RENDU TP3
Entrepôts de données et Big Data

Chamy Kaci et Asma Guermouche

October 2022

Table des matières

1 Partie 1

1.1 Les interrogations : requêtes transactionnelles vs analytiques

1. Restent-ils des billets à Montpellier pour la séance de 20 heures du film "Logan" ?
   — Requête transactionnelle : en effet on interroge notre bdd pour obtenir une réponse binaire OUI/NON. Un client de ce cinéma peut très probablement poser cette question dans l'objectif de passer une transaction.
2. Aurait-on éventuellement pu proposer des plus grandes salles et plus de séances pour le dernier film de Star Wars ?
   — Requête analytique : en effet on cherche à faire de l'optimisation et de la stratégie en utilisant des données déjà existantes. Enfin, le client ne peut pas être a l'origine de cette requête.
3.

```
SELECT Film.titre, Cinema.nom, Date.mois,COUNT(Place.placeID)
FROM Film, Ventes, Cinema, Place, Temps, Date
WHERE Ventes.filmID = Film.filmID AND Ventes.cinemaID =
    Cinema.cinemaID AND Ventes.tempsID = Temps.tempsID AND
    Place.cinemaID = Cinema.cinemaID AND Ventes.dateID = Date
    .dateID
GROUP BY Film.titre, Cinema.nom, Date.mois
```

Listing 1 – Requête

   — Requête analytique : en effet on cherche à comparer le nombre de place par film par cinéma et par mois, chose qu'un client ou un utilisateur n'est pas voué à connaître.
4.

```
SELECT Temps.crenau, COUNT(*)
FROM Ventes, Temps
WHERE Ventes.tempsID = Temps.tempsID
GROUP BY Temps.creneau
```

Listing 2 – Requête

   — Requête analytique : en effet on cherche à comparer le nombre de ventes pour chaque créneau. Une utilisation possible de cette information serait de connaître s'il est nécessaire d'augmenter les effectifs en caisse pour potentiellement augmenter le nombre de ventes.
5.

```
INSERT INTO Ventes
VALUES ('film1','cinema24','date2','temps3','place44','7.50'
       )
```

Listing 3 – Requête

   — Requête transactionnelle : en effet on effectue une insertion qui correspond simplement à la vente d'un ticket de cinéma.

1.2 Un entrepôt de données pour Amazon

1. Le modèle en étoile.

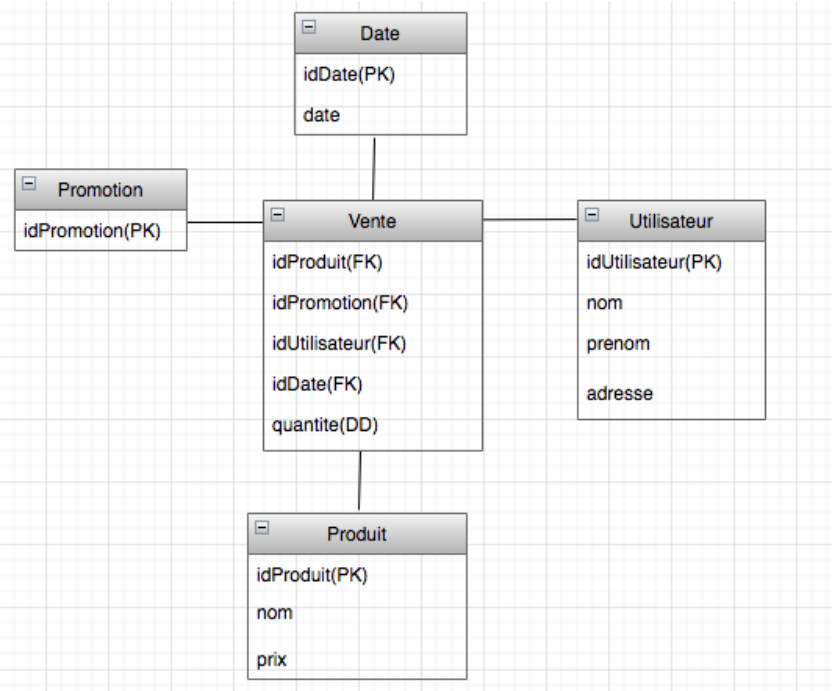

FIGURE 1 – Modèle en étoile

2. L'intégration de commentaires :

   Les commentaires, (qui peuvent également inclure les évaluations produit sur une échelle), se- raient l'objet d'une autre table de faits. Cette table pourrait partager des dimensions avec d'autres tables de faits. Pour garder la métaphore du modèle en étoile, on parle alors de constellation.

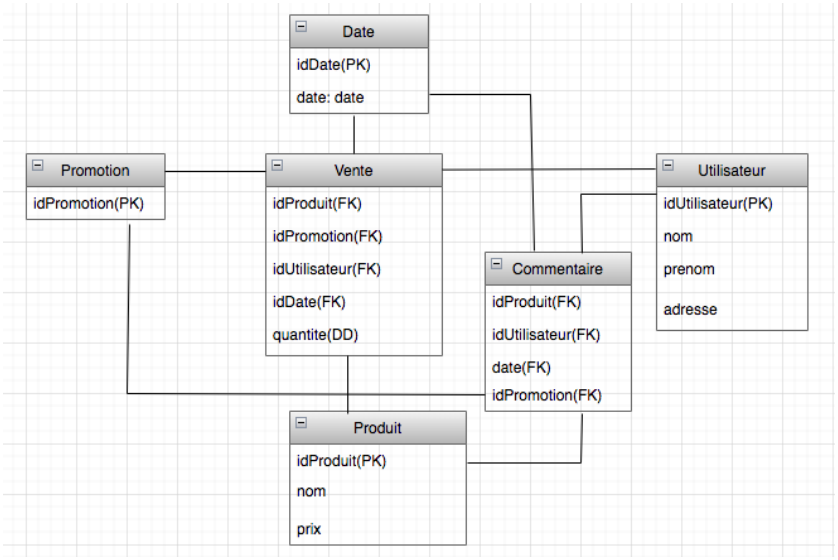

FIGURE 2 – Modèle en étoile 2

3. Proposez trois requêtes analytiques pour le modèle de données en étoile que vous avez conçu.

```
SELECT promotion, sum(quantity), sum(amount) FROM Sales JOIN
    Date on date =Date.id
WHERE date.mois = "decembre"
GROUP BY promotion;
```

Listing 4 – Requête

```
SELECT promotion, avg(amount), avg(quantity) from Sales JOIN
    Date on date =Date.id
WHERE date.mois = "janvier";
```

Listing 5 – Requête

```
SELECT product, sum(quantity)
FROM Sales
GROUP BY product
HAVING sum(quantity) >=
ALL(SELECT sum(quantity)
FROM Sales
GROUP BY product);
```

Listing 6 – Requête

1.3 Requêtes Analytiques

Considérez la table de faits (simplifiée) suivante, enregistrant les ventes journalières chez Monoprix (attention : un script de création et remplissage de la base est à disposition dans la page Moodle du cours).

```sql
CREATE TABLE ventes_monoprix (
id_date VARCHAR(10) NOT NULL,
id_produit VARCHAR(10) NOT NULL,
id_magasin VARCHAR(10) NOT NULL,
id_ville VARCHAR(10) NOT NULL,
montant_journalier NUMBER(10,2) NOT NULL
);
```

Listing 7 – Requête

Exprimez en SQL les interrogations suivantes, à l'aide de l'opérateur GROUP-BY.

1. Donner le montant total des ventes par produit

```sql
select sum(montant_journalier) from ventes_monoprix
group by id_produit;
```

Listing 8 – Requête

2. Donner le montant total des ventes par produit et par ville

```sql
select sum(montant_journalier) from ventes_monoprix
group by (id_produit,id_ville);
```

Listing 9 – Requête

3. Donner le montant total des ventes par produit et par jour

```sql
select sum(montant_journalier) from ventes_monoprix
group by (id_produit, id_date);
```

Listing 10 – Requête

4. Donner la moyenne du montant des ventes par magasin et par jour

```sql
select avg(montant_journalier) from ventes_monoprix
group by (id_produit, id_date);
```

Listing 11 – Requête

5. Donner le montant total des ventes par ville par jour

```sql
select sum(montant_journalier) from ventes_monoprix
group by (id_ville, id_date);
```

Listing 12 – Requête

6. Donner le montant total des ventes par produit, ville et jour

```sql
select sum(montant_journalier) from ventes_monoprix
group by (id_produit,id_ville,id_date);
```

Listing 13 – Requête

Enfin, testez les options ROLLUP et CUBE et comparer les résultats. Pourrait-on regrouper les interrogations grâce à ces options ?

Les opérateurs ROLLUP et CUBE de Oracle calculent des lignes qui vont au-delà de l'objectif des requêtes précédentes prises indépendamment. Prenons comme cas d'exemple la requête N°1 et voyons comment l'utilisation du Rollup ajoute de l'information à notre questionnement de base.

En effet on peut regrouper certaine requêtes ensemble et obtenir les mêmes résultats avec moins de requêtes.

```
select sum(montant_journalier) from ventes_monoprix
group by cube (id_produit,id_ville,id_date);
```

Listing 14 – Requête

Cette requête effectuera un GROUP BY sur toutes les permutations de n'importe quel nombre de colonnes parmi id_produit, id_ville, id_date. Elle nous donne donc les résultats des requêtes 1,2,3,5 et 6.

2 Partie 2

2.1 Classification des faits

1. fait transactionnel, Dans ce fait On s'intéresse au prix total d'une vente d'un client, donc on a besoin du détails.
   La mesure x est additive, en effet elle peut-être calculer avec ces semblables.
2. fait snapshot, Ici on s'intérèsse au chiffre d'affaires d'un magasin donc on a pas besoin d'avoir des détails sur la vente des produits.
   Additive, on peut faire la somme totale des chiffres.
3. fait snapshot, On s'intéresse qu'au stock du produit au magasin pour le jour j, les autres informations n'ont pas d'importance dans le calcul du stock.
   Semi-additive, car on ne pourrait pas avoir le nombre de produits exacte correspondant à une journée j, par contre on peut faire la moyenne.
4. fait snapshot, ici on ne s'intéresse qu'au nombre de vente de produits cummulés depuis le début de l'année, les autres détails n'ont pas un role important dans ce cas.
   Semi-additive, car on ne peut pas avoir le nombre de produits exacte correspondant à une journée j, par contre on pourrait calculer la moyenne.
5. fait transactionnel, On a besoin d'avoir le détail de l'appel (un appel correspond à quel client, et est traité par quel employé ?).
   Non additive, Il n'y a aucune mesure qui nous permet de faire la somme.
6. fait transactionnel, Dans ce cas une note est attribué par un client pour l'achat d'un produit, donc ces détails sont importants pour notre cas.
   Semi-additive, On peut faire la moyenne des notes attribuées aux produits.

7. fait transactionnel, Ici on s'intéresse à la durée d'un appel en secondes effectué par un client, le jour j qui est traité par un employé e, ces détails sont importants dans ce cas.
   Additive, On a la possibilié d'additionner les durées d'appels .
8. fait snapshot, Dans ce fait ce qui nous intéresse c'est le montant total de la monnaie changés en euros, les autres détails n'ont pas d'importance.
   Additive, On a la possibilité d'additionner tous les montants totaux de la monnaie changée en euros
9. fait snapshot, Dans ce fait on s'intéresse au cours de change moyen de m eu euro pour toutes les transactions du jour j, donc dans ce fait non plus, les détails n'ont pas d'importances.
   Non additive