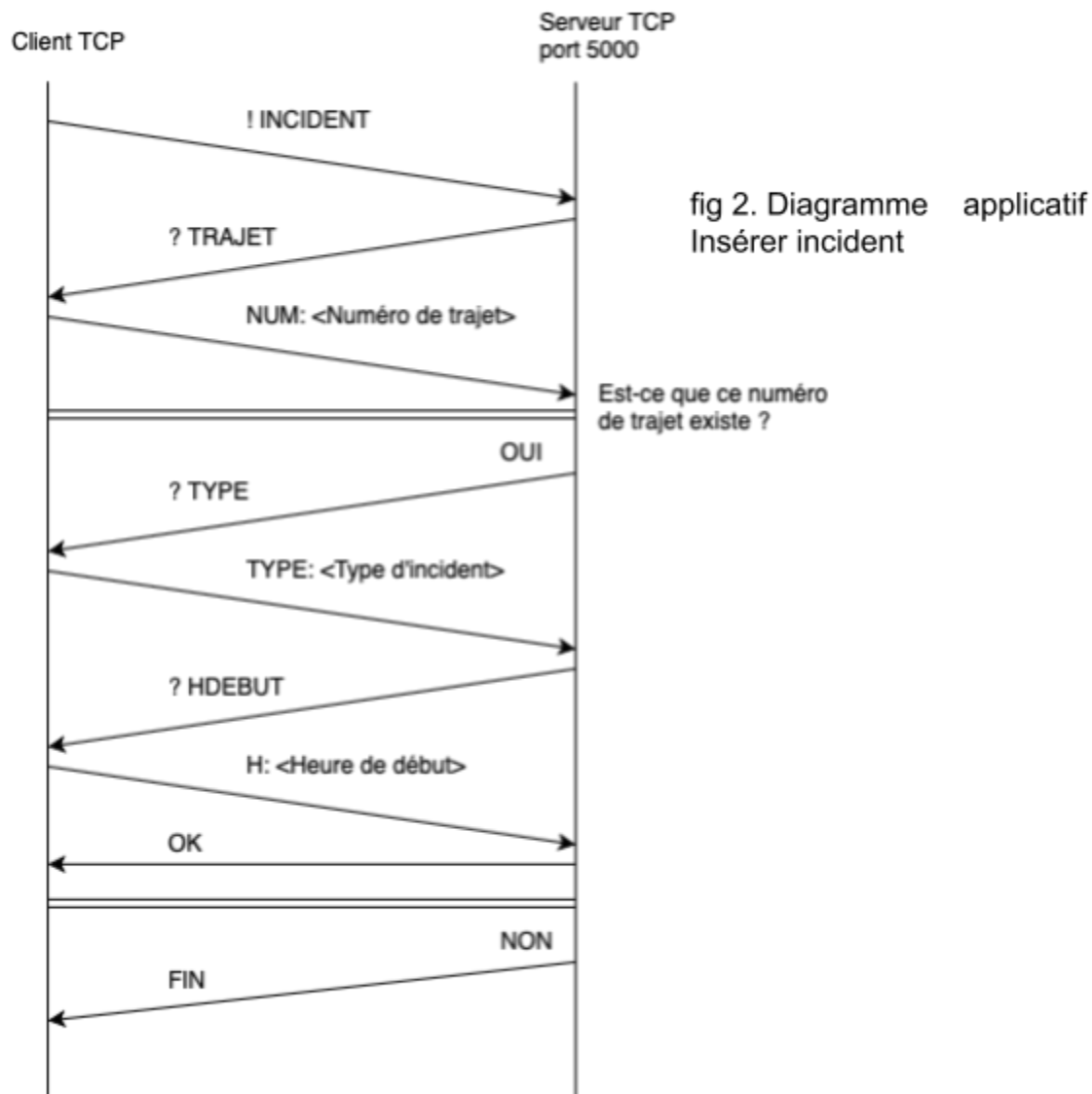


fig 1 : Architecture contextualisée

Diagrammes applicatifs



Le diagramme ci-dessus correspond à l'échange qui se produira lorsque l'opérateur voudra signaler un nouvel incident. Ci-après un diagramme avec la même structure mais pour la mise à jour d'un incident déjà signalé, pour dire qu'il est résolu.

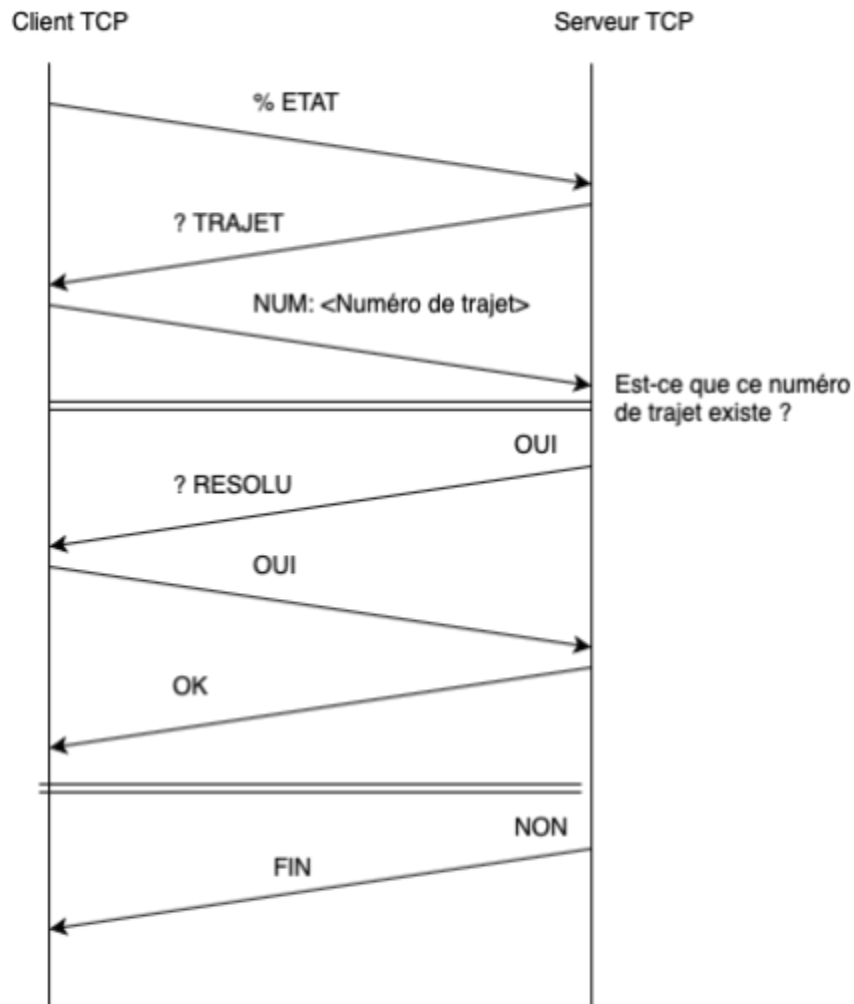


fig 3. Diagramme applicatif Mise à jour d'un incident

Quand on veut enregistrer un nouvel incident, le serveur a besoin du numéro du trajet, du type de l'incident, ainsi que l'heure à laquelle il s'est produit. Il va ensuite insérer un enregistrement correspondant à l'incident dans la table *incident* et un dans la table *se_produit*. Le serveur vérifie avant tout que le numéro de trajet existe bien avant d'aller écrire dans la base de données. De même, si une commande envoyée n'est pas reconnue ou s'il y a un quelconque problème de communication du côté serveur ou bien du côté client, l'autre le détecte et s'arrête. Le serveur se relance automatiquement, quant-au client, on

peut à nouveau utiliser le bouton dédié pour se reconnecter. Cela fonctionne de la même façon pour la mise à jour, sauf que le serveur n'a besoin que du numéro de trajet et de l'heure à laquelle il s'est produit. Le serveur sait si on veut créer un nouvel incident ou en mettre à jour un autre grâce au premier message. Dans les deux cas, une fin sans erreur se termine par le message 'OK' et une fin pour une autre raison se termine par le message 'FIN'.

Dictionnaire de données

Nom symbolique	Description	Domaine/ Type	Remarques éventuelles
no_client	n° du client	int	
mdp	mot de passe du client	varchar(50)	
mail	mail du client	varchar(50)	unique
nom	nom du client qui veut voyager	varchar(50)	
prenom	prénom du client qui veut voyager	varchar(50)	
nb_voyageur	nombre de voyageur pour lesquels il veut prendre des billets	int	
nb_bagage	nombre de bagages que le passager souhaite prendre	int	
moyen_paiement	moyen de paiement choisi par le client	varchar(50)	
montant	montant du paiement	int	
no_reservation	numéro de la réservation du client	int	
date_reservation	date à laquelle la	date	

	réserveation a été faite		
no_trajet	numéro d'un trajet	int	
date_trajet	date d'un trajet	date	
heure_depart	heure du depart	time	
heure_arrivee	heure de l'arrivée	time	
prix	prix du trajet	int	
etat	définit si le trajet est en cours, à venir ou passé	varchar(20)	
no_train	numéro du train emprunté	int	
modele	modèle du train	varchar(50)	
conducteur	identite du conducteur	varchar(100)	
capacite	nombre de places disponibles dans un train	int	avant de proposer une réservation à un client ou de l'accepter, on vérifie qu'il reste de la place dans le train
etat_train	indique l'état du train	varchar(20)	
quai_depart	indique à quel quai de la gare le trajet va partir	int	
quai_arrive	indique à quel quai le trajet va arrivé	int	
nom_gare	nom de la gare	varchar(50)	
nb_quais	nombre de quai d'une gare	int	
no_incident	numéro d'un incident	int	
heure_incident	heure de l'incident	time	
type_incident	type d'incident	varchar(50)	
resolu	indique si l'incident est clos	boolean	

	ou en cours		
--	-------------	--	--

Schéma E/A

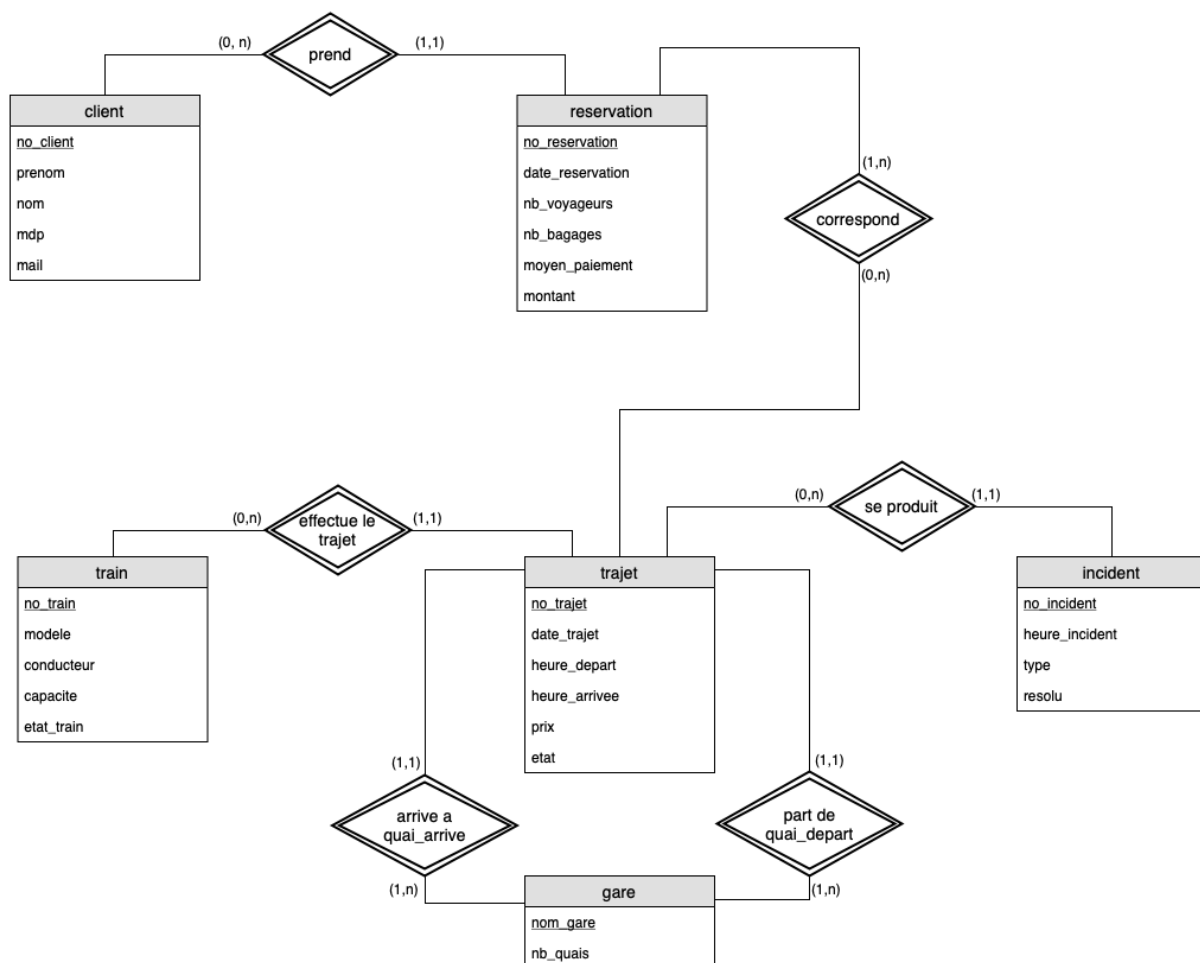


fig 4. E/A

Schéma relationnel

client (no_client, prenom, nom, mdp, mail)

reservation (no_reservation, date_reservation, nb_voyageurs, nb_bagages, moyen_paiement, montant, #no_client)

trajet (no_trajet, date_trajet, heure_depart, heure_arrivee, prix, etat)

gare (nom_gare, nb_quais)

train (no_train, modele, conducteur, capacite, etat_train)

part_de (#nom_gare_depart, #no_trajet, quai_depart)

arrive_a (#nom_gare_arrivee, #no_trajet, quai_arrivee)

effectue_le_trajet (#no_train, #no_trajet)

se_produit (#no_incident, #no_trajet)

correspond (#no_reservation, #no_trajet)

Extrait représentatif du jeu de données

Valeurs de la table **client** :

(1, 'Hugo', 'DUCLY', 'azerty', 'hugo.ducly@gmail.com'),
(2, 'Dorian', 'CHENET', 'azerty2', 'dorian.chenet@gmail.com'),
(3, 'Albert', 'TABLE', 'azerty3', 'albert.atable@gmail.com');

Valeurs de la table **trajet** :

(1, '2020-01-01', '09:30:00', '10:45:00', 20, 'A_VENIR'),
(2, '2020-01-01', '10:46:00', '12:00:00', 30, 'A_VENIR'),
(3, '2020-01-01', '09:30:00', '10:45:00', 25, 'A_VENIR'),
(4, '2020-01-01', '10:46:00', '12:00:00', 50, 'A_VENIR'),
(5, '2020-01-01', '07:00:00', '07:10:00', 10, 'A_VENIR'),
(6, '2020-01-01', '07:10:00', '07:15:00', 10, 'A_VENIR'),
(7, '2020-01-01', '07:15:00', '07:20:00', 10, 'A_VENIR'),
(8, '2020-01-01', '09:00:00', '09:10:00', 10, 'A_VENIR'),
(9, '2020-01-01', '09:11:00', '09:20:00', 10, 'A_VENIR'),
(10, '2020-01-01', '09:20:00', '09:30:00', 10, 'A_VENIR'),
(11, '2020-01-02', '09:30:00', '10:45:00', 20, 'A_VENIR'),
(12, '2020-01-02', '10:46:00', '12:00:00', 30, 'A_VENIR'),
(13, '2020-01-02', '09:30:00', '10:45:00', 25, 'A_VENIR'),
(14, '2020-01-02', '10:46:00', '12:00:00', 50, 'A_VENIR'),
(15, '2020-01-02', '07:00:00', '07:10:00', 10, 'A_VENIR'),
(16, '2020-01-02', '07:10:00', '07:15:00', 10, 'A_VENIR'),
(17, '2020-01-02', '07:15:00', '07:20:00', 10, 'A_VENIR'),
(18, '2020-01-02', '09:00:00', '09:10:00', 10, 'A_VENIR'),

```
(19, '2020-01-02', '09:11:00', '09:20:00', 10, 'A_VENIR'),
(20, '2020-01-02', '09:20:00', '09:30:00', 10, 'A_VENIR'),
(21, '2019-12-01', '09:30:00', '10:45:00', 20, 'PASSE'),
(22, '2019-12-02', '09:20:00', '10:45:00', 20, 'PASSE');
```

Valeurs de la table **incident** :

```
(1, '2019-12-01', '09:32:00', 'malaise voyageur', TRUE),
(2, '2019-12-01', '09:34:00', 'malaise voyageur', TRUE),
(3, '2019-12-01', '09:36:00', 'animal sur les voies', TRUE),
(4, '2019-12-2', '09:42:00', 'conducteur evanoui', TRUE),
(5, '2019-12-2', '09:44:00', 'malaise voyageur', TRUE);
```


Valeurs de la table **se_produit** :

```
(1, 21),
(2, 21),
(3, 21),
(4, 22),
(5, 22);
```

Valeurs de la table **reservation** :

```
(1, '2020-01-01', 1, 1, 'CB', '120'),
(2, '2020-01-01', 1, 2, 'PAYPAL', '100'),
(3, '2020-01-01', 1, 2, 'PAYPAL', '100'),
(4, '2020-01-02', 1, 1, 'CB', '120'),
(5, '2020-01-02', 1, 2, 'PAYPAL', '100'),
(6, '2020-01-02', 1, 2, 'PAYPAL', '100'),
(7, '2019-12-01', 1, 2, 'PAYPAL', '100');
```

Valeurs de la table **correspond** :



(1, 1),
(2, 2),
(3, 3),
(1, 4),
(2, 5),
(3, 6),
(22, 7);

Valeurs de la table **gare** :

('GARE DU NORD', 10),
('LILLE', 5),
('ANGOULEME', 5),
('BORDEAUX', 5),
('CAEN', 5),
('PERSAN BEAUMONT', 5),
('CHAMPAGNE', 5),
('VALMONDOIS', 5),
('PONTOISE', 5),
('CHERBOURG', 5),
('CERGY PREF', 5),
('CONFLANS', 5),
('NANTERRE', 5),
('CHATELET', 5),
('ARRAS', 5);

Valeurs de la table **part_de** :

('GARE DU NORD', 1, 1),

('ARRAS', 2, 1),
('GARE DU NORD', 3, 1),
('CAEN', 4, 1),
('PERSAN BEAUMONT', 5, 1),
('CHAMPAGNE', 6, 1),
('VALMONDOIS', 7, 1),
('CERGY PREF', 8, 1),
('CONFLANS', 9, 1),
('NANTERRE', 10, 1),
('GARE DU NORD', 11, 1),
('ARRAS', 12, 1),
('GARE DU NORD', 13, 1),
('CAEN', 14, 1),
('PERSAN BEAUMONT', 15, 1),
('CHAMPAGNE', 16, 1),
('VALMONDOIS', 17, 1),
('CERGY PREF', 18, 1),
('CONFLANS', 19, 1),
('NANTERRE', 20, 1),
('NANTERRE', 21, 1),
('PERSAN BEAUMONT', 22, 1);

Valeurs de la table **arrive_a** :

('ARRAS', 1, 1),
('LILLE', 2, 1),
('CAEN', 3, 1),
('CHERBOURG', 4, 1),
('CHAMPAGNE', 5, 1),


('VALMONDOIS', 6, 1),
('PONTOISE', 7, 1),
('CONFLANS', 8, 1),
('NANTERRE', 9, 1),
('CHATELET', 10, 1),
('ARRAS', 11, 1),
('LILLE', 12, 1),
('CAEN', 13, 1),
('CHERBOURG', 14, 1),
('CHAMPAGNE', 15, 1),
('VALMONDOIS', 16, 1),
('PONTOISE', 17, 1),
('CONFLANS', 18, 1),
('NANTERRE', 19, 1),
('CHATELET', 20, 1),
('CHATELET', 21, 1),
('GARE DU NORD', 22, 1);

Valeurs de la table **train** :

(1, 'TER', 'Andre ARGH', 3, TRUE),
(2, 'TGV', 'Gilbert OUILLE', 3, TRUE),
(3, 'TGV', 'Almamy CAMERA', 3, TRUE),
(4, 'RER', 'Alexis CHALET', 3, TRUE);

Valeurs de la table **effectue_trajet** :

(2, 1),
(2, 2),
(3, 3),



(3, 4),
(1, 5),
(1, 6),
(1, 7),
(4, 8),
(4, 9),
(4, 10),
(2, 11),
(2, 12),
(3, 13),
(3, 14),
(1, 15),
(1, 16),
(1, 17),
(4, 18),
(4, 19),
(4, 20),
(1, 21)

Requêtes sur la base de données

Requête 1 :

Cette requête permet de récupérer les informations du ou des trajets correspondant à la gare de départ et de destination que l'on choisit. Ici on choisit les gares de CAEN et GARE DU NORD.

```
SELECT * FROM
(SELECT no_trajet FROM (part_de NATURAL JOIN arrive_a) WHERE
nom_gare_arrivee = 'CAEN' AND
nom_gare_depart = 'GARE DU NORD') AS bon_trajet
NATURAL JOIN trajet ORDER BY date_trajet ASC;
```

```
postgres=# SELECT * FROM
(SELECT no_trajet FROM (part_de NATURAL JOIN arrive_a) WHERE
nom_gare_arrivee = 'CAEN' AND
nom_gare_depart = 'GARE DU NORD') AS bon_trajet
NATURAL JOIN trajet ORDER BY date_trajet ASC;
 no_trajet | date_trajet | heure_depart | heure_arrivee | prix | etat
-----+-----+-----+-----+-----+-----
          3 | 2020-01-01 | 09:30:00    | 10:45:00     |    25 | A_VENIR
         13 | 2020-01-02 | 09:30:00    | 10:45:00     |    25 | A_VENIR
(2 rows)
```

Requête 2 :

Si on veut connaître le nombre de places restantes on utilise une requête imbriquée pour calculer la différence entre la capacité du train qui réalise le trajet et le nombre de réservations de ce trajet. La requête concerne le trajet numéro 1, qui a 2 réservations avec un train de capacité 3, on devrait donc trouver 1.

```
SELECT (capacite - somme) AS nombre_places_restantes FROM
(SELECT capacite FROM effectue_trajet NATURAL JOIN train WHERE no_trajet = 1) AS c,
(SELECT COUNT(no_reservation) AS somme FROM
correspond WHERE no_trajet = 1) AS r;
```

```

postgres=# SELECT (capacite - somme) AS nombre_places_restantes FROM
postgres=# (SELECT capacite FROM effectue_trajet NATURAL JOIN train WHERE no_trajet = 1) AS c,
postgres=# (SELECT COUNT(no_reservation) AS somme FROM correspond WHERE no_trajet = 1) AS r;
 nombre_places_restantes
-----
1
(1 row)

```

Requête 3 :

On utilise la requête suivante pour classer les trajets par leur nombre d'accidents en les groupant par gare de départ et destination. On peut ainsi savoir quelles ligne est la plus propice aux accidents.

```

SELECT COUNT(no_incident) AS nombre_incidents, nom_gare_depart AS gare_depart,
nom_gare_arrivee AS gare_arrivee FROM
se_produit NATURAL JOIN trajet NATURAL JOIN arrive_a NATURAL JOIN part_de
GROUP BY nom_gare_depart, nom_gare_arrivee
ORDER BY nombre_incidents DESC;

```

```

postgres=# SELECT COUNT(no_incident) AS nombre_incidents, nom_gare_depart AS gare_depart, nom_gare_arrivee AS gare_arrivee FROM
se_produit NATURAL JOIN trajet NATURAL JOIN arrive_a NATURAL JOIN part_de
GROUP BY nom_gare_depart, nom_gare_arrivee
ORDER BY nombre_incidents DESC;
 nombre_incidents | gare_depart | gare_arrivee
-----+-----+-----
3 | NANTERRE | CHATELET
2 | PERSAN BEAUMONT | GARE DU NORD
(2 rows)

```

Requête 4 :

On utilise cette requête pour connaître les recettes de l'entreprise classées par mois et années .

```
SELECT SUM(montant) AS recette, EXTRACT(MONTH FROM date_reservation) AS mois, EXTRACT(YEAR FROM date_reservation) AS annee FROM reservation
GROUP BY mois, annee
ORDER BY mois, annee DESC;
```

```
postgres=# SELECT SUM(montant) AS recette, extract(MONTH from date_reservation)
AS mois , extract(YEAR from date_reservation) AS annee
FROM reservation
GROUP BY mois, annee
ORDER BY mois, annee DESC;
recette | mois | annee
-----+-----+-----
      640 |    1 | 2020
      100 |   12 | 2019
(2 rows)
```

Requête 5 :

Cette requête est utilisée pour récupérer les différentes réservations d'un client à partir de son numéro de client.

```
SELECT date_reservation, nb_voyageurs, nb_bagages, montant, moyen_paiement,
date_trajet, heure_depart, heure_arrivee, nom_gare_depart, nom_gare_arrivee,
quai_depart, quai_arrivee FROM
(SELECT * FROM (SELECT no_reservation, date_reservation, nb_voyageurs, nb_bagages,
montant, moyen_paiement FROM reservation WHERE no_client = 1) AS j1
NATURAL JOIN correspond NATURAL JOIN trajet NATURAL JOIN part_de NATURAL
JOIN arrive_a) AS j2;
```

date_reservation	nb_voyageurs	nb_bagages	montant	moyen_paiement	date_trajet	heure_depart	heure_arrivee	nom_gare_depart
nom_gare_arrivee	quai_depart	quai_arrivee						
2020-01-01	1	1	120	CB	2020-01-01	09:30:00	10:45:00	GARE DU NORD
ARRAS	1	1						
2020-01-01	1	2	100	PAYPAL	2020-01-01	10:46:00	12:00:00	ARRAS
LILLE	1	1						
2020-01-01	1	2	100	PAYPAL	2020-01-01	09:30:00	10:45:00	GARE DU NORD
CAEN	1	1						

(3 rows)

Requête 6 :

Cette requête sert à visualiser le temps de travail réalisé par un conducteur donné pour chaque période d'un mois.

```
SELECT EXTRACT(YEAR FROM date_trajet) AS annee,
EXTRACT(MONTH FROM date_trajet) AS mois,
SUM(heure_arrivee - heure_depart) AS temps_de_travail FROM
trajet NATURAL JOIN effectue_trajet NATURAL JOIN train
WHERE conducteur = 'Andre ARGH'
GROUP BY mois, annee
```

```
postgres=# SELECT EXTRACT(YEAR FROM date_trajet) AS annee,
EXTRACT(MONTH FROM date_trajet) AS mois,
SUM(heure_arrivee-heure_depart) AS temps_de_travail FROM
trajet NATURAL JOIN effectue_trajet NATURAL JOIN train
WHERE conducteur = 'Andre ARGH'
GROUP BY mois, annee;
 annee | mois | temps_de_travail
-----+-----+-----
  2020 |    1 | 00:40:00
  2019 |   12 | 02:40:00
(2 rows)
```

Requête 7 :

Cette requête sert à connaître le trafic d'une gare.

```
SELECT COUNT(no_trajet) AS nbre, modele AS type_train,
EXTRACT(MONTH FROM date_trajet) AS mois , EXTRACT(YEAR FROM date_trajet) AS annee FROM
arrive_a NATURAL JOIN part_de NATURAL JOIN train NATURAL JOIN effectue_trajet
NATURAL JOIN trajet
WHERE nom_gare_arrivee = 'GARE DU NORD' OR nom_gare_depart = 'GARE DU
NORD'
GROUP BY type_train, mois, annee;
```

```
postgres=# SELECT COUNT(no_trajet) AS nbre, modele AS type_train,
extract(MONTH from date_trajet) AS mois , extract(YEAR from date_trajet) AS annee FROM
arrive_a NATURAL JOIN part_de NATURAL JOIN train NATURAL JOIN effectue_trajet NATURAL JOIN trajet
WHERE nom_gare_arrivee = 'GARE DU NORD' OR nom_gare_depart = 'GARE DU NORD'
GROUP BY type_train, mois, annee;
 nbre | type_train | mois | annee
-----+-----+-----+-----
    1 | TER        |    12 | 2019
    4 | TGV        |     1 | 2020
(2 rows)
```

Requête 8 :

Cette requête sert à récupérer la liste des clients par ordre décroissant d'argent dépensé en billets..

```
SELECT prenom, nom, SUM(montant) AS depenses FROM
client NATURAL JOIN reservation
GROUP BY nom, prenom
ORDER BY depenses DESC;
```

```
postgres=# SELECT prenom, nom, SUM(montant) AS depenses FROM
client NATURAL JOIN reservation
GROUP BY nom, prenom
ORDER BY depenses DESC;
 prenom | nom   | depenses
-----+-----+-----
Hugo    | DUCLY |      320
Dorian  | CHENET |      220
Albert  | TABLE |      200
(3 rows)
```