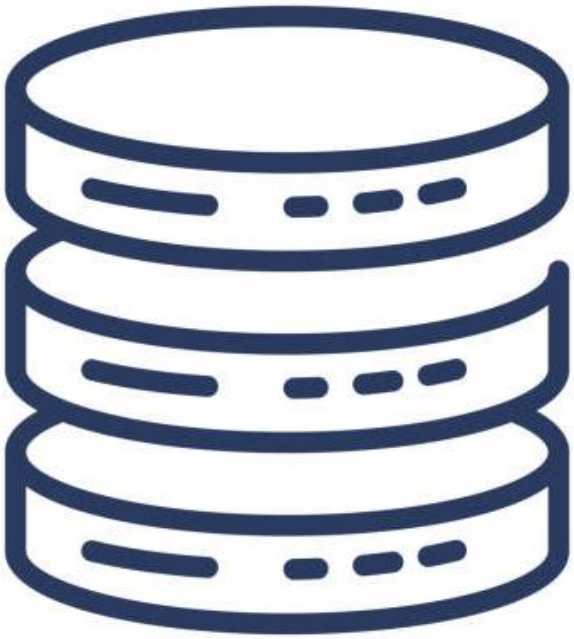
# THE DATABASE PROJECT
# FINAL DELIVERABLE



Benoit BLEE
Luc BARRETO
Thomas FOULON

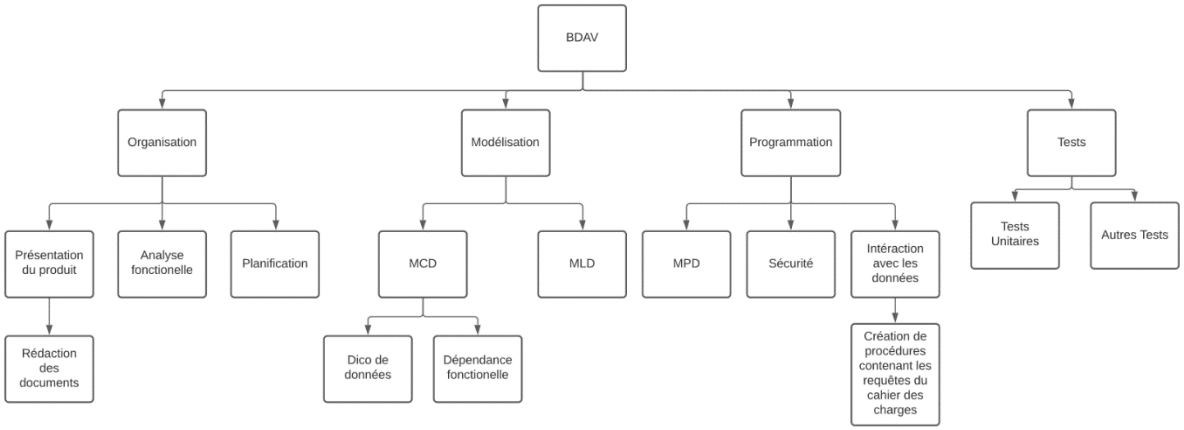
# Table of contents



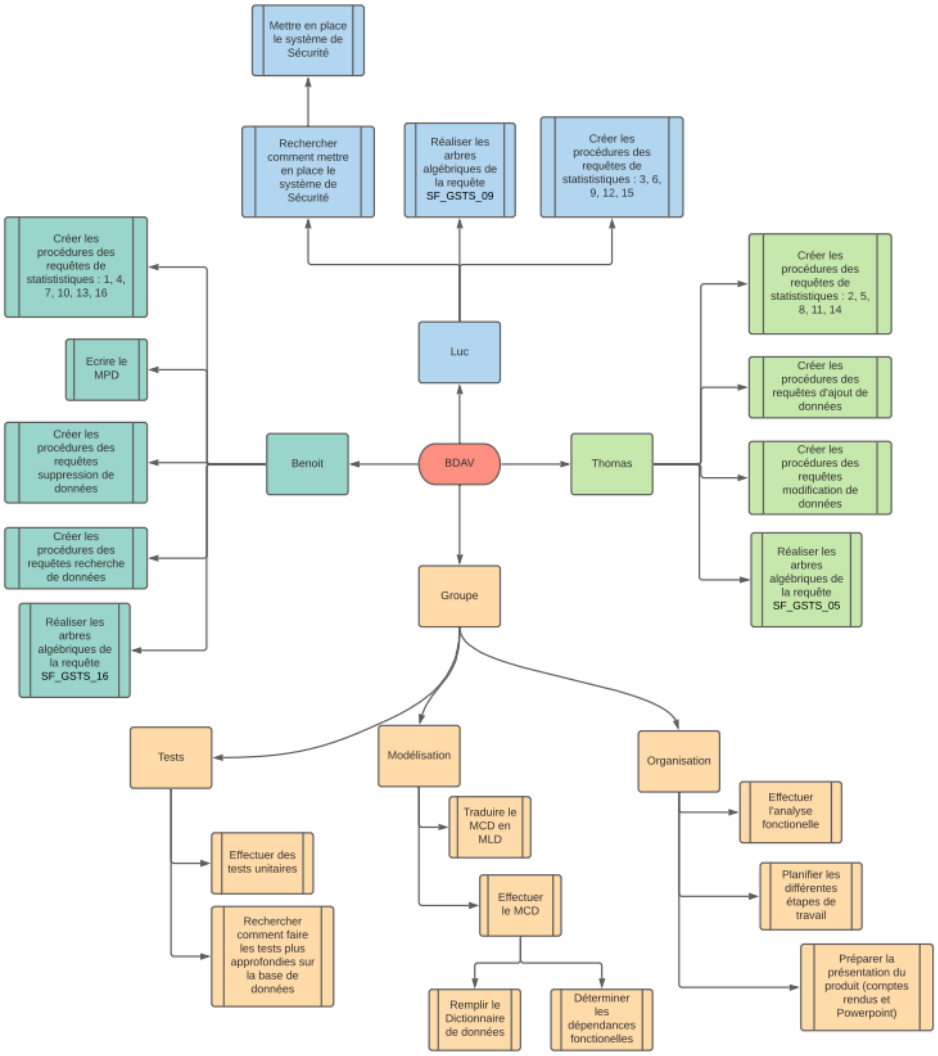
# INTRODUCTION

Since the last deliverable, we didn't change our organization. Still, here is a reminder of our WBS and OBS.

**WBS :**




**OBS :**



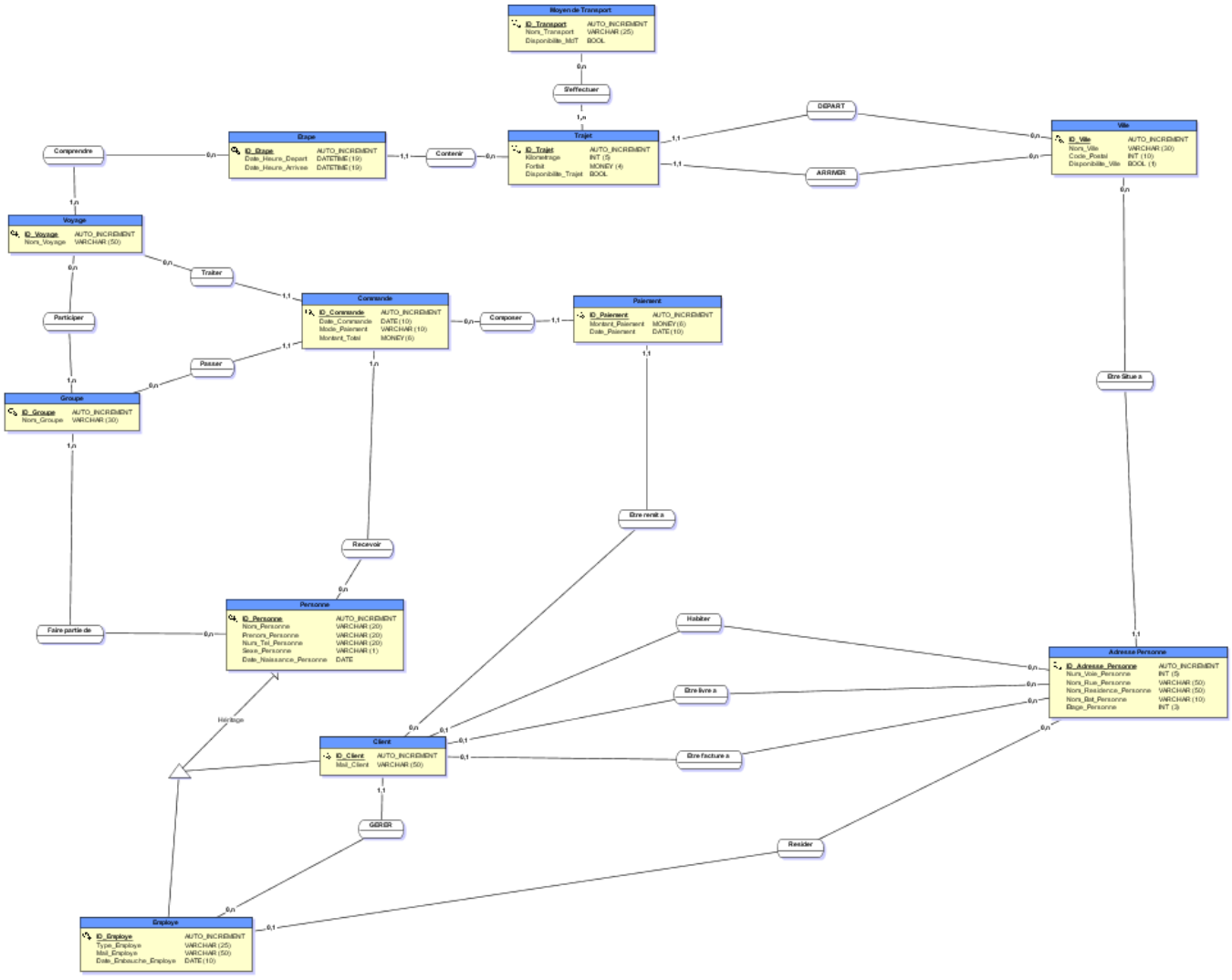
# DATA DICTIONNARY

However, during the creation of our database and its populations, it appeared that we needed more attributes for it to be effective and more efficient. Here is a sample of the updated version of our data dictionary , please find the full one in attachment :

| Code mnémonique | Désignation | Type de Données | Taille |
|---|---|---|---|
| ID_Personne | ID_PERSONNE | Auto_increment | 8 |
| Nom_Personne | NOM_PERSONNE | Varchar | 20 |
| Prenom_Personne | PRENOM_PERSONNE | Varchar | 20 |
| Num_Tel_Personne | NUM_TEL_PERSONNE | Varchar | 20 |
| Mail_Personne | MAIL_PERSONNE | Varchar | 50 |
| Sexe_Personne | SEXE_PERSONNE | Varchar | 1 |
| Date_Naissance_Personne | DATE_NAISSANCE_PERSONNE | Date | |
| Type_Employe | TYPE_PERSONNEL | Varchar | 9 |
| Date_Embauche_Employe | DATE_EMBAUCHE_EMPLOYE | Date | 10 |
| ID_Adresse_Personne | ID_ADRESSE_PERSONNE | Auto_increment | 8 |
| Type_Adresse_Personne | TYPE_ADRESSE_PERSONNE | Varchar | 11 |
| Num_Voie_Personne | NUM_VOIE_PERSONNE | Int | 5 |
| Nom_Rue_Personne | NOM_RUE_PERSONNE | Varchar | 50 |
| Nom_Residence_Personne | NOM_RESIDENCE_PERSONNE | Varchar | 50 |
| Nom_Bat_Personne | NOM_BAT_PERSONNE | Varchar | 50 |
| Etage_Personne | ETAGE_PERSONNE | Int | 3 |
| Code_Postal | CODE_POSTAL | Int | 10 |
| Ville_Personne | VILLE_PERSONNE | Varchar | 50 |
| ID_Groupe | ID_GROUPE | Auto_increment | 8 |
| Nom_Groupe | NOM_GROUPE | Varchar | 30 |
| ID_Voyage | ID_VOYAGE | Auto_increment | 8 |
| Nom_Voyage | NOM_VOYAGE | Varchar | 50 |

# CONCEPTUAL DATA MODEL

We also updated our CDM. You can open the JMerise find sent in attachment for a better view.

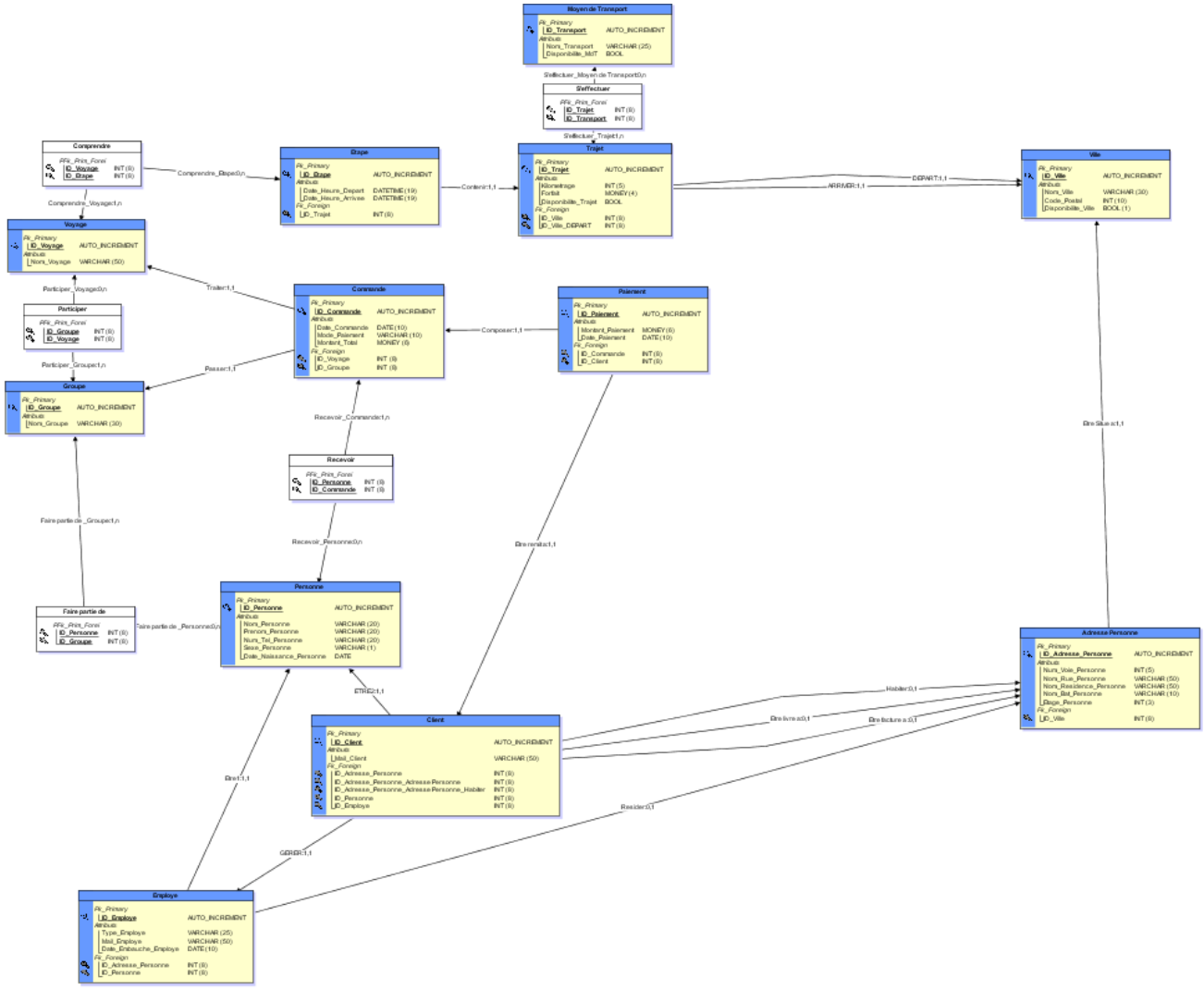


The major changes are the following ones :

- First we changed the gestion of the clients and employees. All of them are stocked into a “Personne” Table and linked to either a client table, either an employee table or even both (since an employee can buy a trip as well). Specifics informations on the personnes depending on their status will be stored into these table (i.e : for an employee, the hire date).
- To avoid repetition, we merged the two “addresses” tables into a unique one. The associations help us to determine if this adresse is one where to deliver a client, where he lives etc…
- To satisfy the need of certain employee to manage clients, we also added an association called “gérer” .
- Finally we deleted departure-town and arrival-town values from the “traject” table and linked it to the a new table called town, also linked to the address table. As a consequence we can avoid repetition of a town which can be a destination and someone else’s address at the same time. We added the “disponibilite” attribute to store the information of whether or not the town is in our trip list. We used the same system in the following tables : “Trajet” and “moyen de transport”. This enable us to delete an information from our catalogue without affecting the ancient orders/trips.

# LOGIC DATA MODEL

As we did in the last deliverable, we convert the CDM into a LDM. You can also open this one in JMerise for a better view of the scheme.



# PHYSIC DATA MODEL

Again we updated or physic data model

```
# Table: Commande
CREATE TABLE Commande(
      ID_Commande   Int  Auto_increment  NOT NULL ,
      Date_Commande Date NOT NULL ,
      Mode_Paiement Varchar (10) NOT NULL ,
      Montant_Total DECIMAL (15,3)  NOT NULL ,
      ID_Voyage     Int NOT NULL ,
      ID_Groupe     Int NOT NULL
           ,CONSTRAINT Commande_PK PRIMARY KEY (ID_Commande)

           ,CONSTRAINT Commande_Voyage_FK FOREIGN KEY (ID_Voyage) REFERENCES Voyage(ID_Voyage)
           ,CONSTRAINT Commande_Groupe0_FK FOREIGN KEY (ID_Groupe) REFERENCES Groupe(ID_Groupe)
)ENGINE=InnoDB;

# Table: Moyen de Transport
CREATE TABLE Moyen_de_Transport(
      ID_Transport      Int  Auto_increment  NOT NULL ,
      Nom_Transport     Varchar (25) NOT NULL ,
      Disponibilite_MdT Bool NOT NULL
           ,CONSTRAINT Moyen_de_Transport_PK PRIMARY KEY (ID_Transport)
)ENGINE=InnoDB;
```

Once the tables were created, we had to populate the whole database and make sure every row of data in a table were linked to the correct row from another table thanks to the IDs. We created a bit more than 100 profiles, gathered or not in families and groups and covering a wide range of age. Those had access to more than 150 different trips and could select as many of them as they wanted to compose their journey. They could also choose different mean of transportation to travel from a point A to a point B.

Then, we developed the 16 statistical queries asked in the specifications and all the one linked to the use and update of the database : adding data, deleting data, modifying data and looking for data.

Some of them had to be divided into separate queries. For example :

**SF_GC_03** is a query that needs to modify informations of a client. However, since a client can have multiple addresses, we decided to split this query into two different one; one where you will change informations about a client, one where you will change informations about one of his addresses.

Another example, **SF_GP_02** is meant to delete an employee from the database. But since some are linked to clients because they manage them, we cannot do so. As a consequence we need to start by linking these clients to another employee, before deleting the ancient one. That is the purpose of **SF_GP_02_remplacement.**

Concerning deleting means of transportation as we mentioned earlier we cannot just delete them from the table. This would affect past trips made with those ones. The same goes with the “Trajet” table. To overcome this problem, we created a “disponibilite” attribute, as mentioned earlier in this document. **The SF_GMT_02** query, firstly meant to delete means of transportation will in fact change the “disponibilite” column from 1 to 0. We also renamed it **SF_GMT_02_mdt** and created another version to do so with the “Trajet” table : **SF_GMT_02_trajet.**

Finally, adding a trip to the database appeared to be harder than we thought. We decided to divide the **SF_GB_ 01** into three different ones. First **SF_GB_01_Etape** was developed to enable the addition of different steps to our database. A step can be used in different trips. Then **SF_GB_01_VoyageEtGroupe** deals with the creation of a trip (mostly giving it a name) and assign it to a group. Finally **SF_GB_01_AttributionEtape** allows to link already existing steps to a trip.

You’ll find in attachment all these stocked procedures in .sql files. You’ll find another one called “call.sql” with different example of how to use them.

**About the statistical queries :**

- **SF_GSTS_11** was meant to find the percentage of men and women travelling last six months. Sadly, the population we created for our database doesn't contain people who travelled during the past year (this wasn't made on purpose we promise…). We decided to generalized this query by since how many months you want to see the percentage of men and women. You'll precise it when calling the query : `call SF_GSTS_11('number_of_months')`
- The same goes with **SF_GSTS_14**, we were asked to create a query returning the payment method that people used the most since a certain date. We used the same principle as above. The query will be called as following : `call SF_GSTS_14('insert_date')`

We did the same for **SF_GSTS_07 , SF_GSTS_09, SF_GSTS_13.** Specifics amount of months or specifics means of transportation were asked as condition for these queries. We generalized them by allowing to enter the desired parameter. For example for **SF_GSTS_07,** you'll specify which means of transportation you want, and it will return to you the trips that only use these specific means of transportation.

About the **SF_GSTS_12** query. It asked "the average of seniors travelling the past six months". We understood this as the average of age of the seniors travelling the past six months. Again we generalized it so that you could choose the "age" from which you consider someone to be a senior and the amount of months you desire. This request was working in a past version of our database where we stored the age of our clients. But such an organization would make our database obsolete through time. Instead, we replaced this attribute by their date of birth. We can obviously deduce from this the age of a person. But we quickly encountered some problems. The main one is that the function used to do so only return a one and only value, we couldn't get the ages of all the people in a new column (where we would have calculate the average). There's obviously a solution to this, but time didn't played in our favor and we couldn't satisfy our deadlines.

All other queries will not be described or explained here since they simply answer the demand. You can find all the calls for add/modify/delete.. queries in a file named "call.sql". They are pre-filled and come with comments.