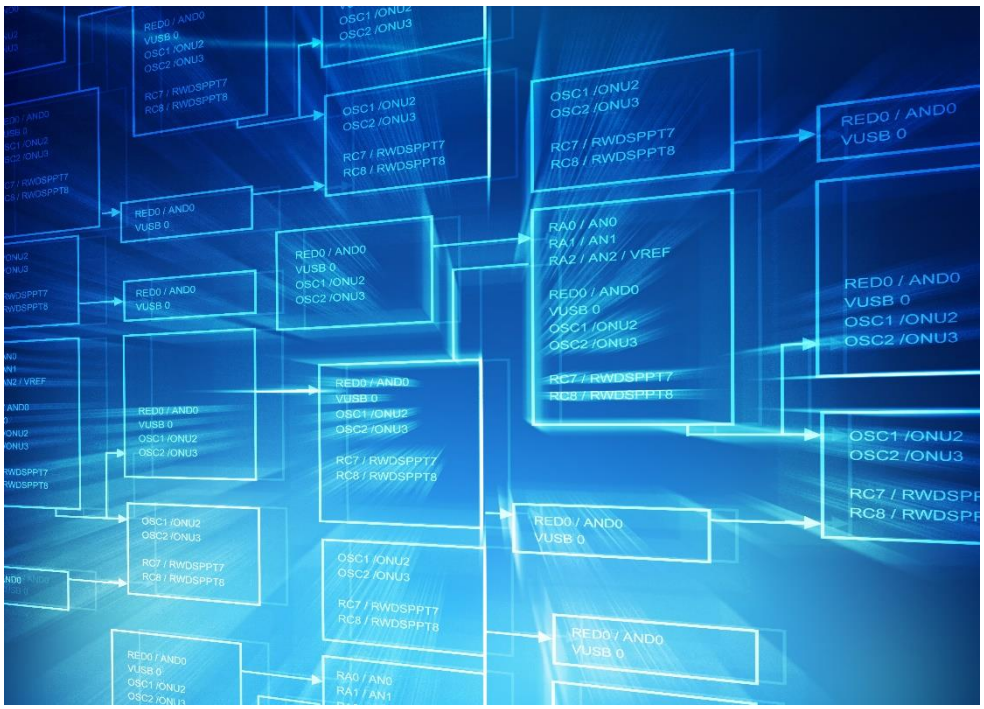
DATA PROJECT AND MODELLING



VERLAAN Vincent

MARCILLA Théo

COUERRE Pierre-Yves

SAURAIS Gaël

Table of contents

I-Context

The travel agency wants to digitize its data system. Every data in this document is from interview led by the decision-making representatives of our client.

- Customer Management
- Personal Management
- Ticket office Management
- Conveyance Management
- Steps Management
- Statistics Management

All these categories have its own set of features.

II-The project team

We are a group of 1st year students in preparatory cycle at CESI Bordeaux. Our goal is to carry out this project and to ensure its well-being.

Its members are: Gaël SAURAIS, Vincent VERLAAN, Théo MARCILLA and Pierre-Yves COUERRE.

III-WBS/OBS

1) WBS (Work Breakdown Structure)

Work Breakdown Structure or WBS subdivide a project into specific tasks to improve productivity and make tasks more manageable and more flexible. In term of task management, the work breakdown structure is the perfect tool to be as efficient as possible. By unifying the range, the cost and the work plan of the project, the whole WBS structure is the core of the project management.

The Work Breakdown Structure has two types:

-1) By deliverables

-2) In phases

The most common approach is by deliverables. It consists of having multiple deadlines for each step of the project. The main difference between those two approaches is the nature of the elements on the first level of the structure WBS.

The WBS tool is designed to simplify the organization of the project, establish the preferential planning and the provisional budget. It also helps each actor to be delegated and contracted with its own mission.

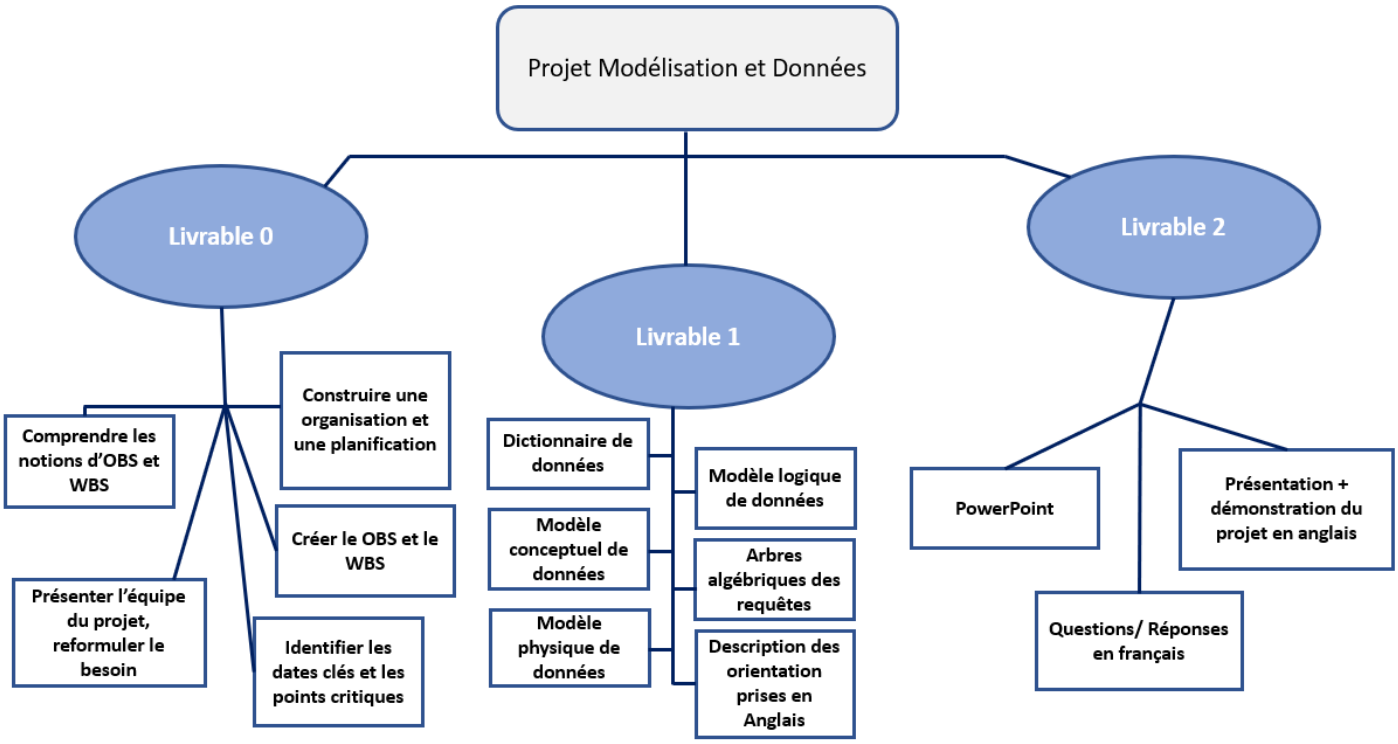


Diagram of the structure of a WBS project modeling and data

2) OBS (Organizational Breakdown Structure)

Organization Breakdown Structure or OBS is a hierarchical model that describe the established organizational framework for project planning, resource management, time and expense tracking, cost allocation, revenue and/or profit reporting and work management.

The main goal is to communicate the way that people in charge of the project have to organize and team up to be the most efficient.

The OBS method allows:

- Officially define the personal in charge of the project.
- Facilitate the coordination and the audit.
- Focus on the actors of the project.
- Improve the communication in the project team.

It can be created the same way that you create a WBS.

- 1) Identify the management structure for all the resources implied in the project.
- 2) Once the structure is filled, you have to identify every team member and to attribute to everyone a specific position in the structure.
- 3) Be sure that the OBS is structured from the highest responsible department to the lowest responsible.

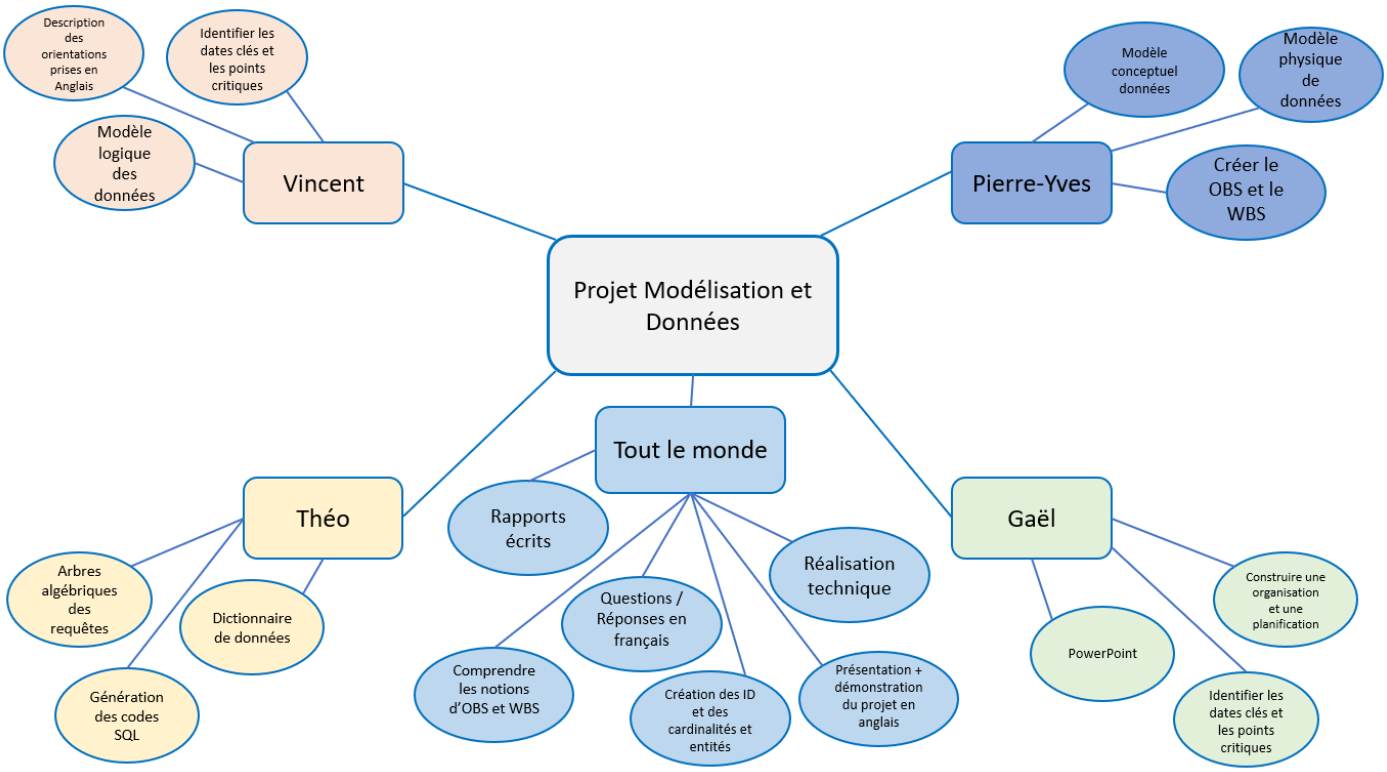


Diagram of the structure of an OBS of the modeling and data project

IV-The different models

A) Data dictionary

To start creating a database. The first step is to know and analyze the client's needs. Here, a specification has been provided to us to analyze these needs.

All the Relational DataBase Management Systems (RDBMS) have a data dictionary which is a collection of tables with metadata stored and automatically updated.

The tables can be seen with a SQL script.

It describes data like the customers, the products classification, the directory. It is the baseline of the company. It is usually represented by a table with four columns with the name, code, type of data and the comments.

We created our data dictionary on JMerise, by giving clear names and codes. We put auto_increment for the IDs which are primary keys, and other types basing on the data we are expecting. The maximum size we put is 50 to let enough of space for the data. (Cf. dictionary data)

Here is an extract of our data dictionary.

<u>Right below, an extract of our data dictionary :</u>

Clients	ID_client	Auto Inc	-
	Nom_client	Alphanumérique	50
	Prenom_client	Alphanumérique	50
	Numero_telephone_mobile_client	Alphanumérique	50
	Adresse_mail_client	Alphanumérique	50
Adresse_facturation_client	ID_Adresse_facturation	Auto Inc	-
	Numero_voie_client	Alphanumérique	50
	Ville_client	Alphanumérique	50
	Nom_rue_client	Alphanumérique	50
	Nom_residence_client	Alphanumérique	50
	Nom_batiment_client	Alphanumérique	50
	Etage_client	Numérique	-
	Code_postal_client	Alphanumérique	50
Adresse_livraison_client	ID_Adresse_livraison_client	Auto Inc	-
	Numero_voie_client_livraison	Alphanumérique	50
	Ville_client_livraison	Alphanumérique	50
	Nom_rue_client_livraison	Alphanumérique	50
	Nom_residence_client_livraison	Alphanumérique	50
	Nom_batiment_client_livraison	Alphanumérique	50
	Etage_client_livraison	Numérique	-
	Code_postal_client_livraison	Alphanumérique	50
Personnel	ID_Personnel	Auto Inc	-
	Nom_personnel	Alphanumérique	50
	Prenom_personnel	Alphanumérique	50
	Numero_de_telephone mobile_personnel	Alphanumérique	50
	Adresse_mail_entreprise_personnel	Alphanumérique	50
	Date_embauche_personnel	Date	-
Adresse_personnel	ID_Adresse_personnel	Auto Inc	-
	Numero_voie_personnel	Alphanumérique	50
	Ville_personnel	Alphanumérique	50
	Nom_de_la_rue_personnel	Alphanumérique	50
	Nom_de_la_residence_personnel	Alphanumérique	50
	Nom_batiment_client	Alphanumérique	50
	Etage_personnel	Numérique	-
	Code_postal_personnel	Alphanumérique	50
Voyage	ID_voyage	Auto Inc	-
	Nom_voyage	Alphanumérique	50
	Nombre_etape_voyage	Numérique	-
	Date_commande_voyage	Date	-

Etapes	ID_etape	Auto Inc	-
	Nom_etape	Alphanumérique	50
	Ville_depart	Alphanumérique	50
	Ville_arrivee	Alphanumérique	50
	Date_heure_depart	Date-time	-
	Date_heure_arrivee	Date-time	-
Moyen_transport	ID_moyen_transport	Auto Inc	-
	Type_transport	Alphanumérique	50
	Nombre_transport_utilise	Alphanumérique	50
	Cout_transport	Numérique	-
Distance	ID_distance	Auto Inc	-
	Valeur_distancier	Numérique	-
	Cumul_kilometrage	Numérique	-
Point_etape_inter	ID_point_etape_inter	Auto Inc	-
	Nom_ville_inter	Alphanumérique	50
	Ville_depart_inter	Alphanumérique	50
	Ville_arrivee_inter	Alphanumérique	50
	Date_heure_depart_inter	Date-time	-
	Date_heure_arrivee_inter	Date-time	-
Tarification	ID_Paiment	Auto Inc	-
	Nombre_paiement_voyage	Numerique	-
	Date_Paiment_voyage	Date	-
	Mode_Paiment_voyage	Alphanumérique	50
	Montant_paiement_voyage	Numérique	-
	Prix_voyage	Numérique	-
Point_etape_intra_ville	ID_point_etape_intra_ville	Auto Inc	-
	Nom_point_etape_intra_ville	Alphanumérique	50
	Forfait_intra_ville	Numérique	-

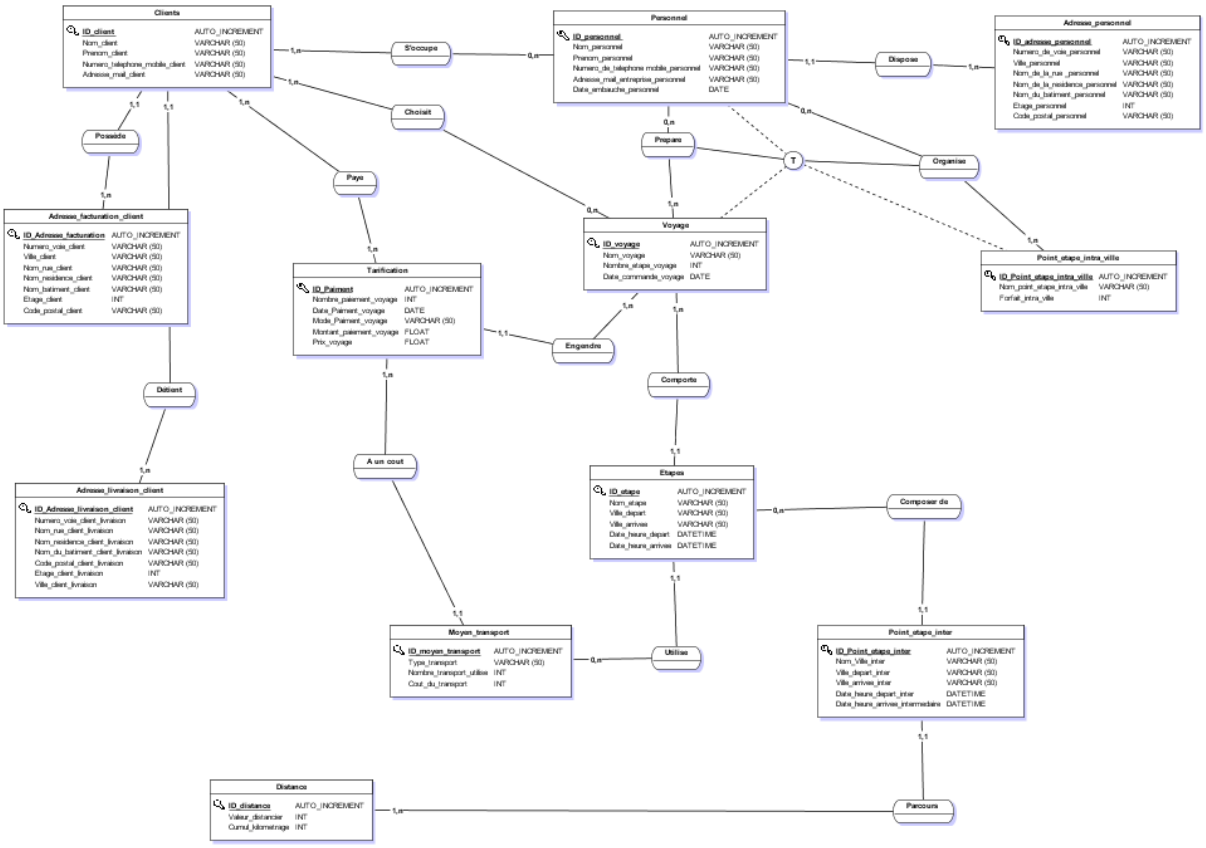
B) The conceptual data model

The CDM models the data without technical, management and economic constraints.

In order to realize the CDM, we modified it several times. First, we checked the instructions, and we took all the information from it. From the data dictionary, we created several entities and links between them. Then, we divided the entities to reduce them and to make the database easier to use. For example, the billing and shipping address of the customer became two distinct entities. We also used a totality constraint which connects the entity “personnel” with the entity “voyage” and “point_etape_intra_ville”. It means that “personnel” can participate at least at one of the two entities.

3rd normal form: a non-key attribute should not determine other non-key attributes.

All our primary keys are auto incremented, and the attributes are independent.



The conceptual data model

C) The logical data model

The Logical Data Model (or LDM) is the further step of the CDM.

Different techniques are possible:

An LDM in the form of files. The data can be stored in files, using it through a small number of procedures that will be written entirely. The LDM is then the format of the data in the files.

An LDM in the form of an XML-type hierarchical database.

An LDM in the form of a relational database.

An LDM in the form of an object database.

The 6 rules to switch from MCD to LDM:

Rule 1 -Entity: Each entity becomes a table. Each attribute of the entity becomes an attribute of that table.

Rule 2-Association "1 to many": The primary key of the upper entity (many side) becomes a foreign key attribute in the table from the lower entity (rate 1). In the case of a reflexive "1-to-many" association, this new attribute must be renamed. In the case of a relative identifier (association (1.1) parenthesis), the primary key of the upper entity (many side) becomes a foreign and primary key attribute in the table from the lower entity (rate 1).

Rule 3-Association "Many to Many ": A " Many to Many " association becomes a table. The primary keys of associated entities become foreign keys in this table. The attributes of the association become attributes of the table. Determining the primary key of this table is not automatic. In general, the primary key of this table consists of the concatenation of the primary keys of the associated entities. However, the question is whether this concatenation forms the primary key. If this is not the case, one can try adding non-key attributes to find the primary key. Then you have to ask yourself if you can remove some of the key foreign attributes to reduce the primary key to a minimum of attributes.

Rule 4 - Association "0.1 to several": 2 possibilities:

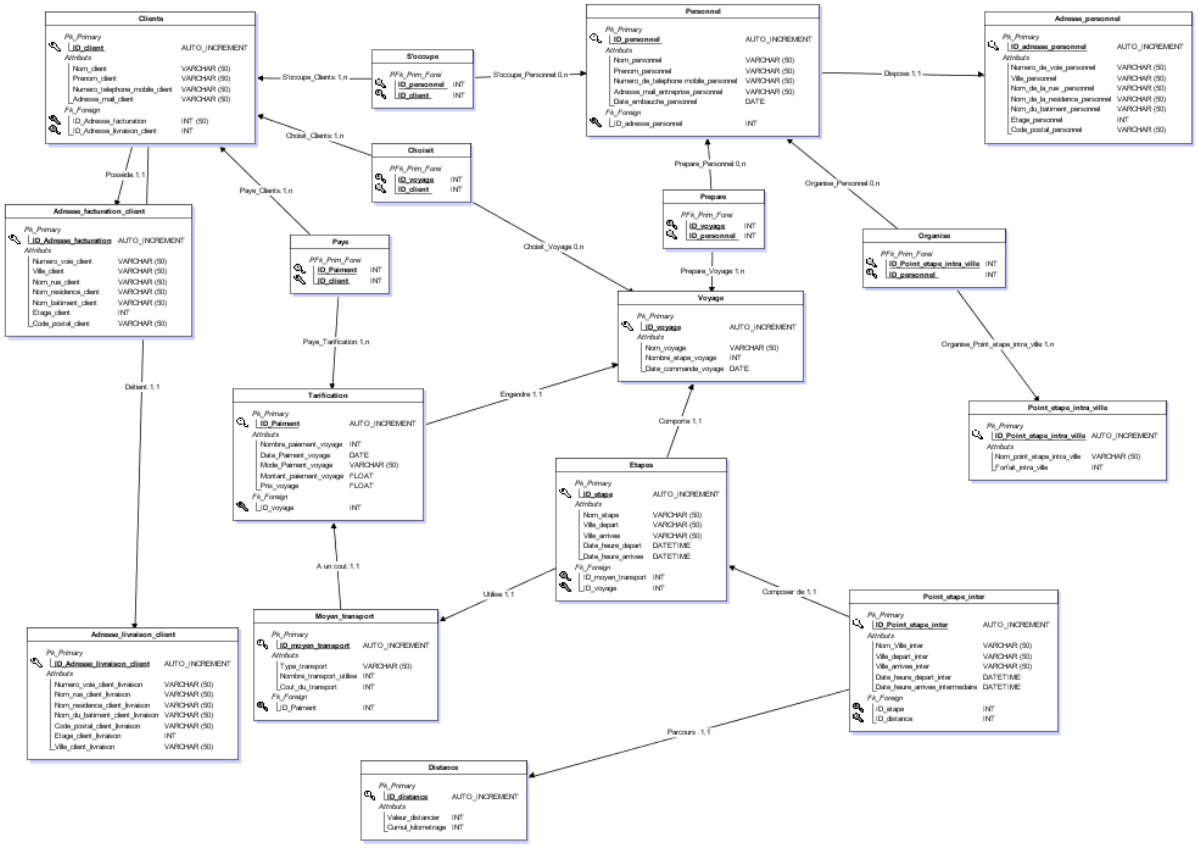
If they have attributes, rule 3 is applied, dealing with several associations. The association gives a table.

If they do not have attributes, rule 2 is applied for associations 1 to several. In this case the foreign key produced is not mandatory since the minimum is at 0 (not NOT NULL).

Rule 5-Inheritance: In the case of an inheritance, each participating entity (species and genus) becomes a table. The primary key of the table from the genus entity becomes a foreign key in the tables from the species entities. If a species entity does not have a primary key, the foreign key derived from the genus entity becomes the primary key to the table derived from the species entity.

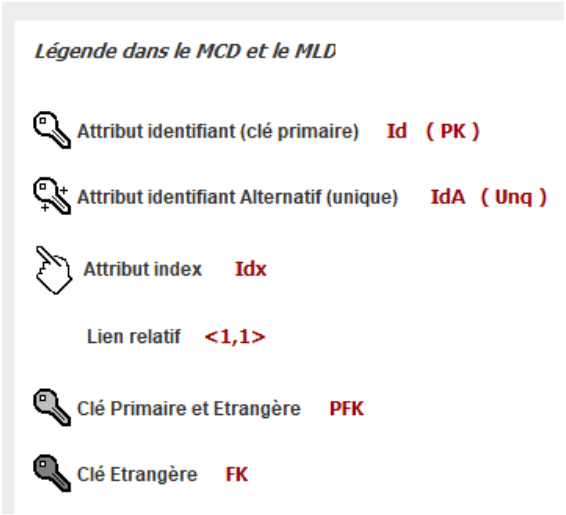
Rule 6-The Complex Key: Complex keys are produced by applying the previous 5 rules with one difference: associations can link entities or associations that

give a table in the relational model. In this case the first rules 5 rules apply by considering the primary key of the table from the linked association as if it came from an entity.



The logical data model

Légende



D) The physical data model

The Physical data model describes the data from a database in a specific syntax. MySQL uses the query language SLQ. We can also use a software that will translate our logical data model into an SQL script.
we will create the tables with this command:

```
CREATE TABLEXX (Property xx type (xx), Property xx type (xx), CONSTRAINTXX PRIMARY KEY (XX))ENGINE=InnoDB;
```

We made the SQL script, which creates the different tables and attributes of the database, from the CDM.

Here is the SQL script.

```sql
#------------------------------------------------------------
# Table: Voyage
#------------------------------------------------------------

CREATE TABLE Voyage(
        ID_voyage            Int  Primary KEY NOT NULL Auto_increment,
        Nom_voyage           Varchar (50) NOT NULL ,
        Nombre_etape_voyage  Int NOT NULL ,
        Date_commande_voyage Date NOT NULL
);
#------------------------------------------------------------
# Table: Distance
#------------------------------------------------------------

CREATE TABLE Distance(
        ID_distance       Int  PRIMARY KEY NOT NULL Auto_increment,
        Valeur_distancier Int NOT NULL,
        Cumul_kilometrage Int NOT NULL
);


#------------------------------------------------------------
# Table: Adresse_facturation_client
#------------------------------------------------------------

CREATE TABLE Adresse_facturation_client(
        ID_Adresse_facturation Int  PRIMARY KEY NOT NULL Auto_increment,
        Numero_voie_client     Varchar (50) NOT NULL ,
        Ville_client           Varchar (50) NOT NULL ,
        Nom_rue_client         Varchar (50) NOT NULL ,
        Nom_residence_client   Varchar (50) NOT NULL ,
        Nom_batiment_client    Varchar (50) NOT NULL ,
        Etage_client           Int NOT NULL ,
        Code_postal_client     Varchar (50) NOT NULL
);


#------------------------------------------------------------
# Table: Adresse_livraison_client
#------------------------------------------------------------

CREATE TABLE Adresse_livraison_client(
        ID_Adresse_livraison_client      Int  PRIMARY KEY Auto_increment  NOT NULL ,
        Numero_voie_client_livraison     Varchar (50) NOT NULL ,
        Nom_rue_client_livraison         Varchar (50) NOT NULL ,
        Nom_residence_client_livraison   Varchar (50) NOT NULL ,
        Nom_du_batiment_client_livraison Varchar (50) NOT NULL ,
        Code_postal_client_livraison     Varchar (50) NOT NULL ,
        Etage_client_livraison           Int NOT NULL ,
        Ville_client_livraison           Varchar (50) NOT NULL
);


#------------------------------------------------------------
# Table: Clients
#------------------------------------------------------------

CREATE TABLE Clients(
        ID_client                        Int  PRIMARY KEY Auto_increment  NOT NULL ,
        Nom_client                       Varchar (50) NOT NULL ,
        Prenom_client                    Varchar (50) NOT NULL ,
        Numero_telephone_mobile_client   Varchar (50) NOT NULL ,
        Adresse_mail_client              Varchar (50) NOT NULL ,
        ID_Adresse_facturation           Int NOT NULL ,
        ID_Adresse_livraison_client      Int NOT NULL

    ,CONSTRAINT Clients_Adresse_facturation_client_FK FOREIGN KEY (ID_Adresse_facturation) REFERENCES Adresse_facturation_client(ID_Adresse_facturation)
    ,CONSTRAINT Clients_Adresse_livraison_client0_FK FOREIGN KEY (ID_Adresse_livraison_client) REFERENCES Adresse_livraison_client(ID_Adresse_livraison_client)
)ENGINE=InnoDB;


#------------------------------------------------------------
# Table: Adresse_personnel
#------------------------------------------------------------

CREATE TABLE Adresse_personnel(
        ID_adresse_personnel         Int  PRIMARY KEY Auto_increment  NOT NULL ,
        Numero_de_voie_personnel     Varchar (50) NOT NULL ,
        Ville_personnel              Varchar (50) NOT NULL ,
        Nom_de_la_rue__personnel     Varchar (50) NOT NULL ,
        Nom_de_la_residence_personnel Varchar (50) NOT NULL ,
        Nom_du_batiment_personnel    Varchar (50) NOT NULL ,
        Etage_personnel              Int NOT NULL ,
        Code_postal_personnel        Varchar (50) NOT NULL
);


#------------------------------------------------------------
# Table: Personnel
#------------------------------------------------------------

CREATE TABLE Personnel(
        ID_personnel                        Int  PRIMARY KEY Auto_increment  NOT NULL ,
        Nom_personnel                       Varchar (50) NOT NULL ,
        Prenom_personnel                    Varchar (50) NOT NULL ,
        Numero_de_telephone_mobile_personnel Varchar (50) NOT NULL ,
        Adresse_mail_entreprise_personnel   Varchar (50) NOT NULL ,
        Date_embauche_personnel             Date NOT NULL ,
        ID_adresse_personnel                Int NOT NULL
);


#------------------------------------------------------------
# Table: Point_etape_intra_ville
#------------------------------------------------------------

CREATE TABLE Point_etape_intra_ville(
        ID_Point_etape_intra_ville  Int  PRIMARY KEY Auto_increment  NOT NULL ,
        Nom_point_etape_intra_ville Varchar (50) NOT NULL ,
        Forfait_intra_ville         Int NOT NULL
);


#------------------------------------------------------------
# Table: Tarification
#------------------------------------------------------------
```

```
CREATE TABLE Tarification(
        ID_Paiment              Int  PRIMARY KEY Auto_increment  NOT NULL ,
        Nombre_paiement_voyage  Int NOT NULL ,
        Date_Paiment_voyage     Date NOT NULL ,
        Mode_Paiment_voyage     Varchar (50) NOT NULL ,
        Montant_paiement_voyage Float NOT NULL ,
        Prix_voyage             Float NOT NULL ,
        ID_voyage               Int NOT NULL
);


#------------------------------------------------------------
# Table: Moyen_transport
#------------------------------------------------------------

CREATE TABLE Moyen_transport(
        ID_moyen_transport       Int  primary key Auto_increment  NOT NULL ,
        Type_transport           Varchar (50) NOT NULL ,
        Nombre_transport_utilise Int NOT NULL ,
        Cout_du_transport        Int NOT NULL ,
        ID_Paiment               Int NOT NULL
);


#------------------------------------------------------------
# Table: Etapes
#------------------------------------------------------------

CREATE TABLE Etapes(
        ID_etape           Int  PRIMARY KEY Auto_increment  NOT NULL ,
        Nom_etape          Varchar (50) NOT NULL ,
        Ville_depart       Varchar (50) NOT NULL ,
        Ville_arrivee      Varchar (50) NOT NULL ,
        Date_heure_depart  Datetime NOT NULL ,
        Date_heure_arrivee Datetime NOT NULL ,
        ID_moyen_transport Int NOT NULL ,
        ID_voyage          Int NOT NULL
);


#------------------------------------------------------------
# Table: Point_etape_inter
#------------------------------------------------------------

CREATE TABLE Point_etape_inter(
        ID_Point_etape_inter            Int  primary key Auto_increment  NOT NULL ,
        Nom_ville_inter                 Varchar (50) NOT NULL ,
        Ville_depart_inter              Varchar (50) NOT NULL ,
        Ville_arrivee_inter             Varchar (50) NOT NULL ,
        Date_heure_depart_inter         Datetime NOT NULL ,
        Date_heure_arrivee_intermedaire Datetime NOT NULL ,
        ID_etape                        Int NOT NULL ,
        ID_distance                     Int NOT NULL
);


#------------------------------------------------------------
# Table: S'occupe
#------------------------------------------------------------

CREATE TABLE S_occupe(
        ID_personnel   Int PRIMARY KEY NOT NULL ,
        ID_client      Int PRIMARY KEY NOT NULL
);


#------------------------------------------------------------
# Table: Prepare
#------------------------------------------------------------

CREATE TABLE Prepare(
        ID_voyage      Int primary key NOT NULL ,
        ID_personnel   Int primary key NOT NULL
);


#------------------------------------------------------------
# Table: Organise
#------------------------------------------------------------

CREATE TABLE Organise(
        ID_Point_etape_intra_ville Int primary key NOT NULL ,
        ID_personnel               Int primary key NOT NULL
);


#------------------------------------------------------------
# Table: Choisit
#------------------------------------------------------------

CREATE TABLE Choisit(
        ID_voyage   Int primary key NOT NULL ,
        ID_client   Int primary key NOT NULL
);


#------------------------------------------------------------
# Table: Paye
#------------------------------------------------------------

CREATE TABLE Paye(
        ID_Paiment  Int primary key NOT NULL ,
        ID_client   Int primary key NOT NULL
);
```

Then we move on to the creation of foreign keys :

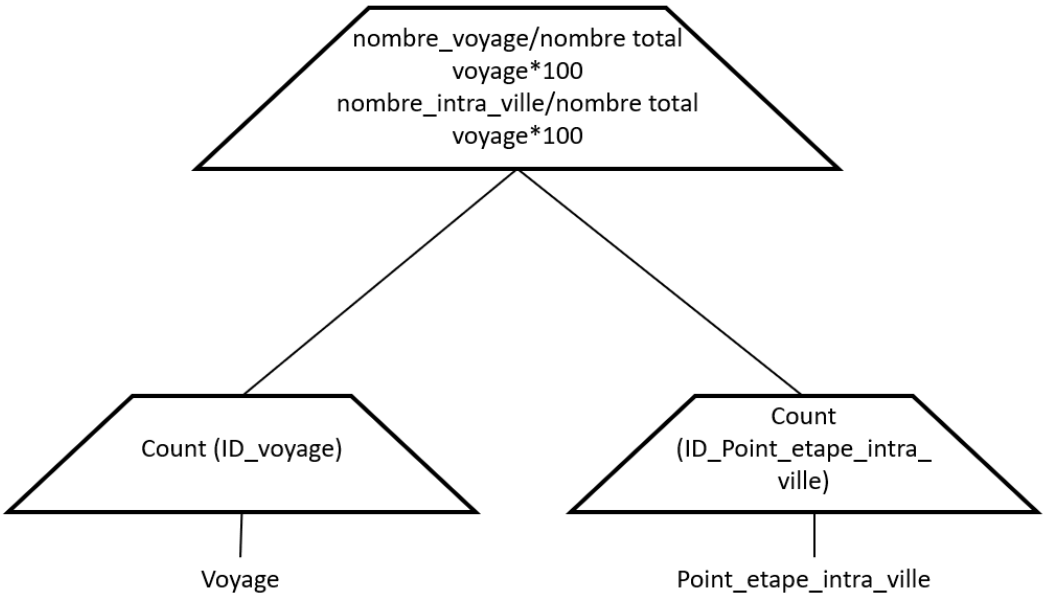
```
225 ALTER TABLE Clients add foreign key Clients_Adresse_facturation_client_FK (ID_Adresse_facturation) REFERENCES Adresse_facturation_client(ID_Adresse_facturation);
226 ALTER TABLE Clients add foreign key Clients_Adresse_livraison_client0_FK (ID_Adresse_livraison_client) REFERENCES Adresse_livraison_client(ID_Adresse_livraison_client);
227 ALTER TABLE Personnel add foreign key Personnel_Adresse_personnel_FK (ID_adresse_personnel) REFERENCES Adresse_personnel(ID_adresse_personnel);
228 ALTER TABLE Tarification add foreign key Tarification_Voyage_FK (ID_voyage) REFERENCES Voyage(ID_voyage);
229 ALTER TABLE Moyen_transport add foreign key Moyen_transport_Tarification_FK (ID_Paiment) REFERENCES Tarification(ID_Paiment);
230 ALTER TABLE Etapes add foreign key Etapes_Moyen_transport_FK (ID_moyen_transport) REFERENCES Moyen_transport(ID_moyen_transport);
231 ALTER TABLE Etapes add foreign key Etapes_Voyage0_FK (ID_voyage) REFERENCES Voyage(ID_voyage);
232 ALTER TABLE Point_etape_inter add foreign key Point_etape_inter_Etapes_FK (ID_etape) REFERENCES Etapes(ID_etape);
233 ALTER TABLE Point_etape_inter add foreign key Point_etape_inter_Distance0_FK (ID_distance) REFERENCES Distance(ID_distance);
234 ALTER TABLE S_occupe add foreign key S_occupe_Personnel_FK (ID_personnel) REFERENCES Personnel(ID_personnel);
235 ALTER TABLE S_occupe add foreign key S_occupe_Clients0_FK (ID_client) REFERENCES Clients(ID_client);
236 ALTER TABLE Prepare add foreign key Prepare_Voyage_FK (ID_voyage) REFERENCES Voyage(ID_voyage);
237 ALTER TABLE Prepare add foreign key Prepare_Personnel0_FK (ID_personnel) REFERENCES Personnel(ID_personnel);
238 ALTER TABLE Organise add foreign key Organise_Point_etape_intra_ville_FK (ID_Point_etape_intra_ville) REFERENCES Point_etape_intra_ville(ID_Point_etape_intra_ville);
239 ALTER TABLE Organise add foreign key Organise_Personnel0_FK (ID_personnel) REFERENCES Personnel(ID_personnel);
240 ALTER TABLE Choisit add foreign key Choisit_Voyage_FK (ID_voyage) REFERENCES Voyage(ID_voyage);
241 ALTER TABLE Choisit add foreign key Choisit_Clients0_FK (ID_client) REFERENCES Clients(ID_client);
242 ALTER TABLE Paye add foreign key Paye_Tarification_FK (ID_Paiment) REFERENCES Tarification(ID_Paiment);
243 ALTER TABLE Paye add foreign key Paye_Clients0_FK (ID_client) REFERENCES Clients(ID_client);
```

The procedures will come in complement to simplify the management of data

V-Algebraic trees

Basically, algebraic trees are the representation of a SQL query. It looks like an upside-down tree and it's composed of branchs and leaves. The leaves represent the starting tables where you get your data from and the root of the tree (the highest point) is the table you get from the query you ordered. Each node of the tree is a relational algebra operator. A relational algebra operator can be many things but it's usually either a union, a difference, a cartesian product or it can even be a join between two or more tables.

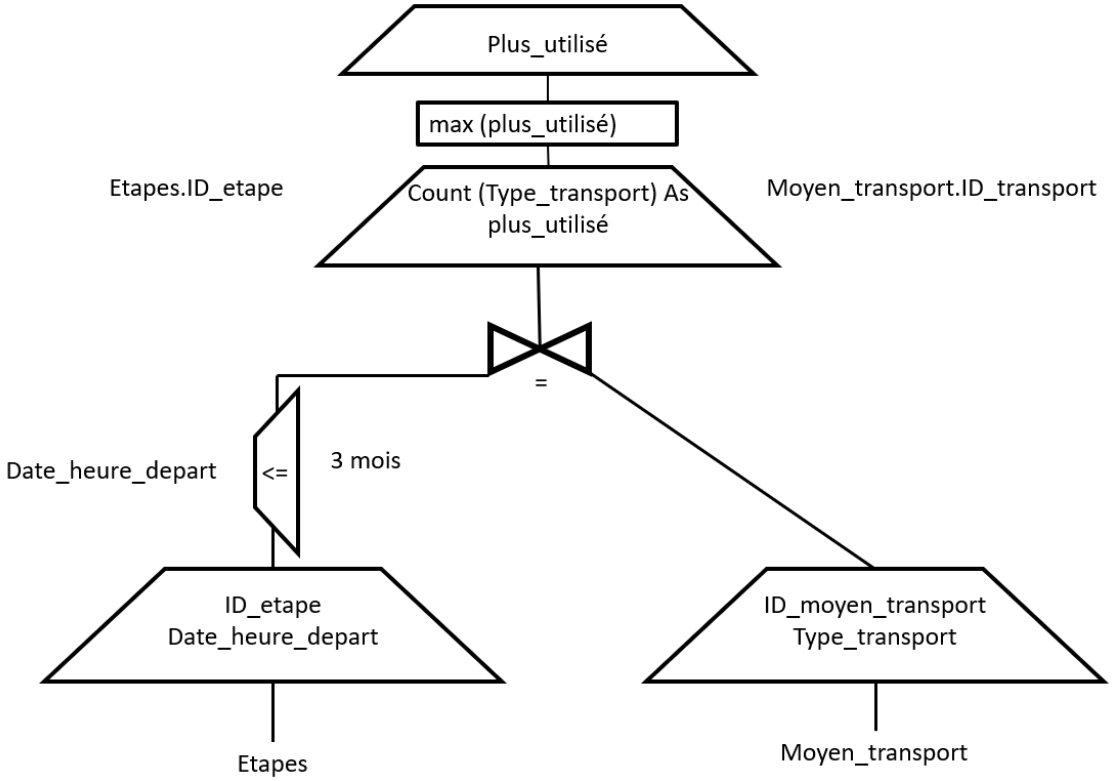
• SF_GSTS_05 - This tree allow us to get the most used transport over the last three months



Algebraic tree of the SF_GSTS_05 request

The objective here is to be able to know the proportion of Intra-and Inter-city travel. So, we made a selection on a projection and a projection followed by a junction, then a projection and finally a sorting.

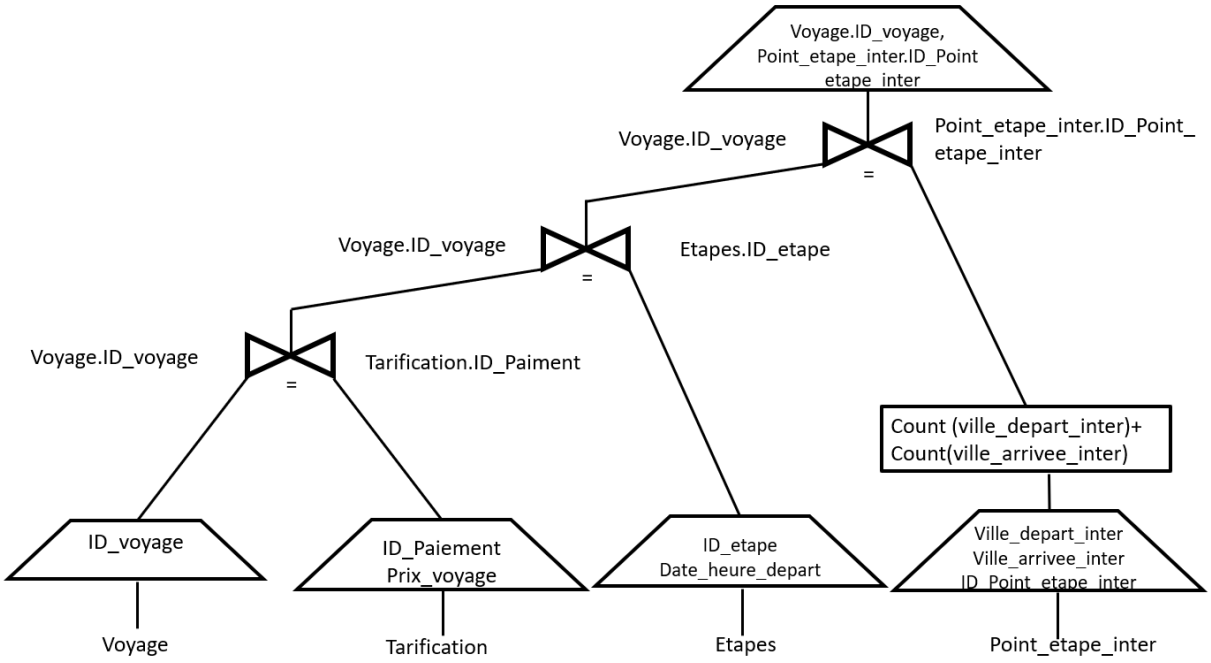
• SF_GSTS_09 - The objective here is to be able to know the proportion of Intra-and Inter-city travel



Algebraic tree of the SF_GSTS_09 request

This tree allows us to get the most used transport over the last three months by using two projections. We needed a percentage that's why we used the rule of three.

• SF_GSTS_16 - Be able to know the different information for the trip : number of the trip, it price, departure date, how many cities crossed



Algebraic tree of the SF_GSTS_16 request

To know the different information for the trip: number of the trip, it price, departure date, how many cities crossed; we made multiple projections followed by different junctions and a count after the last projection.

VI-Conclusion

In conclusion, we created the data dictionary, the different properties for each table, the CDM, the LDM and the database. Furthermore we achieved every single objective of the deliverable. For the future, we will have to carry out the different queries requested by the customer and create procedures that allow greater ease in the manipulation of the data. To do this, it will obviously be necessary to generate data and place them in the different tables. Finally, we will prepare a presentation explaining all the stages of the project and a demonstration.

VII-Bibliographic references

https://ineumann.developpez.com/tutoriels/merise/initiation-merise/#LIV-D

http://perso.modulonet.fr/placurie/Ressources/BTS1-Cgo1/Chap_6_Dictionnaire_des_donnees.pdf

https://merise.developpez.com/faq/?page=MCD#CIF-ou-dependance-fonctionnelle-de-A-a-Z

https://fr.wikipedia.org/wiki/Dictionnaire_des_donn%C3%A9es

https://web.maths.unsw.edu.au/~lafaye/CCM/merise/control.htm

http://bliaudet.free.fr/IMG/pdf/INSIA-SIGL_3-02-Optimisation_-_arbres_algebriques-2.pdf