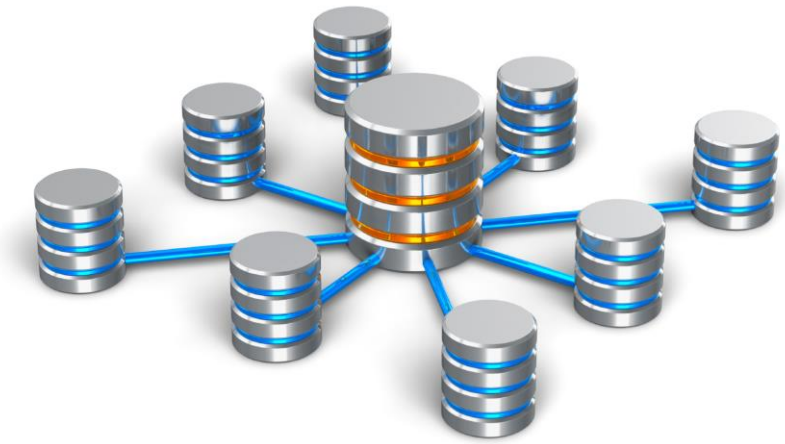
FACULTÉ DES SCIENCES DHAR EL MAHRAZ
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH

Université Sorbonne Paris Nord

Projet DATAWAREHOUSE



Année 2020/2021

Tables des figures

Table des matières

Problématique

Les entreprises et autres organisations utilisent la recherche marketing pour gérer les risques liés à l'offre de nouveaux produits et services. Ces organisations ne veulent pas dépenser trop d'argent pour développer une ligne de produits qui, selon les recherches, ne sera pas couronnée de succès. Une étude de marketing bien conçue et bien exécutée peut contribuer dans une large mesure à identifier les goûts des consommateurs et leurs préférences démographiques pour aider au lancement d'un produit. Cependant, plusieurs types de problèmes courants se posent avec les études de marché qui peuvent les rendre trop coûteuses et produire des résultats d'une valeur douteuse pour l'organisation.

L'informatique décisionnelle englobe une grande variété d'outils, d'applications et de méthodologies qui permettent aux organisations de collecter des données à partir de systèmes internes et de sources externes. Ces données sont ensuite préparées pour l'analyse afin de créer des rapports, tableaux de bord et autres outils de visualisation pour rendre les résultats analytiques disponibles aux décideurs et au personnel opérationnel.

Les entreprises génèrent des grandes quantités de données à propos de leurs actions quotidiennes, de ventes, d'achats et leurs transactions. Ce qui rend le traitement de ces quantités prend beaucoup de temps et d'efforts. De plus, les entreprises génèrent aussi des données massives caractérisant leurs informations qui peuvent être utilisé pour avoir des décisions afin d'améliorer la qualité de service qu'elles fournissent et aussi d'améliorer les chiffres d'affaires de ces derniers.

Tout au long de ce projet datawarehouse, on va présenter les étapes cruciales pour implémenter une Datawarehouse pour répondre à des questions économiques afin d'élever le volume de ventes pour une société commerciale.

- Les allés de magasin qui contiennent les produits les plus vendue ;
- Le temps de jour dans lequel les clients achètent un produit ;
- Le temps et le jour spécifique dans lesquels des produits sont vendu ;
- Etc…

Description des données sélectionnées

Pour ce projet on a opté à utiliser la base des données 'Instacart Market Basket Analysis' disponible sur le site web « Kaggle ». Cet ensemble de données anonymisées contient un échantillon de plus de 3 millions de commandes d'épicerie provenant de plus de 200 000 utilisateurs d'Instacart. Pour chaque utilisateur, l'ensemble des données fournissent entre 4 et 100 de ses commandes, avec la séquence des produits achetés dans chaque commande. Ils fournissent également la semaine et l'heure du jour où la commande a été passée, ainsi qu'une mesure relative du temps entre les commandes.

Cette dataset contient huit fichiers .csv, on a travaillé seulement avec cinq fichiers.

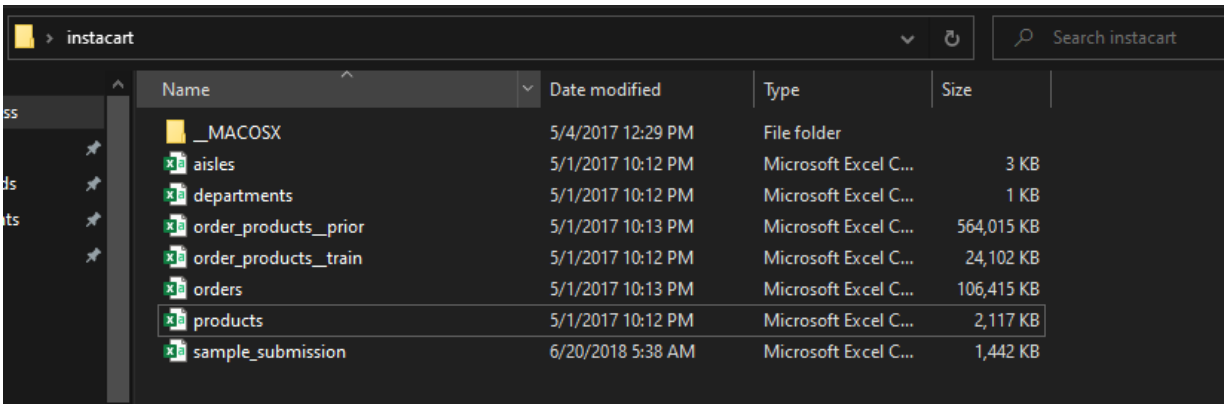


Figure 1: L'ensemble des fichiers de Dataset

Le fichier 'aisles.csv' contient les différentes allées, avec leurs unique id, que le magasin possède.

aisles.csv

```
aisle_id,aisle
1,prepared soups salads
2,specialty cheeses
3,energy granola bars
...
```

Figure 2: Description de fichier 'aisles.csv'

Le fichier 'departements.csv' fournit les noms des départements que le magasin possède.

departments.csv

```
department_id,department
1,frozen
2,other
3,bakery
...
```

Figure 3: Description de fichier 'departements.csv'

Le fichier 'orders.csv' contient des informations à propos des commandes, ce qui nous intéresse c'est le produit commandé et le temps il a été acheté.

orders.csv

This file tells to which set (prior, train, test) an order belongs. You are predicting reordered items only for the test set orders. 'order_dow' is the day of week.

```
order_id,user_id,eval_set,order_number,order_dow,order_hour_of_day,days_since_prior_order
2539329,1,prior,1,2,08,
2398795,1,prior,2,3,07,15.0
473747,1,prior,3,3,12,21.0
...
```

Figure 4: Description de fichier 'orders.csv'

Le fichier ‘products.csv’ contient l’enregistrement de tous les produits de magasin.

products.csv

```
product_id,product_name,aisle_id,department_id
1,Chocolate Sandwich Cookies,61,19
2,All-Seasons Salt,104,13
3,Robust Golden Unsweetened Oolong Tea,94,7
...
```

Figure 5: Description de fichier 'products.csv'

order_products__*.csv

```
order_id,product_id,add_to_cart_order,reordered
1,49302,1,1
1,11109,2,1
1,10246,3,0
...
```

Figure 6: Description de fichier 'order_products__.csv'

Ce fichier précise quels produits ont été achetés dans chaque commande. order_products__prior.csv contient le contenu des commandes précédentes pour tous les clients. La colonne "reordered" indique que le client a déjà passé une commande contenant le produit. Notez que certaines commandes ne contiennent pas d'articles commandés à nouveau.

Modèle E/A normalisé

Pour le modèle entité-association on a opté le schéma suivant :

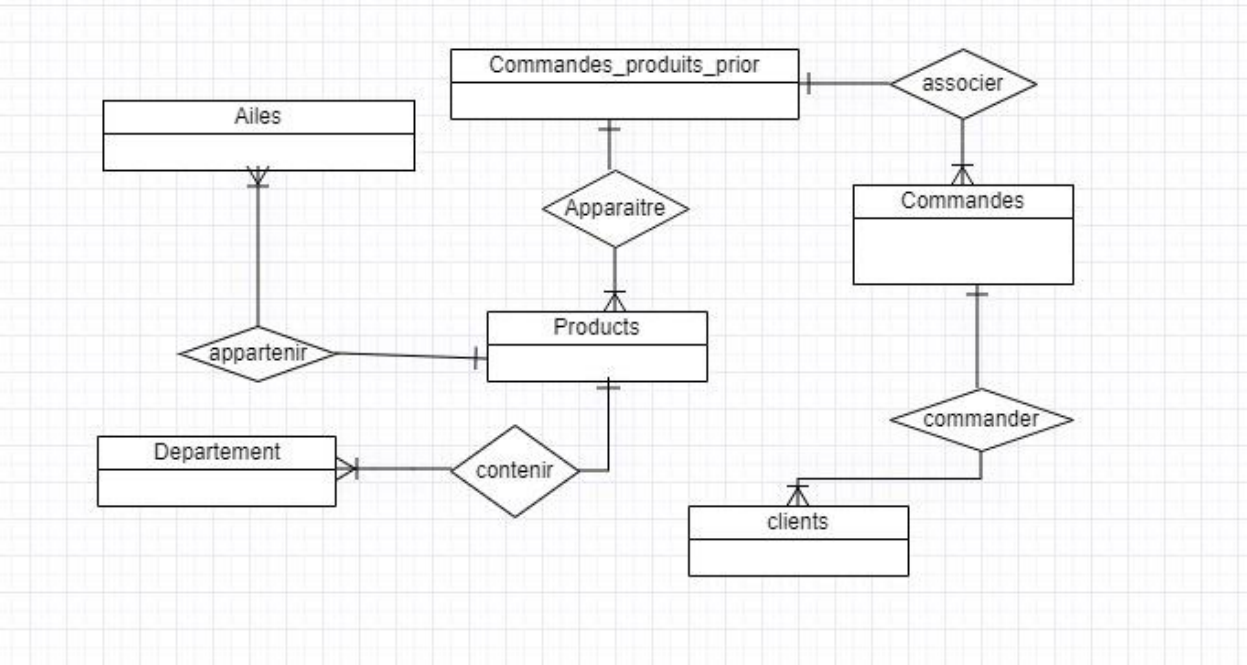


Figure 7: Le modèle entité-association

Dans le schéma on voit que les produits appartiennent à des allées et des départements bien définies. La table 'commandes_produits_prior' définit une jointure entre la table commandes et la table produits, de telle sorte qu'elle montre qu'il produit a été commandé ou dans quelle commande un produit a été commandé.

Modèle dimensionnel

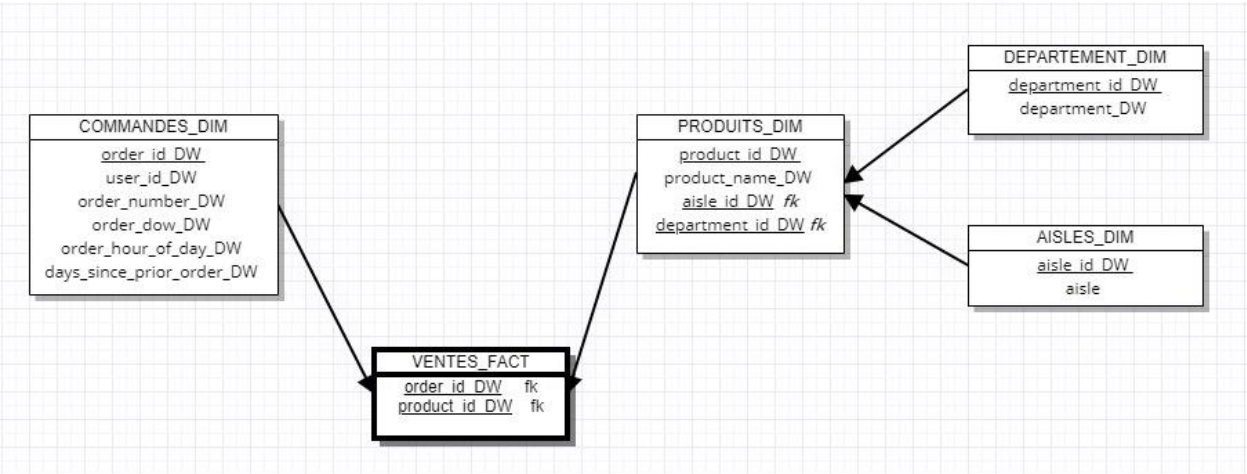


Figure 8: Le modèle dimensionnel

Après l'étude des données et les besoins devant ce projet, on a opté à travailler avec le schéma au-dessus comme modèle dimensionnel. Comme la table 'commandes_produit_prior_' est la plus volumineuse de l'ensemble des données, et que cette table représente les mesures qu'on veut travailler avec ainsi qu'elle expresse la relation 'many-to-many' avec les autres tables, on la met comme table de fait. En effet, cette table représente la jointure entre la table commande et la table produits, donc c'est beaucoup plus pratique de la mettre comme une table de fait.

Comme des tables dimensions, on a mis les tables commandes, produits, départements et allées.

Création d'ETL

Dans la phase d'ETL on a copié des données depuis les tables des systèmes transactionnels vers les tables du modèle en étoile du data warehouse.

La création des tables, Products par copie des données depuis le fichier 'products.csv', Commandes par copie des données depuis le fichier 'orders.csv', Aisles par copie des données depuis le fichier 'aisles.csv', Departements par copie des données depuis le fichier 'departements.csv' et la table 'order_products_prior_' par copie des données depuis le fichier 'order_products__prior_.csv'.

D'une part, pour la zone de transformation on a créé une table pour chaque table de la zone d'extraction. On a déclaré une méthode pour chaque attribut que l'on souhaite exporter dans le modèle dimensionnel. Cette méthode permettra de réaliser dynamiquement les transformations et vérifications adéquates.

```
CREATE or replace  type T_Commandes as object (

order_id INTEGER,
user_id INTEGER,
eval_set varchar2(10),
order_number INTEGER,
order_dow INTEGER,
order_hour_of_day INTEGER,
days_since_prior_order INTEGER,

MEMBER FUNCTION fun_order_id RETURN INTEGER,
MEMBER FUNCTION fun_user_id RETURN INTEGER,
MEMBER FUNCTION fun_eval_set RETURN varchar2,
MEMBER FUNCTION fun_order_number RETURN INTEGER,
MEMBER FUNCTION fun_order_dow RETURN INTEGER,
MEMBER FUNCTION fun_order_hour_of_day RETURN INTEGER,
MEMBER FUNCTION fun_days_since_prior_order RETURN INTEGER

);
```

Figure 9: Exemple de la phase de transformation de la table Commande

D'une autre part, on a aussi ajouté les contraintes dans les tables quand ils sont nécessaires.

```
ALTER TABLE BDT_Commandes ADD CONSTRAINT pk_order_id  PRIMARY KEY (order_id);

ALTER TABLE BDT_Commandes DISABLE CONSTRAINT pk_order_id;
--------------------------------------------------------------------------------
ALTER TABLE BDT_Commandes ENABLE CONSTRAINT pk_order_id;
```

Figure 10: Exemple d'ajout des contraintes pour la phase Transformation

On a aussi vérifié que la réactivation des contraintes n'a pas rejeté de données (100% des données sont passées de la BDE à la BDT).

```
INSERT INTO BDT_Commandes SELECT * FROM  Commandes where eval_set='prior'
and order_id in (select order_id from order_products__prior where
product_id in (select product_id from Products));

select * from BDT_Commandes b ;
```

Figure 11: Vérification que la réactivation des contraintes n'a pas rejetée de données.

Après la phase de transformation, la zone de chargement reçoit une copie des données transformées pour implémenter la phase de DataWarehouse. On a utilisé la BD en flocon pour implémenter la DataWareHouse.

Dans cette étape, on a créé les tables de schéma flocon. On ajoute les contraintes. Puis on les désactive pour charger chaque dimension du data warehouse via l'API de la zone T et charger la table des faits du data warehouse via l'API de la zone T. Et on active les contraintes pour vérifier si tout est passé bien.

```
CREATE table PRODUITS_DIM (

product_id_DW INTEGER,
product_name_DW varchar2(50),
aisle_id_DW INTEGER,
department_id_DW INTEGER

);
alter table PRODUITS_DIM add constraint pk_product_id_DW primary key(product_id_DW);

alter table PRODUITS_DIM add constraint fk_aisle_id_DW foreign key(aisle_id_DW)
REFERENCES AISLES_DIM(aisle_id_DW);

alter table PRODUITS_DIM add constraint fk_department_id_DW foreign key(department_id_DW)
REFERENCES DEPARTEMENT_DIM(department_id_DW);

alter table PRODUITS_DIM disable constraint pk_product_id_DW;
alter table PRODUITS_DIM disable constraint fk_aisle_id_DW;
alter table PRODUITS_DIM disable constraint fk_department_id_DW;
```

Figure 12: Exemple de la phase de dataWareHouse

Exploration des données et réponses aux questions posées

Dans la problématique, on a posé des questions afin de les fournir des réponses après avoir implémenter la DatawareHouse.

Pour la première question, les allés de magasin qui contiennent les produits les plus vendus, on peut par exemple chercher les dix allés de magasin qui contiennent les produits les plus vendus. La requête est comme suivante :

```
-- Les 10 allées de magasin qui contiennent les produits les plus vendus ;
select aisle from AISLES_DIM
where aisle_id_DW IN ( select aisle_id_DW from PRODUITS_DIM where product_id_DW IN (
    SELECT  *
    FROM    (
            SELECT  PRODUCT_ID_DW
            from    VENTES_FACT
            group by PRODUCT_ID_DW
            order by count(*) desc
            )
    WHERE   rownum <= 10
));
```

Figure 13: La requête pour les dix allés de magasin qui contiennent les produits les plus vendus

Et comme résultat de la requête on trouve :

	AISLE
1	preserved dips spreads
2	prepared meals
3	condiments
4	nuts seeds dried fruit
5	trail mix snack mix
6	cat food care
7	cleaning products
8	candy chocolate
9	hot cereal pancake mixes
10	salad dressing toppings

Figure 14: Les dix allés de magasin qui contiennent les produits les plus vendus

A propos de la deuxième question, on peut chercher à savoir le temps de jour dans lequel les clients achètent un produit. On peut prendre comme produit le 'Petit Suisse Fruit'. La requête est comme suivante :

```
select order_hour_of_day_DW, count(order_hour_of_day_DW) from COMMANDES_DIM where order_id_DW IN (
    select order_id_DW from
    VENTES_FACT where product_id_DW IN (select product_id_DW from PRODUITS_DIM where product_name_DW LIKE '%All-Seasons Salt%')
    )
    GROUP BY order_hour_of_day_DW
HAVING count(order_hour_of_day_DW) IN ( select  MAX(count(order_hour_of_day_DW)) from COMMANDES_DIM where order_id_DW IN (
    select order_id_DW from
    VENTES_FACT where product_id_DW IN (select product_id_DW from PRODUITS_DIM where product_name_DW LIKE '%All-Seasons Salt%')
    )
    GROUP BY order_hour_of_day_DW )
;
```

Figure 15: La requête de temps de jour dans lequel les clients achètent un produit

Et comme résultat de la requête on trouve que ce produit est vendu envers 9h de matin.

Ce qui concerne la troisième question, pour trouver les dix produits les plus vendus en Vendredi de 8H à 12H, une requête possible est :

```
SELECT product_name_DW, product_id_DW FROM PRODUITS_DIM JOIN ( SELECT product_id_DW, count(product_id_DW) FROM VENTES_FACT WHERE order_id_DW IN(
    SELECT order_id_DW from COMMANDES_DIM where (order_hour_of_day_DW BETWEEN 8 AND 12) AND (order_dow_DW = 5)
)
GROUP BY product_id_DW
ORDER BY count(product_id_DW) DESC) USING (product_id_DW) WHERE rownum <= 10;
```

Figure 16: : La requête des dix produits les plus vendus en Vendredi de 8H à 12H

Et cette requête a comme résultat :

	PRODUCT_NAME_DW	PRODUCT_ID_DW
1	Butter Flavor Pan Popcorn	771
2	Light and Fluffy Blueberry Pancake Mix	565
3	Super Enzymes	390
4	Whole Wheat 100% Stone Ground Bread	1760
5	Iced Tea with Lemon Flavor	1581
6	White Chicken Chili with Beans Pouch	570
7	Classics Earl Grey Tea	87
8	Calcium 600mg + D3 Tablets	1966
9	Classic Original Hummus	544
10	Chile Con Queso Potato Crisps	851

Figure 17: Les dix produits les plus vendus en Vendredi de 8H à 12H

Optimisation des requêtes

Dans cette partie, on va implémenter l'indexation. Si on veut trouver une information qui se trouve dans une grande base de données. Pour extraire cette information de la base de données, l'ordinateur va parcourir chaque ligne jusqu'à ce qu'il la trouve. Si les données que vous recherchez se trouvent vers la fin, cette requête prendrait beaucoup de temps à exécuter.

L'indexation rend les colonnes plus rapides à interroger en créant des pointeurs vers l'endroit où les données sont stockées dans une base de données.

Dans un premier temps, on trouve que la première requête a cette table comme des ressources nécessaires pour exécuter la requête.

```
PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                |     1 |    32 |    21  (24)| 00:00:01 |
|   1 |  NESTED LOOPS                 |                |       |       |            |          |
|   2 |   NESTED LOOPS                |                |     1 |    32 |    21  (24)| 00:00:01 |
|   3 |    VIEW                       | VW_NSO_2       |    10 |   130 |    19  (22)| 00:00:01 |
|   4 |     HASH UNIQUE               |                |     1 |   390 |            |          |
|*  5 |      HASH JOIN                |                |    10 |   390 |    19  (22)| 00:00:01 |
|   6 |       VIEW                    | VW_NSO_1       |    10 |   130 |    13  (24)| 00:00:01 |
|*  7 |        COUNT STOPKEY          |                |       |       |            |          |
|   8 |         VIEW                  |                | 16512 |   209K|    13  (24)| 00:00:01 |
|*  9 |          SORT ORDER BY STOPKEY|                | 16512 |   209K|    13  (24)| 00:00:01 |
|  10 |           HASH GROUP BY       |                | 16512 |   209K|    13  (24)| 00:00:01 |
|  11 |            TABLE ACCESS FULL  | VENTES_FACT    | 16512 |   209K|    10   (0)| 00:00:01 |
|  12 |       TABLE ACCESS FULL       | PRODUITS_DIM   |  1782 | 46332 |     5   (0)| 00:00:01 |
|* 13 |    INDEX UNIQUE SCAN          | PK_AISLE_ID_DW |     1 |       |     0   (0)| 00:00:01 |
|  14 |   TABLE ACCESS BY INDEX ROWID | AISLES_DIM     |     1 |    19 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------
```

Figure 18: Les traces de consommation lors d'exécution d'une requête

Après avoir effectué l'indexation des attributs utilise par la requête :

```
EXPLAIN PLAN FOR select aisle from AISLES_DIM
where aisle_id_DW IN ( select aisle_id_DW from PRODUITS_DIM where product_id_DW IN (
    SELECT  *
    FROM    (
            SELECT  PRODUCT_ID_DW
            from    VENTES_FACT
            group by PRODUCT_ID_DW
            order by count(*) desc
            )
    WHERE   rownum <= 10
));
select * from table(dbms_xplan.display);

CREATE INDEX index_ventes ON VENTES_FACT(order_id_DW);
CREATE INDEX index_products ON PRODUITS_DIM(product_id_DW);
--CREATE INDEX index_aisles ON AISLES_DIM(aisle_id_DW);
```

Figure 19: Indexation des clés primaires

On voit que l'indexation n'a pas d'effet sur la performance d'exécution des requêtes. Cela peut s'expliquer par le fait que la table fait de modèle dimensionnel a des clés primaires qui sont des clés étrangères déjà indexé. Et de plus, la requête ne contient pas des jointures qui peuvent compliquer l'exécution des requêtes.