PROJET CUBE
DONNEES ET MODELISATION
AGENCE DE VOYAGE

SOMMAIRE

INTRODUCTION

Dans le cadre de la validation du module Modélisation de base de données, via le projet CUBE en groupe de 3, nous avons travaillé sur la conception d'une base de données pour une agence de voyage fictive. Nous nous sommes donc basés sur le cahier des charges fourni pour réaliser cet exercice, mais avons créer de A à Z le système en réfléchissant à un modèle logique et fonctionnel qui pourrait être utilisé dans une agence de voyage réelle.

La gestion de projet étant à l'honneur, nous avons décidé de nommer un chef d'équipe pour guider, aider et répartir équitablement le travail au sein de son équipe. C'est Maewenn qui aura hérité de ce rôle durant le projet.

Selon le cahier des charges et les demandes spécifiques nous avons créé un dictionnaire de données, réfléchit aux dépendances fonctionnelles et conçu un modèle conceptuel de données (MCD). De ce MCD est ressorti notre modèle logique de données qui nous a permis de finaliser et d'implanter notre base de données dans l'outil Workbench de MySQL. Nous avons également généré et créer des données fictives pour vérifier que les procédures et les requêtes créées fonctionnaient.

Nous conclurons ce livrable sur les difficultés rencontrées et la gestion de ce projet.

I. Règles de gestion

Afin de mener à bien ce projet, nous avons dû établir quelques règles de gestion en plus de respecter un cahier des charges précis. Nous avons donc établi quels seront les outils que nous utiliserons, les dates clés du projet et les conditions à appliquer pour que la base de données de l'agence de voyage soit fonctionnelle.

1) Le cahier des charges

Dans ce projet étude de cas, une agence de voyage veut numériser son système d'information. Ses processus métiers se compartiment en plusieurs catégories : gestion des clients, gestion du personnel, gestion de la billetterie, gestion des moyens de transports, gestion des points d'étapes et gestion des statistiques. Et il est bien spécifié que « toutes ces catégories de processus comportent chacune son lot de fonctionnalités » :

- Un client doit avoir une adresse de livraison et une adresse de facturation.
- Un client peut avoir un numéro de téléphone principale et d'autres numéros de téléphone.
- Les adresses des clients et des salariés doivent avoir un numéro de voie, un nom de rue, parfois le nom de la résidence, du bâtiment et de l'étage, le code postal et la ville.
- Un employé peut gérer jusqu'à 50 dossiers clients en moyenne.
- Le circuit est composé d'une adresse de départ et une adresse d'arrivée.
- Une étape est déterminée par son moyen de transport, sa ville de départ et sa ville d'arrivée, et aussi ses dates et heures de départ et d'arrivée.
- Les moyens de transports possibles sont le train, l'avion, le car et le bateau.
- La gestion du kilométrage est réalisée à l'aide d'un distancier.
- Le prix d'un voyage se calcule en fonction du cumul du kilométrage de toutes les étapes elles-mêmes assujetties au moyen de transport utilisé.
- Une étape peut être intra-ville. Si elle est intra-ville son prix est soumis à un forfait.
- Un client peut payer en plusieurs fois.
- Il faut pourvoir identifier la date de passage de la commande, la date du ou des paiements, le mode de paiement et le montant du paiement.

La création de procédures pour chaque catégorie est aussi nécessaire pour pouvoir gérer les créations, les modifications, les recherches, les éditions et les suppressions de données sans affecter l'historique de la base de données. De plus, afin de mieux connaître les tendances et attentes des clients, il été également demander de réaliser plusieurs statistiques.

2) Les dates clés et planification

La réalisation du projet s'est déroulée par étape. Tout d'abord, la prise de connaissance du sujet, le choix du chef d'équipe et le brainstorming pour établir les bases. Ensuite, la phase la plus importante, et qui a pris le plus de temps, fut la réflexion et l'établissement du dictionnaire des données ainsi que le modèle conceptuel des données. En effet, il nous a fallu réaliser plusieurs versions et faire preuve de beaucoup de réflexion et de logique pour atteindre un MCD viable. Après avoir établi les fondations de la base de données nous nous sommes réparti les tâches. Cependant, un suivi très régulier et une entraide continue ont rythmé le projet.

Les dates clés sont :

- La prise de connaissance du projet : le 19 décembre 2022
- Point d'avancement : le 04 janvier 2023
- Présentation orale du projet : le 19 janvier 2023

3) Les outils utilisés

Pour mener à bien notre projet nous avons utilisé plusieurs outils/logiciels :

- Google Drive et ses outils : pour que tout le groupe puisse avoir accès directement aux travaux de chacun et que l'actualisation soit dynamique.
- Looping : pour la réalisation du Modèle Conceptuel de Données et du Modèle Logique de Données.
- MySQL et son Workbench : pour la création de la base de données, l'implémentation des données, la réalisation des procédures et du requêtage SQL.
- GenerateData.com & Mockaroo.com : pour générer des données aléatoires pour fournir notre base en données. Mais ces données ont été assez rapidement obsolètes à la suite de modifications tardives du MCD et l'ajout d'attributs dans les tables.
- Chat GPT pour générer des données pour fournir en données la nouvelle base de données.

4) Les contraintes

Pour réaliser le projet, il nous été demandé de respecter la troisième forme normale, c'est-à-dire que nous devions prendre en compte, lors de la création de notre MCD, les règles de vérification 1, 3 et 5. Plus précisément :

- Règle 1 : Toutes les propriétés doivent être élémentaires (elles ne peuvent pas être décomposées).
- Règle 2 : Dans une entité, toutes les propriétés doivent être en dépendance fonctionnelle de l'identifiant/la clé.
- Règle 3 : Une propriété non identifiant ne peut pas être en dépendance fonctionnelle avec une autre propriété non identifiant.

En d'autres termes, les attributs d'une relation doivent contenir une valeur atomique, qui ne se calcule pas (la troisième forme normale permet d'éviter les redondances) et les attributs non-clés d'une relation doivent tous dépendre totalement d'une clé primaire.

Par la suite, pour créer un MCD viable nous avons établir quelques règles et contraintes :

- Une personne est considérée comme Client lorsqu'elle a commandé un voyage
- Un voyage est composé d'une ou plusieurs étapes. Et une étape peut être dans plusieurs voyages.
- Nous devons connaître la date de la commande qui correspond à la date de facturation.
- Un voyage n'est pas le même si le nombre de voyageurs n'est pas le même.
- Nous arrondissons nos prix aux centimes près.
- Le prix forfait d'une étape intra-ville est un prix national de 2€.

II. Le dictionnaire des données

Le dictionnaire de données crée pour ce projet regroupe l'ensemble des données de la base de données tout en spécifiant :

- Le nom code de la donnée (code mnémotechnique)
- La désignation (à quoi correspond la donnée)
- Le type (N = Numérique ; AN = Alphanumérique ; A = Alphabétique ; Booléen)
- La taille de la donnée
- Et l'information pour savoir si la donnée est obligatoire (O) ou non (N).

Le dictionnaire de données est créé au moment de la création de la base de données et doit être mis à jour s'il y a des modifications apportées à la base.

Codes mnémotechniques	Désignations	Type	Taille	Obligatoire
id_client	Numéro du client	N	100	O
nom_client	Nom du client	A	50	O
prenom_client	Prénom du client	A	50	O
date_naissance	Date de naissance du client	A	10	O
genre	Genre du client (H/F/N)	A	1	O
adresse_mail	Adresse mail du client	AN	100	O
actif	Savoir si le client est actif (True) ou non (False)	Booléen	1	O
id_coordonnees	Numéro identifiant de l'adresse	N	100	O
num_voie	Numéro de la voie	N	100	O
nom_rue	Nom de la rue (boulevard, avenue...)	A	150	O
nom_residence	Nom de résidence	A	100	N
nom_batiment	Nom du bâtiment	AN	50	N
etage	Etage	N	10	N
code_postal	Code postal	N	5	O
ville	Ville	A	100	O
tel_mobile	Téléphone principal	N	20	N
autre_tel_1	Autre téléphone n°1	N	20	N
autre_tel_2	Autre téléphone n°2	N	20	N
actif	Savoir si la coordonnée est active (True) ou non (False)	Booléen	1	O
id_personnel	Numéro du salarié	N	100	O
nom_personnel	Nom du salarié	A	50	O
prenom_personnel	Prénom du salarié	A	50	O
date_embauche	Date d'embauche du salarié	N	10	O
mail_pro	Adresse mail du salarié dans l'entreprise	AN	100	O
actif	Savoir si le personnel est actif (True) ou non (False)	Booléen	1	O
id_moyen_transport	Numéro du moyen de transport par transporteur	N	100	O

nom_moyen_transport	Nom du moyen de transport (avion, car, bateau, train)	A	10	O
nom_transporteur	Nom du transporteur	AN	50	O
prix_km	Prix au kilomètre par moyen de transport	N	100	O
actif	Savoir si le moyen de transport est actif (True) ou non (False)	Booléen	1	O
id_etape	Identifiant de l'étape	N	100	O
date_depart	Date de départ de l'étape	N	10	O
date_arrive	Date d'arrivée de l'étape	N	20	O
kilometrage	Nombre de kilomètres parcourus pour l'étape	N	50	O
n_commande	Numéro de la commande	N	100	O
date_commande	Date à laquelle la commande a été passée	N	20	O
id_paiement	Numéro du paiement	N	100	O
date_paiement	Date à laquelle le paiement a été effectué	N	100	O
moyen_paiement	Moyen de paiement utilisé pour le règlement	A	10	O
taux_paiement	Pourcentage payé du prix de la facture	N	100	O
id_facture	Numéro de la facture	N	100	O
tva	Taxe sur la valeur ajoutée	N	100	O
marge	Taux de marge effectuée	N	100	O
acquittee	Situation de la facture : réglée ou non.	Booléen	100	O
id_voyage	Numéro identifiant du voyage	N	100	O
nom_voyage	Nom du voyage	A	100	O
id_voyageur	Id du voyageur	N	100	O
nom_voyageur	Nom du voyageur	A	50	O
prenom_voyageur	Prénom du voyageur	A	50	O
naissance_voyageur	Date de naissance du voyageur	A	20	O
genre_voyageur	Genre du voyageur	A	1	O

III. Les dépendances fonctionnelles et les cardinalités

Les dépendances fonctionnelles ont été choisies sur le principe de la 3eme forme normale. En clé primaire nous n'avions que des identifiants uniques et les attributs ont été choisis en fonction de ce que nous pouvions lire dans le cahier des charges et de ce qui nous était utile pour les statistiques.

Les cardinalités ont été choisies en fonction de ce qui nous semblait logique au premier abord, mais nous nous sommes rendu compte que parfois cela pouvait être problématique, notamment au niveau de la facturation. Nous avions le choix entre faire une (maximum) ou plusieurs (maximum) factures par commande ; en premier lieu nous avions fait un choix et il s'est avéré que c'était plus simple de faire plusieurs paiements pour une facture mais qu'une commande ne pouvait correspondre qu'à une seule facture.

1) Les dépendances fonctionnelles directes de la base de données

Le premier élément de chaque dépendance correspond aux clés primaires de chaque relation. Les clés primaires déterminent chaque attribut ; les attributs sont dépendant de leur clé primaire :

id_client → nom_client, prenom_client, date_naissance, genre, adresse_mail, actif

id_coordonnees → num_voie, nom_rue, nom_residence, nom_batiment, etage, code_postal, ville, tel_mobile, autre_tel_1, autre_tel_2, actif

id_personnel → nom_personnel, prenom_personnel, date_embauche, mail_pro, actif

id_moyen_transport → nom_moyen_transport, nom_transporteur, prix_km, actif

id_etape → date_depart, date_arrivee, kilometrage

id_voyage → nb_voyageurs

id_voyageur → nom_voyageur, prenom_voyageur, naissance_voyageur, genre_voyageur

n_commande → date_commande

id_facture → tva, marge, acquittee

id_paiement → date_paiement, moyen_paiement, montant_paiement

2) Verbalisation des relations

Nous avons donc verbalisé les relations entre chacune des tables en expliquant chacune des cardinalités. Les cardinalités représentent pour chaque couple entité-relation, le nombre minimum et maximum d'occurrences de l'association que peut avoir un objet.

- Une coordonnée peut être habitée par un ou plusieurs (n) personnels
- On peut livrer un ou plusieurs (n) clients à une seule coordonnée
- On peut facturer un ou plusieurs (n) clients à une seule coordonnée

- Une coordonnée peut correspondre au point de départ d'aucune ou plusieurs (n) étapes
- Une coordonnée peut correspondre au point d'arrivée d'aucune ou plusieurs (n) étapes
-
- Un personnel habite à une et une seule coordonnée
- Un personnel s'occupe de 0 à plusieurs (n) commandes

- Un moyen de transport peut être utilisé dans 0 à plusieurs (n) étapes

- Un client se fait facturer à une et une seule coordonnée
- Un client se fait livrer à une et une seule coordonnée
- Un client passe une à plusieurs (n) commandes
- Un client est facturé de 1 à plusieurs (n) factures

- Une étape utilise un seul moyen de transport
- Une étape compose un ou plusieurs (n) voyages
- Une étape possède une unique coordonnée de départ
- Une étape possède une unique coordonnée d'arrivée

- Une commande est composée d'un ou plusieurs (n) voyages
- Une commande n'est prise en charge que par un seul personnel
- Une commande est réglée par une ou plusieurs (n) factures
- Une commande peut être passée par un et un seul client

- Un voyage correspond à une ou plusieurs commandes
- Un voyage est composé d'une ou plusieurs étapes
- Un voyage peut avoir de 0 à plusieurs (n) voyageurs

- Un voyageur participe à un ou plusieurs (n) voyages

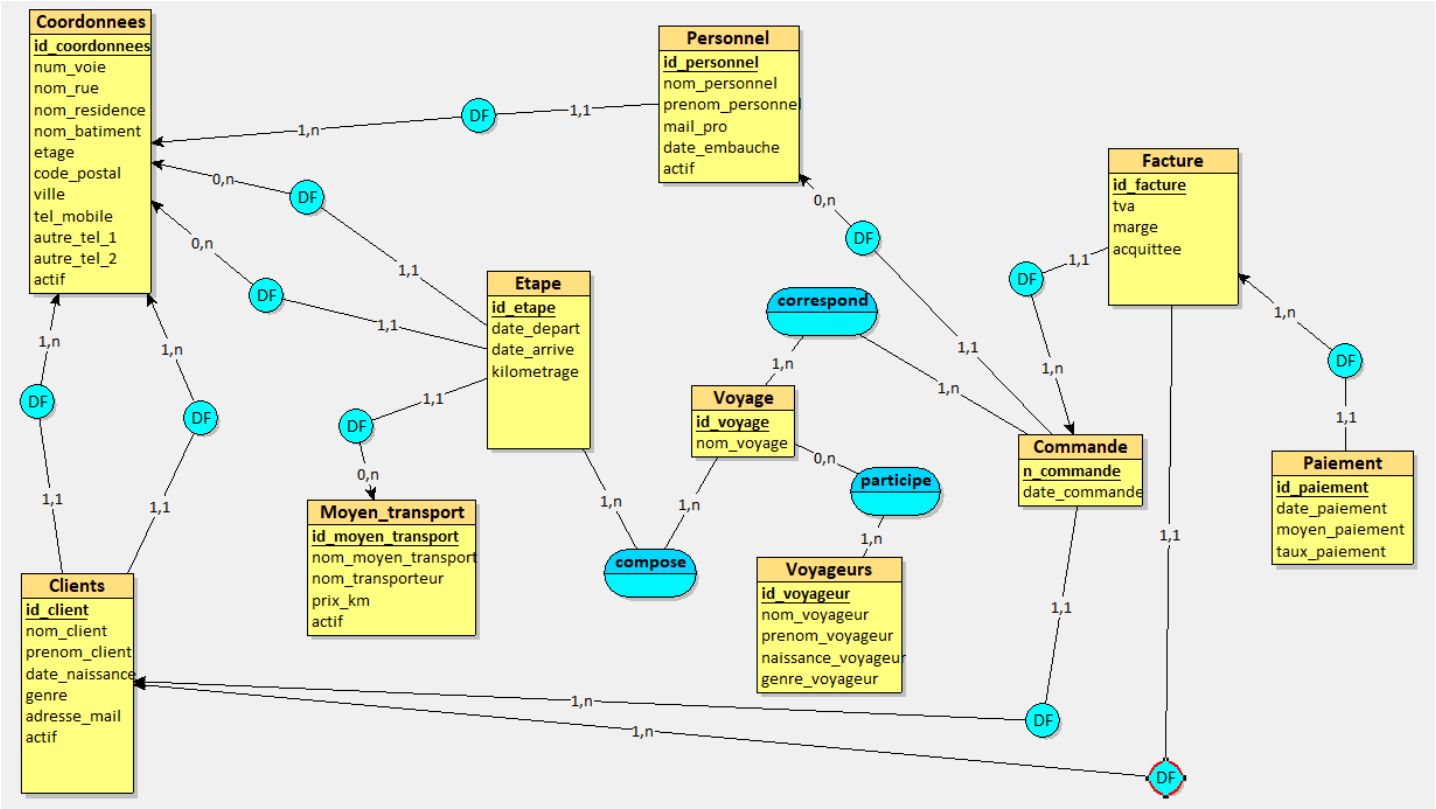
- Une facture ne règle qu'une et une seule commande
- Une facture concerne un ou plusieurs (n) paiements
- Une facture est facturée à un unique client

- Un paiement ne concerne qu'une et une seule facture

IV. Le Modèle Conceptuel de Données (MCD)

Le schéma conceptuel de données, également appelé Modèle Conceptuel de Données, est une représentation des données du système d'information à concevoir. Cette représentation, facilement compréhensible, détermine les relations entre ces données grâce à leurs dépendances fonctionnelles et leurs cardinalités.

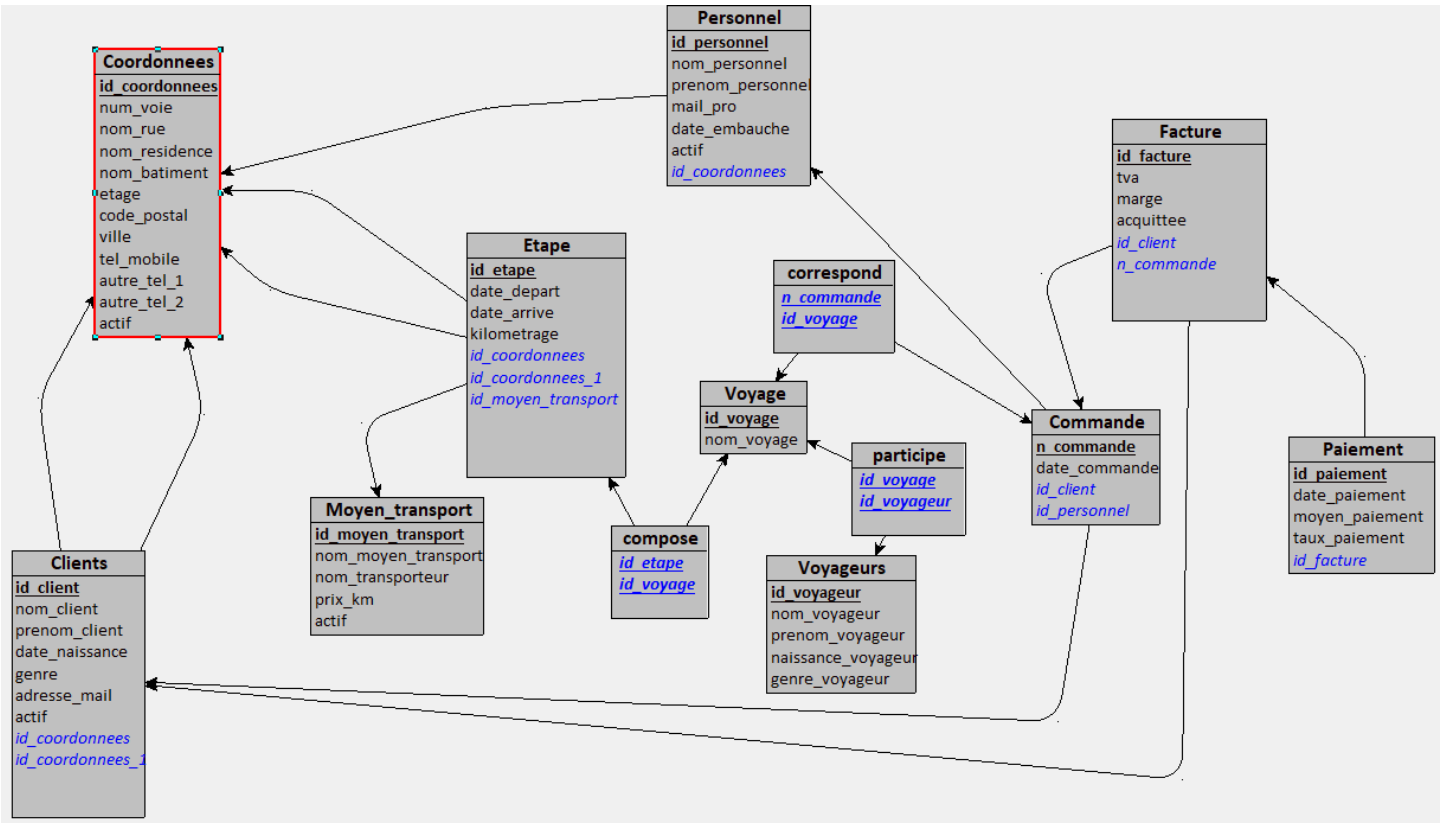
Ainsi, nous retrouvons nos 10 entités (Coordonnées, Personnel, Clients, Moyen_transport, Etape, Voyage, Voyageurs, Commande, Facture et Paiement) et leurs relations et associations.



V. Le Modèle Logique de Données (MLD)

Le Modèle Logique des Données est l'étape intermédiaire entre le modèle conceptuel et le modèle physique de données. Il correspond au MCD auquel on ajoute la définition de l'organisation logique des données. Les entités deviennent des relations, les propriétés des attributs et les identifiants des clés primaires. De plus, les associations disparaissent pour faire place à :

- Soit de nouvelles relations dans le cas d'une association de cardinalités N-N. Les clés primaires des deux relations associées forment la clé primaire de cette nouvelle relation tout en restant des clés étrangères.
- Soit dans le cas de cardinalité 1-N, la relation qui a la cardinalité 1 récupère la clé primaire de la relation à N cardinalité pour en faire une clé étrangère.



VI. Le Modèle Physique de Données

Dans la méthode Merise, le modèle physique des données consiste à implanter une base de données dans un système de gestion de base de données relationnelle. Le langage utilisé pour réaliser cette opération est le SQL.

Ici nous nous sommes basés sur le MLD textuel pour réaliser l'implantation de notre base de données.

1) MLD textuel

On présente ainsi les données issues de la modélisation Merise sous la forme suivante :

- Chaque ligne représente une table (une relation)
- Le nom de la table est écrit en premier
- Les champs (attributs) sont listés entre parenthèses et séparés par des virgules
- Les clés primaires sont soulignées et placé en début de la liste des champs
- Les clés étrangères sont préfixées par un dièse

Ainsi, voici notre MLD textuel :

Coordonnees (id_coordonnees, num_voie, nom_rue, nom_residence, nom_batiment, etage, code_postal, ville, tel_mobile, autre_tel_1, autre_tel_2, actif)

Personnel (id_personnel, nom_personnel, prenom_personnel, mail_pro, date_embauche, actif, #id_coordonnees)

Moyen_transport (id_moyen_transport, nom_moyen_transport, nom_transporteur, prix_km, actif)

Etape (id_etape, date_depart, date_arrivee, kilometrage, #id_depart, #id_arrivee, #id_moyen_transport)

Voyage (id_voyage, nom_voyage)

Voyageurs (id_voyageur, nom_voyageur, prenom_voyageur, naissance_voyageur, genre_voyageur)

Clients (id_client, nom_client, prenom_client, date_naissance, genre, adresse_mail, actif, #id_livraison, #id_facturation)

Commande (n_commande, date_commande, #id_client, #id_personnel)

Facture (id_facture, tva, marge, acquittee, #id_client, #n_commande)

Paiement (id_paiement, date_paiement, moyen_paiement, taux_paiement, #id_facture)

compose (#id_etape, #id_voyage)

correspond (#n_commande, #id_voyage)

participe (#id_voyage, #id_voyageur)

2) Script SQL de création de la base de données

L'ordre de création des tables a son importance et doit être respecter au vu des relations entre les tables.

```
CREATE SCHEMA `bdd_voyage` ;

DROP TABLE IF EXISTS participe;
DROP TABLE IF EXISTS correspond;
DROP TABLE IF EXISTS compose;
DROP TABLE IF EXISTS Paiement;
DROP TABLE IF EXISTS Facture;
DROP TABLE IF EXISTS Commande;
DROP TABLE IF EXISTS Clients;
DROP TABLE IF EXISTS Voyageurs;
DROP TABLE IF EXISTS Voyage;
DROP TABLE IF EXISTS Etape;
DROP TABLE IF EXISTS Moyen_transport;
DROP TABLE IF EXISTS Personnel;
DROP TABLE IF EXISTS Coordonnees;
```

```
CREATE TABLE Coordonnees(
    id_coordonnees INT PRIMARY KEY AUTO_INCREMENT,
    num_voie INT NOT NULL,
    nom_rue VARCHAR(100) NOT NULL,
    nom_residence VARCHAR(100),
    nom_batiment VARCHAR(50),
    etage INT,
    code_postal INT NOT NULL,
    ville VARCHAR(50) NOT NULL,
    tel_mobile VARCHAR(15),
    autre_tel_1 VARCHAR(15),
    autre_tel_2 VARCHAR(15),
    actif BOOLEAN NOT NULL
);
```

```
CREATE TABLE Personnel(
    id_personnel INT PRIMARY KEY AUTO_INCREMENT,
    nom_personnel VARCHAR(50) NOT NULL,
    prenom_personnel VARCHAR(50) NOT NULL,
    mail_pro VARCHAR(100) NOT NULL,
    date_embauche INT NOT NULL,
    id_coordonnees INT NOT NULL,
    actif BOOLEAN NOT NULL,
    FOREIGN KEY(id_coordonnees) REFERENCES Coordonnees(id_coordonnees)
);
```

```
CREATE TABLE Moyen_transport(
    id_moyen_transport INT PRIMARY KEY AUTO_INCREMENT,
    nom_moyen_transport VARCHAR(50) NOT NULL,
    nom_transporteur VARCHAR(50) NOT NULL,
    prix_km FLOAT NOT NULL,
    actif BOOLEAN NOT NULL
);
```

```sql
CREATE TABLE Etape(
    id_etape INT PRIMARY KEY AUTO_INCREMENT,
    date_depart INT NOT NULL,
    date_arrivee INT NOT NULL,
    kilometrage FLOAT NOT NULL,
    id_depart INT NOT NULL,
    id_arrivee INT NOT NULL,
    id_moyen_transport INT NOT NULL,
    FOREIGN KEY(id_depart) REFERENCES Coordonnees(id_coordonnees),
    FOREIGN KEY(id_arrivee) REFERENCES Coordonnees(id_coordonnees),
    FOREIGN KEY(id_moyen_transport) REFERENCES Moyen_transport(id_moyen_transport)
);
CREATE TABLE Voyage(
    id_voyage INT PRIMARY KEY AUTO_INCREMENT,
    nom_voyage VARCHAR(50) NOT NULL
);

CREATE TABLE Voyageurs(
    id_voyageur INT PRIMARY KEY AUTO_INCREMENT,
    nom_voyageur VARCHAR(50) NOT NULL,
    prenom_voyageur VARCHAR(50) NOT NULL,
    naissance_voyageur VARCHAR(50) NOT NULL,
    genre_voyageur VARCHAR(1) NOT NULL
);
CREATE TABLE Clients(
    id_client INT PRIMARY KEY AUTO_INCREMENT,
    nom_client VARCHAR(50) NOT NULL,
    prenom_client VARCHAR(50) NOT NULL,
    date_naissance VARCHAR(10) NOT NULL,
    genre CHAR(1) NOT NULL,
    adresse_mail VARCHAR(100) NOT NULL,
    id_livraison INT NOT NULL,
    id_facturation INT NOT NULL,
    actif BOOLEAN NOT NULL,
    FOREIGN KEY(id_livraison) REFERENCES Coordonnees(id_coordonnees),
    FOREIGN KEY(id_facturation) REFERENCES Coordonnees(id_coordonnees)
);
```

```sql
CREATE TABLE Commande(
   n_commande INT PRIMARY KEY AUTO_INCREMENT,
   date_commande INT NOT NULL,
   id_client INT NOT NULL,
   id_personnel INT NOT NULL,
   FOREIGN KEY(id_client) REFERENCES Clients(id_client),
   FOREIGN KEY(id_personnel) REFERENCES Personnel(id_personnel)
);

CREATE TABLE Facture(
   id_facture INT PRIMARY KEY AUTO_INCREMENT,
   tva FLOAT NOT NULL,
   marge FLOAT NOT NULL,
   acquittee BOOLEAN NOT NULL,
   id_client INT NOT NULL,
   n_commande INT NOT NULL,
   FOREIGN KEY(id_client) REFERENCES Clients(id_client),
   FOREIGN KEY(n_commande) REFERENCES Commande(n_commande)
);
CREATE TABLE Paiement(
   id_paiement INT PRIMARY KEY AUTO_INCREMENT,
   date_paiement INT NOT NULL,
   moyen_paiement VARCHAR(10) NOT NULL,
   taux_paiement FLOAT NOT NULL,
   id_facture INT NOT NULL,
   FOREIGN KEY(id_facture) REFERENCES Facture(id_facture)
);

CREATE TABLE compose(
   id_etape INT,
   id_voyage INT,
   PRIMARY KEY(id_etape, id_voyage),
   FOREIGN KEY(id_etape) REFERENCES Etape(id_etape),
   FOREIGN KEY(id_voyage) REFERENCES Voyage(id_voyage)
);
CREATE TABLE correspond(
   n_commande INT,
   id_voyage INT,
   PRIMARY KEY(n_commande, id_voyage),
   FOREIGN KEY(n_commande) REFERENCES Commande(n_commande),
   FOREIGN KEY(id_voyage) REFERENCES Voyage(id_voyage)
);
CREATE TABLE participe(
   id_voyage INT,
   id_voyageur INT,
   PRIMARY KEY(id_voyage, id_voyageur),
   FOREIGN KEY(id_voyage) REFERENCES Voyage(id_voyage),
   FOREIGN KEY(id_voyageur) REFERENCES Voyageurs(id_voyageur)
);
```

VII. Import des données

Pour vérifier que notre base de données soit opérationnelle et pour réaliser et vérifier les requêtes et les procédures crées nous avons généré un jeu de données avec ChatGPT principalement et notre imagination.

Nous avons décidé de partir sur un import de donné de masse. Tout d'abord nous nous sommes assuré qu'il n'y ait aucune donnée dans les tables crées et que les id qui s'auto-incrémentent, repartent de 1.

```
DELETE FROM participe;
DELETE FROM correspond;
DELETE FROM compose;
DELETE FROM Paiement;
DELETE FROM Facture;
DELETE FROM Commande;
DELETE FROM Clients;
DELETE FROM Voyage;
DELETE FROM Voyageurs;
DELETE FROM Etape;
DELETE FROM Moyen_transport;
DELETE FROM Personnel;
DELETE FROM Coordonnees;
```

```
ALTER TABLE Coordonnees AUTO_INCREMENT=1;
ALTER TABLE Personnel AUTO_INCREMENT=1;
ALTER TABLE Moyen_transport AUTO_INCREMENT=1;
ALTER TABLE Etape AUTO_INCREMENT=1;
ALTER TABLE Voyageurs AUTO_INCREMENT=1;
ALTER TABLE Voyage AUTO_INCREMENT=1;
ALTER TABLE Clients AUTO_INCREMENT=1;
ALTER TABLE Commande AUTO_INCREMENT=1;
ALTER TABLE Facture AUTO_INCREMENT=1;
ALTER TABLE Paiement AUTO_INCREMENT=1;
```

Exemples d'import de données :

```
INSERT INTO Coordonnees (num_voie, nom_rue, nom_residence, nom_batiment, etage, code_postal, ville, tel_mobile, autre_tel_1, autre_tel_2,actif)
VALUES
(123, 'Rue du Pont', NULL, NULL, NULL, 75000, 'Paris', '0123456789', NULL, NULL,1),
(456, 'Rue de la Montagne', 'Résidence du Soleil', 'Batiment A', 6, 69000, 'Lyon', '0987654321', '0612345678', NULL,1),
(789, 'Avenue de la Forêt', NULL, 'Immeuble Le Chêne', NULL, 13000, 'Marseille', '0123456789', '0612345678', '0712345678',1),
(321, 'Rue du Jardin', 'Résidence des Fleurs', NULL, 2, 13100, 'Aix-en-Provence', '0987654321', NULL, NULL,1),
(654, 'Boulevard de la Mer', NULL, 'Immeuble Le Phare', 7, 13200, 'Arles', '0123456789', '0612345678', '0712345678',1),
(333, 'Rue des Tilleuls', NULL, 'Immeuble Les Tilleuls', 4, 34000, 'Montpellier', '0987654321', '0612345678', '0712345678',1);
```

```
INSERT INTO Personnel (nom_personnel, prenom_personnel, mail_pro, date_embauche, actif, id_coordonnees)
VALUES
('Elrick', 'Kacy', 'kelrick0@voyage.fr', '1503697375',1,1),
('Moncaster', 'Ferd', 'fmoncaster1@voyage.fr', '1653697375',1,2),
('Blondell', 'Wynn', 'wblondell2@voyage.fr', '1603697375',1,3),
('Trasler', 'Britney', 'btrasler3@voyage.fr', '1610990164',1,4),
('Middis', 'Rebekkah', 'rmiddis4@voyage.fr', '1522539820',1,5);
```

Voir import de données complet dans le fichier « donnees.sql ».

VIII. Les procédures stockées

Une procédure stockée est un ensemble d'instructions que l'on peut exécuter sur commande. Une procédure est un objet de la base de données stocké de manière durable, au même titre qu'une table. Elle n'est pas supprimée à la fin de la session comme l'est une requête préparée. Les procédures stockées peuvent permettre de gagner en performance en diminuant les allers-retours entre le client et le serveur. Elles peuvent également aider à sécuriser une base de données et à s'assurer que les traitements sensibles soient toujours exécutés de la même manière.

Ainsi, nous avons donc dû réaliser les procédures de Création, Modification, Recherche, Edition et Suppression de données.

- Exemple script de procédure pour la création d'une étape :

```
# Potentiel pré-requis : ajout 1 ou 2 coordonees et un moyen de transport s'ils n'existent pas déjà
DROP PROCEDURE IF EXISTS AjoutEtape;
DELIMITER //
CREATE PROCEDURE AjoutEtape(     #Ajout d'une étape
IN date_depart INT,
IN date_arrivee INT,
IN kilometrage FLOAT,
IN id_depart INT,
IN id_arrivee INT,
IN id_moyen_transport INT)
BEGIN
INSERT INTO Etape (date_depart, date_arrivee, kilometrage, id_depart, id_arrivee, id_moyen_transport)
VALUES
(date_depart, date_arrivee, kilometrage, id_depart, id_arrivee, id_moyen_transport);
END //
DELIMITER ;
```

- Exemple de procédure de modification de toutes les données correspondant à un id particulier :

```
#modifier toutes les données concernant 1 étape
DROP PROCEDURE IF EXISTS ModifEtape;
DELIMITER //
CREATE PROCEDURE ModifEtape
(IN id INT,
IN date_depart INT,
IN date_arrivee INT,
IN kilometrage FLOAT,
IN id_depart INT,
IN id_arrivee INT,
IN id_moyen_transport INT)
BEGIN
UPDATE Etape
SET date_depart=date_depart,
date_arrivee=date_arrivee,
kilometrage=kilometrage,
id_depart=id_depart,
id_arrivee=id_arrivee,
id_moyen_transport=id_moyen_transport
WHERE id_etape = id;
END //
DELIMITER ;
```

- Exemple de modification d'une seule donnée correspondant à un id particulier :

```
# modifier 1 donnée
DROP PROCEDURE IF EXISTS ModifDonneeEtape;
DELIMITER //
CREATE PROCEDURE ModifDonneeEtape
(IN id INT,
IN kilometrage FLOAT)
BEGIN
UPDATE Etape
SET
kilometrage=kilometrage
WHERE id_etape = id;
END //
DELIMITER ;
```

- Exemple de procédure de recherche en fonction d'un id :

```
#recherche d'une etape via son id
DROP PROCEDURE IF EXISTS RechercheEtape;
DELIMITER //
CREATE PROCEDURE RechercheEtape
(IN id INT)
BEGIN
SELECT id_etape, date_depart, date_arrivee, kilometrage, id_depart, id_arrivee, id_moyen_transport
FROM Etape
WHERE id_etape=id;
END //
DELIMITER ;
```

- Exemple de procédure d'édition en fonction d'un id :

```
# edition d'une etape via son id
DROP PROCEDURE IF EXISTS EditionEtape;
DELIMITER //
CREATE PROCEDURE EditionEtape
(IN id INT,
IN kilometrage FLOAT)
BEGIN
SELECT id_etape, date_depart, date_arrivee, kilometrage, id_depart, id_arrivee, id_moyen_transport
FROM Etape WHERE id_etape=id;
CALL ModifDonneeEtape (11, 145);
SELECT id_etape, date_depart, date_arrivee, kilometrage, id_depart, id_arrivee, id_moyen_transport
FROM Etape WHERE id_etape=id;
END //
DELIMITER ;
```

- Exemple de procédure de suppression d'une étape en fonction d'un id :

```
# procédure de suppression d'une étape en fonction de son id
DROP PROCEDURE IF EXISTS SuppEtape;
DELIMITER //
CREATE PROCEDURE SuppEtape
(IN id INT)
BEGIN
DELETE FROM Etape
WHERE id_etape=id;
END//
DELIMITER ;
```

Cependant, comme il y a des relations entre les tables, certaines données ne peuvent pas être supprimées sans qu'elles soient supprimées au préalable dans les autres tables qui l'utilisent mais qui interviennent après (références à la création des tables). Ainsi, pour palier à ce problème nous avons créés des suppressions en cascade. A noter tout de même qu'au niveau fonctionnel, la suppression de données n'est pas utilisée.

```
# procédure de suppression en cascade d'une étape en fonction de son id
DROP PROCEDURE IF EXISTS SuppEtapeCascade;
DELIMITER //
CREATE PROCEDURE SuppEtapeCascade
(IN id_stop INT)
BEGIN
WHILE (Select count(*) FROM compose WHERE id_etape=id_stop) > 0 do
    CALL SuppComposeEtape((SELECT (id_etape) FROM compose  WHERE id_etape=id_stop LIMIT 1));
END WHILE;
DELETE FROM Etape
WHERE id_etape=id_stop;
END//
DELIMITER ;
```

Dans la procédure SuppEtapeCascade nous appelons la procédure SuppComposeEtape ci-dessous :

```
DROP PROCEDURE IF EXISTS SuppComposeEtape;
DELIMITER //
CREATE PROCEDURE SuppComposeEtape
(IN id_stop INT)
BEGIN
WHILE (Select count(*) FROM compose WHERE id_etape=id_stop) > 0 do
    DELETE FROM compose WHERE id_etape=id_stop LIMIT 1;
END WHILE;
END //
DELIMITER ;
```

Les procédures au complet sont à retrouver dans le fichier sql « Procédures ».

IX. Les requêtes

Comme évoqué précédemment, dans le cadre du projet de développement d'une base de données pour une agence de voyage, il nous était demander de réaliser des statistiques pour connaître les tendances clients, par exemple, mais aussi pour avoir une vision de l'activité globale et précise de cette agence de voyage. Pour ce faire, nous avons créer des requêtes plus ou moins compliquées. Certaines demandaient seulement des informations récupérables dans une seule table mais la plupart des requêtes nécessitaient des jointures entre les différentes tables pour obtenir les informations demandées.

Le type de jointure que nous avons utilisé afin de réaliser nos requêtes est l'INNER JOIN, qui est une opération de jointure interne. En effet, ce type de jointures internes combinent les enregistrements de deux tables chaque fois qu'il existe des valeurs correspondantes dans un champ commun aux deux tables. Toutefois, malgré que nous nous soyons basés sur les jointures internes pour réaliser nos requêtes, il ne faut pas oublier qu'il existe plusieurs autres jointures possibles. De plus, nous avons aussi utilisé des VIEW, sorte de tables virtuelles, pour facilité les jointures et réaliser des requêtes plus légères.

Exemples de requêtes (liste exhaustive des requêtes à retrouver dans le fichier sql « requetes »)

- Requête 4 :

```
#Requête 4 : Connaître les villes les plus prisées
SELECT Coordonnees.ville,count(Etape.id_etape) AS 'Villes les plus prisées' #On selectionne la ville et on les comptes
FROM Etape #Depuis la table etape
INNER JOIN Coordonnees ON Coordonnees.id_coordonnees=Etape.id_arrivee #Qu'on joint a coordonnée pour avoir la ville d'arrivée
GROUP BY Coordonnees.ville #On regroupe par ville pour avoir le compte par ville
ORDER BY count(Etape.id_etape) DESC; #On trie par ls villes les plus presentes
```

- Requête 7 :

```
#Requête 7 : Connaître le nombre de voyage dont les moyens de transports sont l'avion et le car.
DROP VIEW req7; #Cette fois je suis passé par une view pour alleger la requete

CREATE VIEW req7 AS SELECT Moyen_transport.nom_moyen_transport,compose.id_voyage,compose.id_etape
FROM Etape
INNER JOIN Moyen_transport ON Etape.id_moyen_transport=Moyen_transport.id_moyen_transport
INNER JOIN Compose ON compose.id_etape=Etape.id_etape; #Meme chose que pour la requete numero 6

SELECT count(DISTINCT id_voyage) AS "Nb voyages par avion et car"
FROM req7
WHERE id_voyage IN
(SELECT id_voyage FROM req7 AS T1 WHERE EXISTS  #On est obligé d'appeler deux fois req7 pour avoir deux cases moyen de transport pour les differentes etapes
(SELECT * FROM req7 AS T2 WHERE (T1.nom_moyen_transport = "Avion") #On ne selectionne uniquement ceux qui ont pour moyen de transport l'avion
AND (T2.nom_moyen_transport = "Bus") #Et le bus
AND (T1.id_voyage=T2.id_voyage))); #Et qui ont le meme id_voyage
```

- Requête 12 :

```
#Requête 12 : Connaître le taux de séniors qui ont voyagé les 6 derniers mois
SELECT COUNT(DISTINCT DATEDIFF(NOW(),str_to_date(naissance_voyageur, "%Y-%m-%d"))/365) #on selectionne les voyageurs de plus de 65 ans
/(SELECT DISTINCT COUNT(DATEDIFF(NOW(),str_to_date(naissance_voyageur, "%Y-%m-%d"))/365) # on divise par le nombre total de voyageurs
FROM Voyageurs #De la table voyageurs
INNER JOIN participe ON participe.id_voyageur=voyageurs.id_voyageur
INNER JOIN voyage ON participe.id_voyage=voyage.id_voyage
INNER JOIN compose ON voyage.id_voyage=compose.id_voyage
INNER JOIN etape ON compose.id_etape=etape.id_etape #Qu'on lie jusqu'à etape
WHERE date_depart>unix_timestamp(NOW())-15778800) AS taux #uniquement les voyages datant de moins de 6 mois
FROM Voyageurs #De la table voyageus
INNER JOIN participe ON participe.id_voyageur=voyageurs.id_voyageur
INNER JOIN voyage ON participe.id_voyage=voyage.id_voyage
INNER JOIN compose ON voyage.id_voyage=compose.id_voyage
INNER JOIN etape ON compose.id_etape=etape.id_etape WHERE date_depart>unix_timestamp(NOW())-15778800 #Pareil que dans la requete imbriqué
AND DATEDIFF(NOW(),str_to_date(naissance_voyageur, "%Y-%m-%d"))/365>65; #Sauf qu'on selectionne uniquement ceux dont l'age est supperieur à 65 ans
```

Requête 3 :

```
#Requête 3 : Connaître le coût moyen des voyages
DROP VIEW IF EXISTS Req3;

CREATE VIEW Req3 AS SELECT etape.id_etape, etape.kilometrage, moyen_transport.prix_km, Facture.marge, facture.tva, moyen_transport.nom_moyen_transport,
moyen_transport.id_moyen_transport FROM facture
INNER JOIN Commande ON Facture.n_commande=Commande.n_commande
INNER JOIN correspond ON commande.n_commande=correspond.n_commande
INNER JOIN Voyage ON Voyage.id_voyage=Correspond.id_voyage
INNER JOIN Compose ON compose.id_voyage=Voyage.id_voyage
INNER JOIN Etape ON etape.id_etape=Compose.id_etape
INNER JOIN moyen_transport ON Etape.id_moyen_transport=moyen_transport.id_moyen_transport; #View qui lie les tables des moyens de transport jusqu'à la facture

SELECT round(avg(prix),2) AS Prix_Moyen_Voyages #Selectionne la moyenne des prix arrondi au centime (que l'on nomme Prix_Moyen_Voyages)
FROM (SELECT
(CASE WHEN req3.id_etape=T1.id_etape AND T1.ville=T2.ville THEN 2 #Qui renvoit 2 quand c'est en intra ville, la ville de depart (T1) est la meme que celle d'arrivée (T2)
ELSE (round(avg(req3.kilometrage*req3.prix_km*(1+req3.marge)*(1+req3.tva)),2))END) #Quand c'est en extra-ville, il renvoit le kilometrage*le prix au km*la marge*la tva
AS prix #Que l'on nomme prix
FROM Req3 #Depuis la view Req3
INNER JOIN (SELECT coordonnees.ville, etape.id_etape, coordonnees.id_coordonnees, etape.date_depart, etape.id_depart, etape.id_arrivee, etape.kilometrage
FROM Coordonnees INNER JOIN etape ON coordonnees.id_coordonnees=etape.id_depart)
AS T1 ON T1.id_etape = Req3.id_etape
INNER JOIN (SELECT coordonnees.ville, etape.id_etape, coordonnees.id_coordonnees, etape.date_depart, etape.id_depart, etape.id_arrivee
FROM Coordonnees INNER JOIN etape ON coordonnees.id_coordonnees=etape.id_arrivee)
AS T2 ON T2.id_etape = Req3.id_etape # On lie deux fois etape pour avoir la ville de depart et d'arrivée
GROUP BY req3.id_etape ORDER BY req3.id_etape) AS prix_total; #On le regroupe par etape pour avoir la moyenne des prix par etapes et on trie dans l'ordre des etapes
```

X. Les difficultés rencontrées

1) Illona

J'ai rencontré des difficultés avec le projet car il y a beaucoup de nouvelles notions à intégrer. Nous avons eu des difficultés avec la gestion du projet. J'ai trouvé ce projet assez stressant car pour ma part, il est plus lourd est difficile que la programmation en C. Mais je commence à intégrer les bases même si j'ai encore certaines difficultés avec les requêtes.

2) Eloïc

Certaines requêtes étaient plutôt dures notamment au niveau des jointure et des étapes intra-ville, nous avions un manque d'explications là-dessus et la doc sur internet n'était pas simple à trouver, sans intervenant régulier on se retrouvait vite bloqué. Comme c'était la première fois qu'on créait une base de données depuis 0, on a eu beaucoup de mal à estimer notre temps de travail. On a aussi rencontré des difficultés sur les types (notamment sur les dates/timestamp), savoir lesquelles mettre dans le MLD et s'il fallait les changer dans le dictionnaire et/ou dans le Script SQL. Le cahier des charges n'était parfois pas clair (même si je pense que dans la vraie vie, les besoins donnés d'un client ne sont pas toujours clairs), vis-à-vis des étapes et points d'étape par exemple on a du décider d'une compréhension de la phrase en classe entière. Au global le projet vu de l'extérieur semble plutôt simple et adaptée à une première approche des bases de données. Une fois le projet fini, je trouve que c'est un peu lourd, peut-être qu'un tout petit projet juste avant pour se faire la main sur le langage ne serait pas de trop.

Avis sur répartition des taches : Sachant qu'on avait beaucoup de mal à estimer la durée d'une simple tache, c'était compliquer de se les repartir et ça a pas forcement été bien fait, sans forcément que ça ne soit la faute que quelqu'un en particulier. Par exemple j'ai fait pratiquement toutes les requêtes et à peine quelques procédures.

3) Maewenn

Ce premier projet en groupe n'était pas des plus faciles. J'ai accepté d'être chef d'équipe par défaut car personne d'autre ne voulait cette place. Je pense que pour une meilleure gestion de projet et un travail d'équipe efficace il aurait fallu que nous ayons un minimum d'expérience sur la création d'une base de données, c'est-à-dire, que les simples cours et le peu d'exercices effectués en amont n'étaient pas suffisant pour nous préparer à un tel projet. De plus, chacun ayant des lacunes dans certains domaines, répartir les tâches m'a semblé assez compliqué pour éviter les retards et les blocages, exemple : Eloïc a réalisé presque toutes les requêtes (même si la réflexion était collective) et j'ai réalisé presque toutes les procédures. Je n'ai donc pas réussi à bien répartir les tâches. Le fait d'être perfectionniste ne m'a pas aidé non plus car cela nous a poussé à une réflexion au-delà de ce qui été attendu pour ce projet et m'a surchargé de travail.

XI. Conclusion

Suite à une longue réflexion et plusieurs versions de notre MCD, nous sommes assez contents du résultat obtenu. Selon nous, la base de données que nous avons créé peut être viable et utilisable par une agence de voyage réelle. Nous avons rencontré plusieurs difficultés lors de ce projet qui pourtant nous semblait de prime abord assez simple. Mais il nous aura tout de même apporté des connaissances et plus d'aisance en conception de base de données et requêtage SQL. Ainsi, malgré les difficultés rencontrées, nous avons su tirer profit de l'intelligence collective pour s'entendre sur un sujet qui n'était pas forcément très clair et pour nous orienter dans nos choix et réalisations.

Nous restons tout de même réalistes et savons que pour une meilleure utilisation de notre système d'information plusieurs amélioration sont possibles notamment …

L'exhaustivité des requêtes (commentées), des procédures, de l'implantation des données générées, la création de la base de données, les schémas du MCD et du MLD se trouvent sous ce lien GIT :