


M1 Informatique AIGLE

HMIN122M

Mini-Projet : entrepôts de données

Rapport

Bachar Rima
Joseph Saba
Tasnim Shaqura Muhammad
Jérémy Bourgin

9 novembre 2018

Table des matières

Chapitre 1

Introduction

1.1 Problématiques

Dans le cadre du mini-projet du module **HMIN122M**, nous avons décidé de modéliser un entrepôt de données pour le réseau de transport publique de Montpellier, *tam-voyages*. Pour ce faire, nous avons proposé des *data marts* formant le *data warehouse*, et permettant de réaliser des requêtes analytiques sur un ensemble important de données. Cette modélisation permettra ainsi de mettre en œuvre un outil d'analyse pour répondre aux problématiques suivantes :

1. Comment peut-on tirer partie de la fréquentation des véhicules en se basant sur la circulation du réseau [1] afin d'améliorer la qualité de service (e.g. *nombre de véhicules à envoyer sur une ligne pendant une certaine période de la journée*) ?
2. Comment peut-on suivre l'évolution et la maintenance des matériaux de manière à réduire les dépenses associées ?

Pour répondre à ces problématiques, nous énumérons les actions et opérations effectuées par *tam-voyages*. Ensuite, nous choisirons celles qui paraissent les plus pertinentes en termes de données intégrées et flexibilité des critères d'analyse.

1.2 Actions et opérations

Les actions/opérations effectuées par *tam-voyages* considérées :
— les voyages.
— la maintenance de véhicules.

1. en particulier en examinant les lignes de tramway et les bus

— les ventes de tickets et les abonnements.
— les amendes.

1.3 Exemples de requêtes analytiques possibles

1. exemples de requêtes analytiques pour l'action « voyages » :
   — `le nombre de voyages par bus, utilisant des tickets pour le mois de juillet.`
   — `le nombre de voyageurs abonnés par ligne pour chaque voyage pour les deux derniers mois.`
   — `l'arrêt le plus fréquenté par ligne.`
2. exemples de requêtes analytiques pour l'action « maintenance » :
   — `le coût total de maintenance de chaque véhicule.`
   — `le nombre total de maintenances effectuées sur les bus par employé pour l'année 2018.`
   — `le nombre total de maintenances effectuées par véhicule pour les 6 dernier mois.`
3. exemples de requêtes analytiques pour l'action « ventes » :
   — `le nombre d'abonnés ayant plus que 26 ans pour le mois d'août 2018.`
   — `le nombre d'abonnés par date de naissance pour l'année 2018.`
   — `les types d'abonnement les plus fréquents pour l'année 2018.`
4. exemples de requêtes analytiques pour l'action « amendes » :
   — `les lignes qui ont générées le plus d'amendes pour les deux derniers mois.`
   — `les lignes les plus contrôllées de la semaine dernière.`
   — `le nombre des abonnés qui ont reçu des amendes par ligne, l'avant-midi.`
   — `la somme total d'amendes rapportée par type de voyageur par ligne pour le dernier mois.`

Chapitre 2

Modélisation

2.1 Choix des actions à modéliser

Les actions considérées, par ordre d'importance :

1. « voyages ».
2. « ventes ».
3. « maintenance ».
4. « amendes ».

Les actions les plus pertinentes à analyser vis-à-vis les problématiques avancées sont « voyages »et « maintenance »qu'on traitera de la manière suivante :

voyages : modèle en étoile détaillé.

maintenance : modèle en étoile *moins* détaillé, en particulier le modèle intitulé "*record transaction*".

2.2 Modélisation des voyages

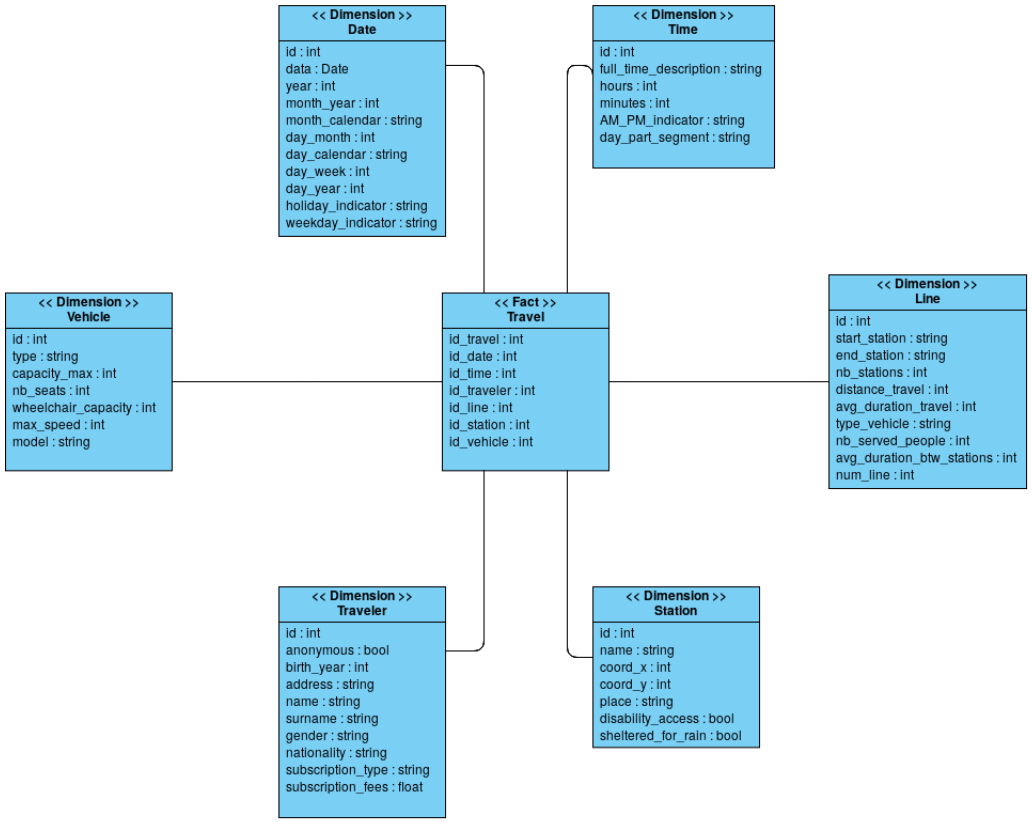


FIGURE 2.1 – modèle en étoile de l'action « voyages »

2.2.1 Discussion

— un fait correspond à un voyage (`Travel`) effectué par un voyageur à un temps donné et une date donnée, en prenant un véhicule d'une ligne depuis un arrêt spécifié.
— chaque tuple dans la table `Vehicle` désigne un véhicule possédé par la société.
— chaque tuple dans la table `Ligne` désigne une itinéraire prise par une ligne. Autrement dit, la même ligne peut avoir plusieurs itinéraires différentes (*e.g. la ligne 3 possède 4 itinéraires différentes*)
— chaque tuple dans la table `Station` désigne une station desservie par un véhicule.
— la table `Traveler` est une dimension qui contient deux dimensions corrélées (les abonnés et le voyageur anonyme non abonné, utilisant un ticket) :

1. si le voyageur est **abonné**, alors on traite le tuple correspondant en

tant qu'un **voyageur concret** dont les informations sont à notre disposition.

2. sinon, tous les **voyageurs non abonnés** seront représentés par un seul tuple.
3. cette décision de corrélation est utilisée pour éviter la normalisation et l'introduction d'une superclasse abstraite étendue par les classes désignant les voyageurs abonnés et non abonnés.
4. nous utilisons ainsi l'attribut **anonymous** afin de distinguer les deux types de voyageurs. En effet, **anonymous** valera *true* quand le voyageur est non abonné, sinon il valera *false*.
5. le tuple du **voyageur non abonné** contiendra ainsi des **valeurs nulles** pour les attributs décrivant un **voyageur abonné**.

La table des voyages n'admet aucune mesure ; on se contentera d'utiliser les informations fournies par les dimensions qui suffiront largement pour analyser la fréquentation des différentes lignes et véhicules correspondant.

Remarques

— l'attribut **id_travel** de la table `Travel` est la clé primaire utilisée pour identifier un voyage (*dimension dégénérée*).
— l'attribut **nb_served_people** de la table `Line` désigne le nombre de passagers désservis par la ligne.
— l'attribut **place** de la table `Station` désigne l'endroit où se trouve la station (*e.g. avenue X, rue Y, ...*).
— l'attribut **wheelchair_capacity** de la table `Vehicle` désigne la capacité théorique maximale de personnes handicappées et de leurs fauteuils roulants.

2.3 Modélisation des maintenances

2.3.1 Discussion

— un fait désigne l'état d'une opération lors de la maintenance `Maintenance` d'un véhicule (*i.e. le grain choisi pour la maintenance est la transaction*).
— chaque tuple dans la table `Employee` désigne un employé chez *tam-voyages*.
— chaque tuple dans la table `TechnicalArea` désigne un dépôt utilisé par *tam-voyages* pour maintenir des véhicules.

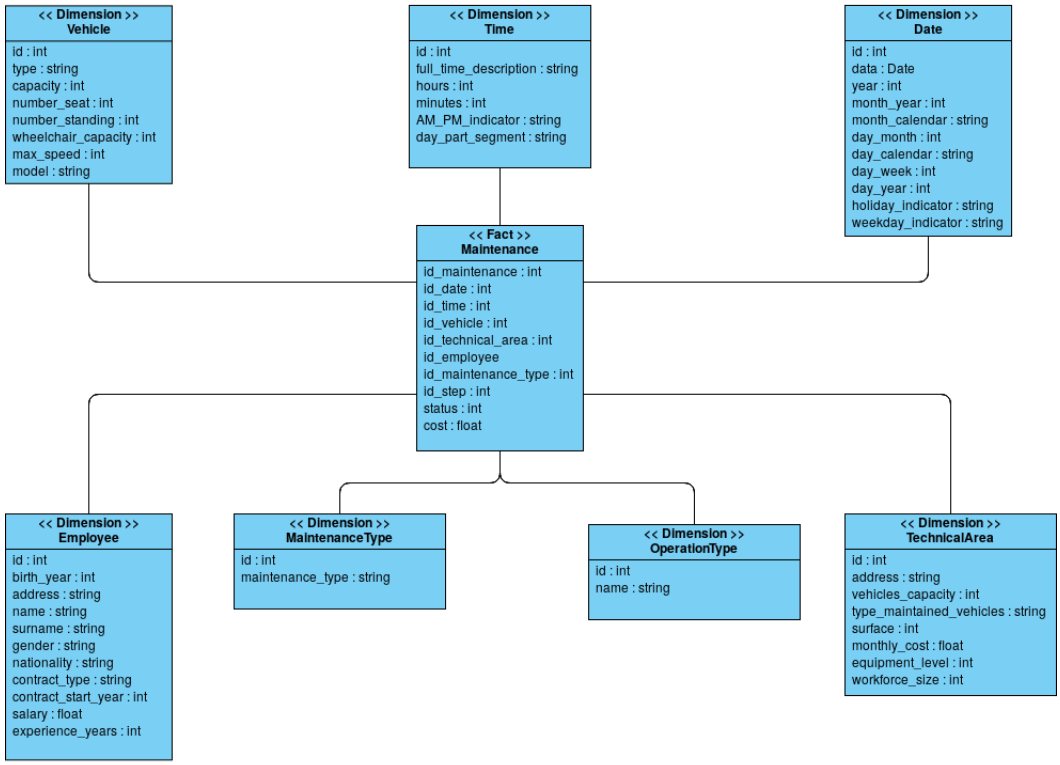


FIGURE 2.2 – modèle en étoile de l'action « maintenance »

— chaque tuple dans la table `MaintenanceType` désigne un type de maintenance effectuée (*e.g. changement de roues, maintenance du moteur, ...*)

— chaque tuple dans la table `OperationType` désigne une opération faite lors de la maintenance d'un véhicule (*i.e. type de transaction effectuée*). Ne connaissant pas les détails des opérations effectuées par *tam-voyages*, nous proposons les opérations suivantes :

1. conduire le véhicule concerné vers un dépôt de maintenance (*i.e. début de la maintenance*)
2. allouer un garage pour la maintenance du véhicule.
3. attribuer la maintenance du véhicule à un spécialiste.
4. éventuellement attendre l'acquisition de ressources nécessaires à la maintenance du véhicule.
5. mise en marche du véhicule dans le réseau après sa maintenance (*i.e. fin de la maintenance*)

Les mesures de la table `Maintenance` sont :

— `cost` : mesure additive désignant le coût de maintenance d'un véhicule pour une transaction donnée.

Remarques

l'attribut **equipment_level** de la table `TechnicalArea` désigne le niveau de matériaux disponibles au local et peut prendre une valeur entre 1 (*pas assez équipé*) et 5 (*très bien équipé*).

2.4 Entrepôt de données obtenu

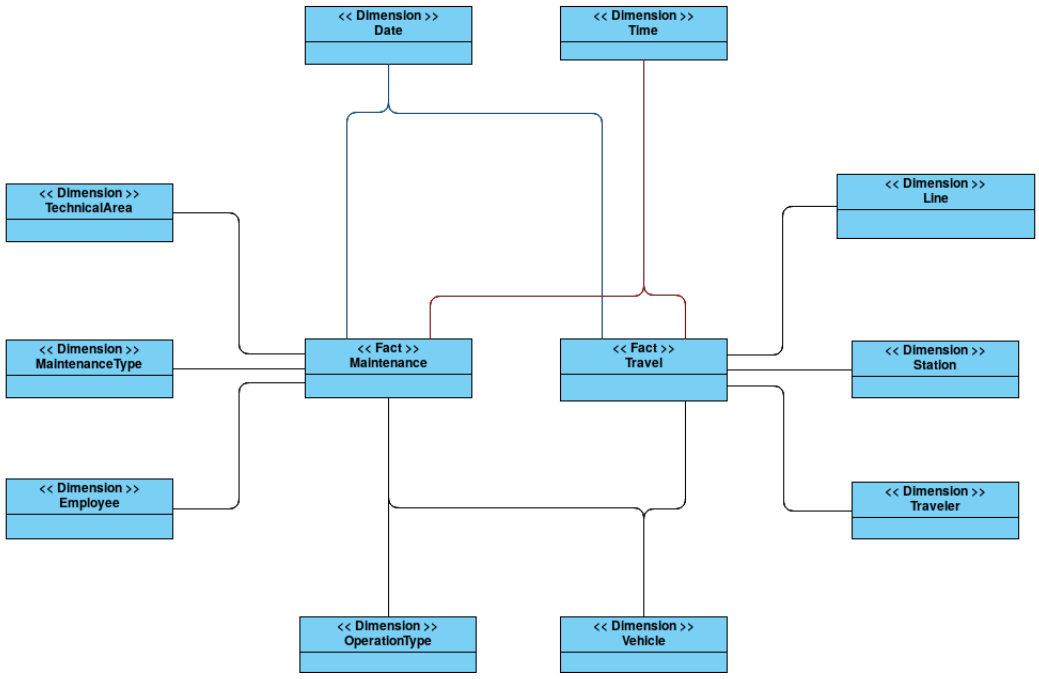


FIGURE 2.3 – le *data warehouse* résultant

2.4.1 Estimation de la taille de l'entrepôt

Dans notre entrepôt, on estime qu'on aura 365 lignes pour la table `Date_t` (*i.e. le nombre de jours d'une année*) et 1440 lignes pour la table `Time_t` (*i.e. le nombre d'heures et minutes par jour*).

En outre, les estimations faites pour les autres tables sont basées sur des informations récoltées depuis les liens suivants :

— https://fr.wikipedia.org/wiki/Autobus_de_Montpellier
— https://www.verif.com/bilans-gratuits/TRANSPORTS-AGGLOMERATION-DE-MONTPELL
— https://fr.wikipedia.org/wiki/Tramway_de_Montpellier
— https://e-metropolitain.fr/2016/12/17/une-visite-dans-les-coulisses-de-la-
— http://taminst.tsi.cityway.fr/presentation/

Table	Taille
Date_t	365
Time_t	1440
Vehicle	270
Line	116
Station	654
TechnicalArea	2
MaintenanceType	~ 500
Employee	1144

TABLE 2.1 – taille estimée de chaque table de l'entrepôt

Chapitre 3

Implémentation

3.1 Choix des technologies

Afin d'implémenter notre entrepôt de données, nous avons décidé d'utiliser `Oracle`. De plus, pour chaque dimension partagée entre les deux *data marts* nous avons créé deux vues virtuelles selon le niveau de détails nécessité pour l'analyse concernée.

3.2 Requêtes analytiques du *data-mart* des voyages

Pour répondre à la problématique du *data-mart* des voyages, nous avons effectué les requêtes analytiques suivantes [1] :

```
1  SELECT Travel.id_vehicle, COUNT(*) AS number_travel
2  FROM Travel
3  INNER JOIN Vehicle_travel ON Travel.id_vehicle =
       Vehicle_travel.id
4  INNER JOIN Traveler ON Travel.id_traveler = Traveler.id
5  INNER JOIN Date_travel ON Travel.id_date =
       Date_travel.id
6  WHERE Vehicle_travel.type = 'bus'
7  AND Traveler.anonymous = 1
8  AND Date_travel.month_year = 7
9  GROUP BY Travel.id_vehicle
10 ORDER BY Travel.id_vehicle;
```

Listing 3.1 – le nombre de voyages par bus, utilisant des tickets pour le mois de juillet

1. on affiche uniquement les premières lignes des résultats de chaque requête

ID_VEHICLE	NUMBER_TRAVEL
88	2
94	1
95	2
97	2
98	1
99	1
103	1
107	3
109	1
113	1
121	1

ID_VEHICLE	NUMBER_TRAVEL
124	1
125	1
131	2
132	1
133	1
139	1
141	1
143	1
146	1
147	1
148	1

FIGURE 3.1 – résultats de la requête 3.1

```
1  SELECT Line.num_line, COUNT(Traveler.id) AS
       number_travelers
2  FROM Travel
3  INNER JOIN Line
4    ON Travel.id_line = Line.id
5  INNER JOIN Traveler
6    ON Travel.id_traveler = Traveler.id
7  INNER JOIN Date_travel
8    ON Travel.id_date = Date_travel.id
9  WHERE Date_travel.year = EXTRACT(YEAR FROM SYSDATE)
10 AND Date_travel.month_year >= (EXTRACT(MONTH FROM
       SYSDATE) - 2)
11 AND Traveler.anonymous = 0
12 GROUP BY Line.num_line
13 ORDER BY Line.num_line;
```

Listing 3.2 – le nombre de voyageurs abonnés par ligne pour chaque voyage pour les deux derniers mois

NUM_LINE	NUMBER_TRAVELERS
1	38
2	55
3	92
4	60
6	3
7	10
8	4
9	8
10	5
11	4
12	5

NUM_LINE	NUMBER_TRAVELERS
13	8
14	6
15	9
17	10
18	16
19	4
20	11
21	6
22	5
23	7
24	6

FIGURE 3.2 – résultats de la requête 3.2

```
1  SELECT Line.num_line, Station.id, Station.name,
      COUNT(station.id) AS frequentation
2  FROM Travel
3  INNER JOIN Line
4        ON Travel.id_line = Line.id
5  INNER JOIN Station
6        ON Travel.id_station = Station.id
7  GROUP BY Line.num_line, Station.id, Station.name
8  HAVING COUNT(Station.id) = (
9    SELECT MAX(COUNT(*))
10       FROM Travel
11       INNER JOIN Line sl
12               ON Travel.id_line = sl.id
13       INNER JOIN Station
14               ON Travel.id_station = Station.id
15       WHERE sl.num_line = Line.num_line
16       GROUP BY Station.id)
17 ORDER BY Line.num_line, Station.id;
```

Listing 3.3 – l'arrêt le plus fréquenté par ligne

NUM_LINE	ID	NAME	FREQUENTATION
1	5	Port Marianne	15
2	49	Croix D'Argent	21
2	52	Victoire 2	21
3	66	Saint-Denis	44
4	8	Place de l'Europe	32
4	80	Saint-Martin	32
6	150	Route De Ganges	4
7	551	Gr??zes	7
8	229	Gare Saint-Roch (Pont De S??te)	6
9	424	Vieille Poste	5
9	453	Albert Einstein	5

NUM_LINE	ID	NAME	FREQUENTATION
10	126	La Pile	3
10	515	Clolus	3
10	537	Petit Bard	3
11	410	Cit?? Mion	7
12	229	Gare Saint-Roch (Pont De S??te)	7
12	565	L??on Cord??s	7
13	127	Campus Agropolis	5
13	420	Zoo	5
13	518	Hortus	5
13	570	Pic Saint-Loup	5
14	103	Palombes	6

FIGURE 3.3 – résultats de la requête 3.3

```
1  SELECT Line.num_line, Time_travel.hours AS Hour,
       COUNT(*) AS number_travel
2  FROM Travel
3  INNER JOIN Line ON Travel.id_line = Line.id
4  INNER JOIN Time_travel ON Travel.id_time =
       Time_travel.id
5  GROUP BY Line.num_line, Time_travel.hours
6  ORDER BY Line.num_line, Time_travel.hours;
```

Listing 3.4 – le nombre de voyages par heure pour chaque ligne

NUM_LINE	HOUR	NUMBER_TRAVEL
1	0	12
1	1	11
1	2	12
1	6	21
1	7	5
1	8	10
1	9	12
1	10	11
1	11	11
1	12	12
1	13	9

NUM_LINE	HOUR	NUMBER_TRAVEL
1	14	13
1	15	13
1	16	11
1	17	12
1	18	15
1	19	12
1	20	7
1	21	7
1	22	9
1	23	10
2	0	13

NUM_LINE	HOUR	NUMBER_TRAVEL
2	1	14
2	2	16
2	6	16
2	7	10
2	8	12
2	9	30

FIGURE 3.4 – résultats de la requête 3.4

```
1  SELECT Line.num_line, Vehicle_travel.id,
       COUNT(Vehicle_travel.id) AS number_vehicle
2  FROM Travel
3  INNER JOIN Line
4    ON Travel.id_line = Line.id
5  INNER JOIN Vehicle_travel
6    ON Travel.id_vehicle = Vehicle_travel.id
7  GROUP BY Line.num_line, Vehicle_travel.id
8  HAVING COUNT(Vehicle_travel.id) = (
9    SELECT MAX(COUNT(*))
10   FROM Travel
11   INNER JOIN Line sl
12     ON Travel.id_line = sl.id
13   INNER JOIN Vehicle_travel
14     ON Travel.id_vehicle = Vehicle_travel.id
```

```
15      WHERE sl.num_line = Line.num_line
16      GROUP BY Vehicle_travel.id)
17    ORDER BY Line.num_line, Vehicle_travel.id;
```

Listing 3.5 – le véhicule le plus utilisé par les voyageurs par ligne

NUM_LINE	ID	NUMBER_VEHICLE
1	18	8
2	38	9
3	10	13
3	14	13
4	3	11
4	37	11
6	89	2
6	97	2
6	121	2
6	123	2
6	261	2

NUM_LINE	ID	NUMBER_VEHICLE
7	137	3
7	223	3
8	163	2
8	221	2
9	167	2
9	179	2
9	202	2
9	231	2
9	247	2
9	270	2
10	107	3

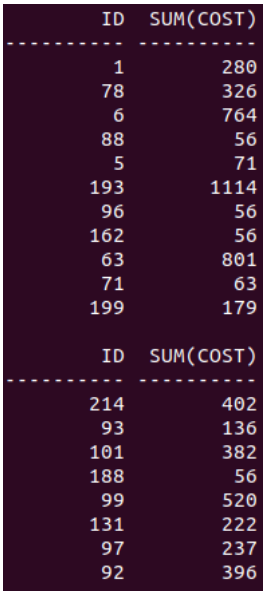
FIGURE 3.5 – résultats de la requête 3.5

3.3 Requêtes analytiques du *data-mart* des maintenances

Pour répondre à la problématique du *data-mart* des maintenances, nous avons effectué les requêtes analytiques suivantes :

```
1   SELECT Vehicle_maintenance.id, SUM(cost)
2   FROM Maintenance
3   INNER JOIN Vehicle_maintenance
4     ON Maintenance.id_vehicle = Vehicle_maintenance.id
5   GROUP BY Vehicle_maintenance.id;
```

Listing 3.6 – le coût total de maintenance de chaque véhicule



ID	SUM(COST)
1	280
78	326
6	764
88	56
5	71
193	1114
96	56
162	56
63	801
71	63
199	179

ID	SUM(COST)
214	402
93	136
101	382
188	56
99	520
131	222
97	237
92	396

FIGURE 3.6 – résultats de la requête 3.6

```
1  SELECT Employee.id AS Employee, COUNT(*) as
       intervention
2  FROM Maintenance
3  INNER JOIN Employee
4    ON Maintenance.id_employee = Employee.id
5  INNER JOIN Vehicle_maintenance
6    ON Maintenance.id_vehicle = Vehicle_maintenance.id
7  INNER JOIN Date_maintenance
8    ON Maintenance.id_date = Date_maintenance.id
9  WHERE Vehicle_maintenance.type = 'bus'
10 AND Date_maintenance.year = 2018
11 GROUP BY Employee.id;
```

Listing 3.7 – le nombre total de maintenances effectuées sur les bus par employé pour l'année 2018

EMPLOYEE	INTERVENTION
13	4
29	4
11	20
28	12
112	2
149	4
14	12
21	20
94	8
163	8
37	8

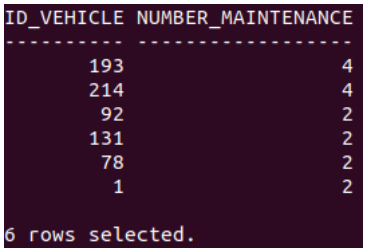
EMPLOYEE	INTERVENTION
135	18
23	8
17	4
165	4
45	8
136	2
7	20
115	2
18	16
27	8
16	10

EMPLOYEE	INTERVENTION
12	4
15	4

FIGURE 3.7 – résultats de la requête 3.7

```
1  SELECT Maintenance.id_vehicle, COUNT(*) AS
       number_maintenance
2  FROM Maintenance
3  INNER JOIN Vehicle_maintenance ON
       Maintenance.id_vehicle = Vehicle_maintenance.id
4  INNER JOIN Date_maintenance ON Maintenance.id_date =
       Date_maintenance.id
5  WHERE Date_maintenance.month_year >= (EXTRACT(MONTH
       FROM SYSDATE) - 6)
6  AND Maintenance.id_operation_type = 5
7  GROUP BY Maintenance.id_vehicle
8  ORDER BY number_maintenance DESC;
```

Listing 3.8 – le nombre total de maintenances effectuées par véhicule pour les 6 dernier mois.



ID_VEHICLE	NUMBER_MAINTENANCE
193	4
214	4
92	2
131	2
78	2
1	2

6 rows selected.

FIGURE 3.8 – résultats de la requête 3.8

```sql
SELECT MaintenanceType.maintenance_type AS type,
   COUNT(Maintenance.id_maintenance_type) AS totalCount
FROM Maintenance
INNER JOIN MaintenanceType
  ON  MaintenanceType.id =
     Maintenance.id_maintenance_type
GROUP BY MaintenanceType.maintenance_type
HAVING COUNT(Maintenance.id_maintenance_type) = (
  SELECT MAX(COUNT(*))
  FROM Maintenance
  GROUP BY Maintenance.id_maintenance_type);
```

Listing 3.9 – le type de maintenance le plus fréquent

TYPE	TOTALCOUNT
Battery Replacement	60

FIGURE 3.9 – résultats de la requête 3.9

```sql
SELECT TechnicalArea.id, Date_maintenance.month_year,
   COUNT(Maintenance.id_technical_area) AS
   numberMaintenance
FROM  Maintenance
INNER JOIN  TechnicalArea
  ON Maintenance.id_technical_area = TechnicalArea.id
INNER JOIN Date_maintenance
  ON Maintenance.id_date = Date_maintenance.id
GROUP BY TechnicalArea.id, Date_maintenance.month_year
ORDER BY TechnicalArea.id, Date_maintenance.month_year;
```

Listing 3.10 – le nombre de maintenances par mois pour chaque local technique

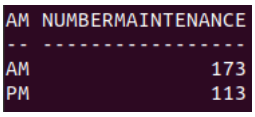
ID	MONTH_YEAR	NUMBERMAINTENANCE
1	1	60
1	2	32
1	3	20
1	9	50
2	1	20
2	2	44
2	3	10
2	4	20
2	5	10
2	9	20

10 rows selected.

FIGURE 3.10 – résultats de la requête 3.10

```sql
SELECT Time_maintenance.AM_PM_indicator,
   COUNT(Vehicle_maintenance.id) AS numberMaintenance
FROM Maintenance
INNER JOIN Vehicle_maintenance
  ON Maintenance.id_vehicle = Vehicle_maintenance.id
INNER JOIN Time_maintenance
  ON Maintenance.id_time = Time_maintenance.id
GROUP BY Time_maintenance.AM_PM_indicator;
```

Listing 3.11 – le nombre de véhicules maintenus le matin, et ceux maintenus l'après midi



AM	NUMBERMAINTENANCE
AM	173
PM	113

FIGURE 3.11 – résultats de la requête 3.11

Chapitre 4

Conclusion

4.1 Plan

Dans ce modèle nous avons proposé un outil permettant d'effectuer des analyses afin de repondre à des problématiques précises et bien définies. Pour ce faire, analyser chaque voyage s'est avéré essentiel. En effet, l'objectif principal de *tam-voyages* est de maximiser le nombre de voyages effectués dans son réseau et d'améliorer sa qualité de service. Il était donc logique de prioriser l'analyse des voyages effectués selon les voyageurs et titre de transport (ticket, abonnement) utilisés.

D'autre part, nous avons réalisé un outil répondant à des questions pertinentes à la problématique secondaire. Par conséquent, nous avons proposé d'analyser les différentes transactions effectuées lors des maintenances des véhicules.

En conclusion, notre modèle répond bien à un ensemble de questions primordiales par rapport aux objectifs de *tam-voyages*.

4.2 Perspectives

Bien que, notre modèle nous a permis de réaliser des analyses indispensables, nous avons constaté qu'il était assez limité. Effectivement, il est incapable de calculer le montant exact du chiffre d'affaires de *tam-voyages* en termes de ventes de tickets et de frais d'abonnements. Par ailleurs, on ne peut pas connaître la fréquentation de chaque trajet effectué par un véhicule ; on peut uniquement connaître les arrêts.

Par suite, des perspectives d'évolution de l'entrepôt sont possibles.

4.2.1 *Data Marts* pour la vente des tickets et les abonnements

Afin de pallier le probléme de la vente des tickets et les abonnements, nous proposons la perspective d'évolution suivante :

1. l'ajout d'un *Data mart* ayant comme action la vente d'un ticket à un voyageur à une date donnée avec une mesure désignant le prix du ticket vendu
2. l'ajout d'un *Data mart* ayant comme action l'abonnement d'un voyageur à une date donnée avec des mesures désignant les frais de l'abonnement et sa durée de validité

Ainsi, l'analyse des bénéfices recoltées à travers ces deux actions permet, entre autres, d'approximer le chiffre d'affaires de *tam-voyages*[1].

4.2.2 *Data Mart* pour les trajets effectués par des véhicules

Afin de pallier le problème de fréquentation de chaque trajet effectué par un véhicule, nous proposons la perspective suivante :

1. l'ajout d'un *Data Mart* tel que chaque fait soit désigné par un trajet effectué par un véhicule sur une ligne du réseau de transport à une date et une heure donnée, sans aucune mesure à expliciter.
2. l'idée dans cette vision consiste à modifier la table des voyages (développée auparavant) pour raffiner les analyses de manière à inclure des informations supplémentaires sur les trajets, non incluses dans la version initiale proposée.
3. un fait dans ladite table désignera ainsi un voyage effectué par un voyageur dans un véhicule faisant un trajet sur une ligne du réseau de transport à une date et une heure donnée.
4. pour ce faire, il faut lier la table des voyages avec la table des trajets (i.e. rajouter un attribut **id_trajet** dans la table `Travel`).

Ainsi, cette solution nous permettra de raffiner le grain d'analyse d'un voyage en incluant le trajet effectué par le véhicule concerné.

1. on ne compte pas les bénéfices récoltées via d'autres moyens tels que *VéloMagg* ou d'autres