EDBD - TP3

Dorian Rey & Elliot Durand

1) Les interrogations : requêtes transactionnelles vs analytiques

1.1 Restent-ils des billets à Montpellier pour la séance de 20 heures du film Logan ?

Transactionnel : On cherche une information dans la base de données assez précise, ici dans une ville, un film et une heure particulière

1.2 Aurait-on éventuellement pu proposer des plus grandes salles et plus de séances pour le dernier film de Star Wars ?

Analytique : On cherche à faire une analyse pour savoir si on aurait du faire une certaine action, ici augmenter le nombre de place et de séance pour Star Wars

1.3 (Traduire la requête suivante en langage naturel, puis indiquer s'il s'agit d'une requête transactionnelle ou analytique)

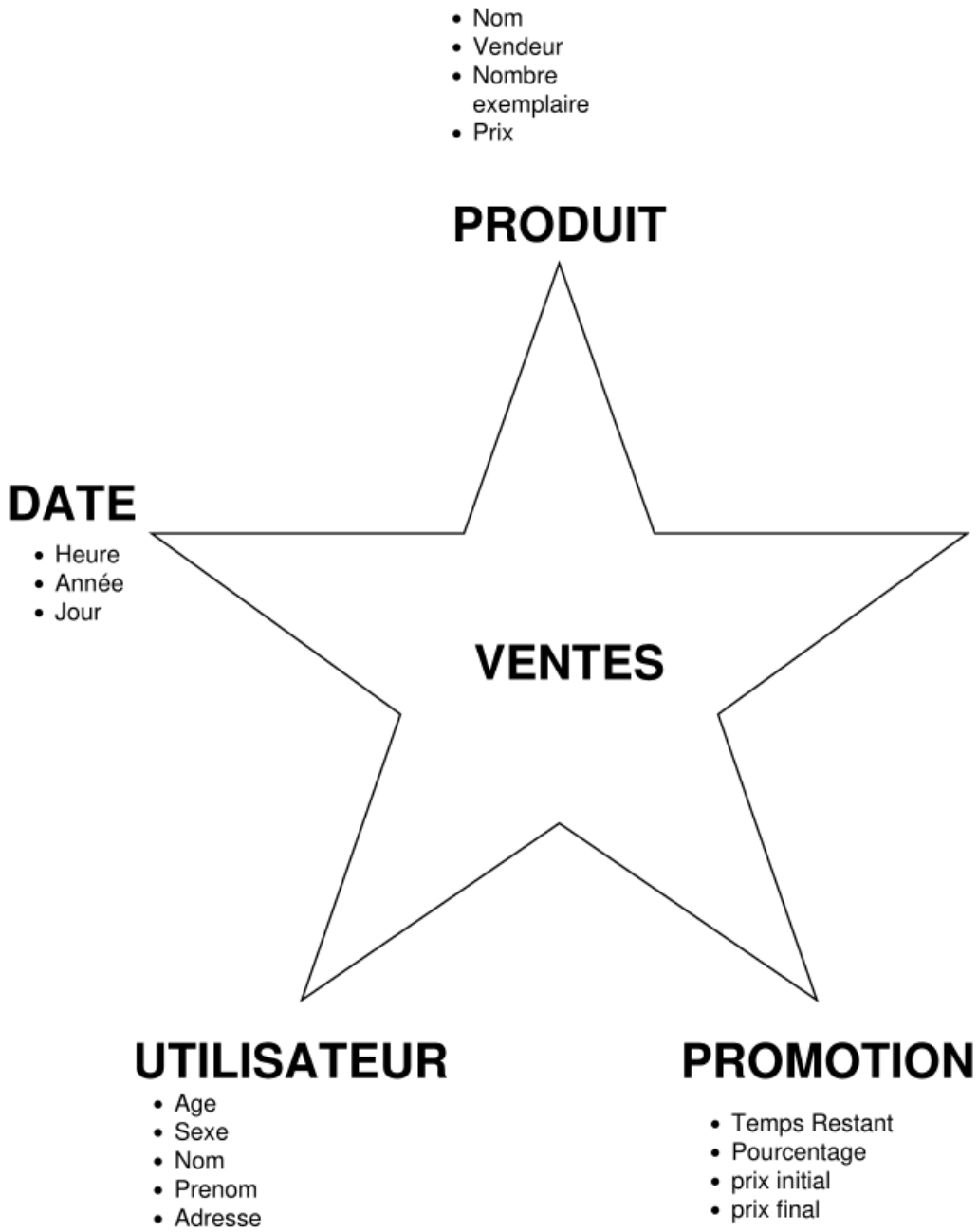
1. Le nombre de spectateurs par titre de film, par nom de cinéma et par mois. C'est une requête analytique car on peut faire des statistiques sur le la fréquentation des différents films en fonction des cinémas (par exemples des régions) et des périodes (par exemple Noël etc)

2. La fréquentation des cinémas en fonction du temps. C'est une requête analytique également car on peut analyser quand les gens vont le plus au cinéma

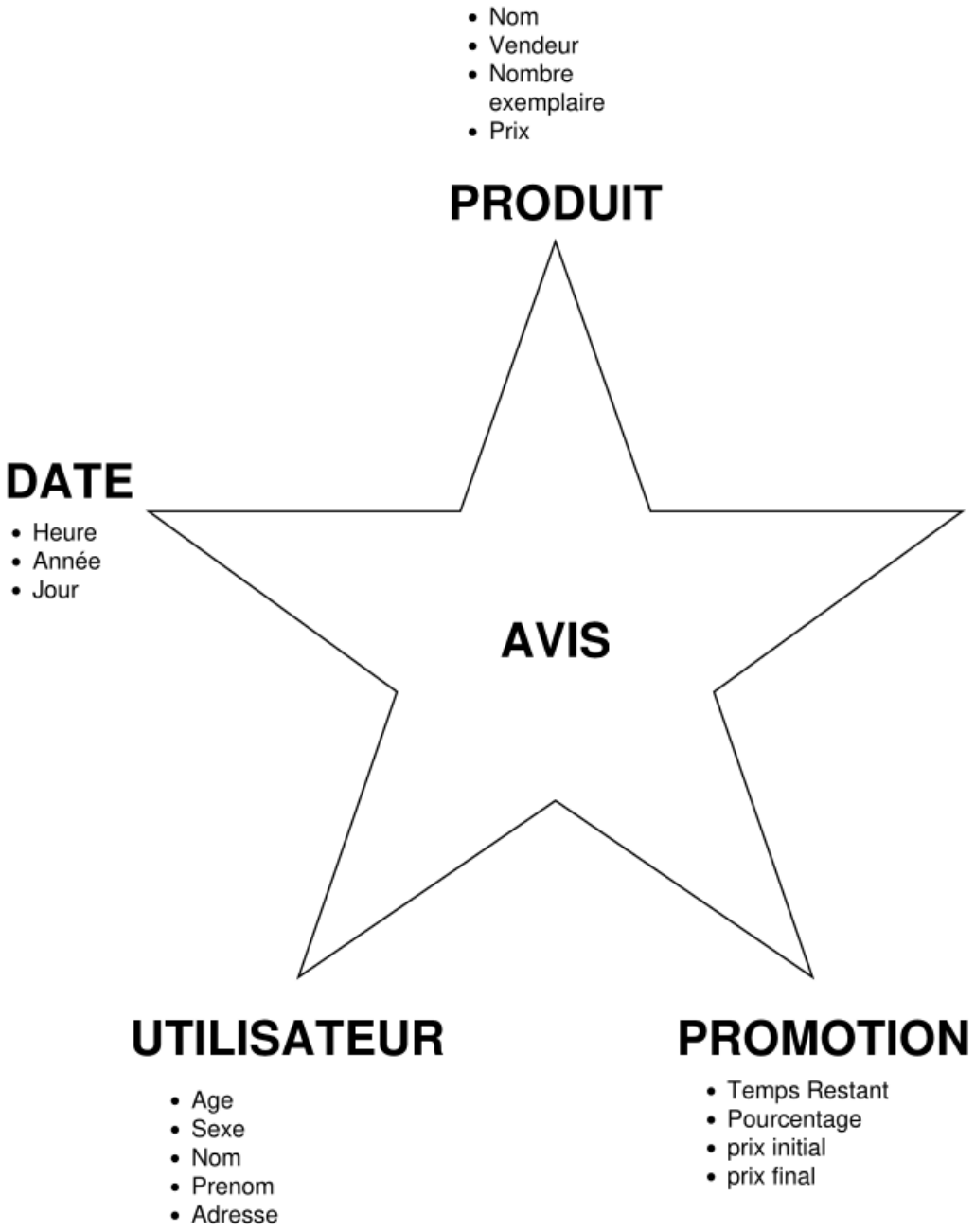
3. Insertion d'une vente de place. C'est une requête transactionnel car les insert to sont plus propice aux base de données relationnels par rapport aux entrepôts de données

2) Un entrepôt de données pour Amazon

2.1 Proposez un modèle en étoile pour un entrepôt de données permettant l'analyse des ventes dans Amazon



2.2 Comment feriez-vous pour intégrer les commentaires clients dans ce modèle ?



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

- Produits les plus vendus par mois
- Nombre de ventes par mois d'un produit
- Produit le plus vendu par région

3) Requêtes Analytiques Monoprix

3.1. Donner le montant total des ventes par produit.

```
SELECT id_produit, SUM(montant_journalier)
FROM ventes_monoprix GROUP BY id_produit;
-- Exemple : Z = 19 796 575


SELECT id_produit, SUM(montant_journalier) FROM ventes_monoprix
GROUP BY rollup(id_produit);
-- Pareil que la requête de base


SELECT id_produit, SUM(montant_journalier) FROM ventes_monoprix
GROUP BY cube(id_produit);
-- Pareil que la requête de base
```

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

```
SELECT id_produit, id_ville, SUM(montant_journalier) FROM ventes_monoprix
GROUP BY id_produit, id_ville;
-- Exemple : Produit Q dans la ville Y = 556 974


SELECT id_produit, id_ville, SUM(montant_journalier) FROM ventes_monoprix
GROUP BY rollup(id_produit, id_ville);
-- Pareil que la requête de base avec en plus montant par produit
-- Exemple : Z = 19 796 575


SELECT id_produit, id_ville, SUM(montant_journalier) FROM ventes_monoprix
GROUP BY cube(id_produit, id_ville);
-- Pareil que rollup avec en plus total par ville
-- Exemple : ville Z = 20 344 082
```

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

```
SELECT id_produit, id_date, SUM(montant_journalier) FROM ventes_monoprix
GROUP BY id_produit, id_date;
-- Exemple : Produit Z à la date Y = 911 710


SELECT id_produit, id_date, SUM(montant_journalier) FROM ventes_monoprix
GROUP BY rollup(id_produit, id_date);
-- Pareil que la requête de base avec en plus montant par produit
-- Exemple : Z = 19 796 575


SELECT id_produit, id_date, SUM(montant_journalier) FROM ventes_monoprix
GROUP BY cube(id_produit, id_date);
-- Pareil que rollup avec en plus total par date
-- Exemple : date Z = 17 305 200
```

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

```
SELECT id_magasin, id_date, AVG(montant_journalier) FROM ventes_monoprix
GROUP BY id_magasin, id_date;
-- Exemple : Produit Z à la date Y = 60 780


SELECT id_magasin, id_date, AVG(montant_journalier) FROM ventes_monoprix
GROUP BY rollup(id_magasin, id_date);
-- Pareil que la requête de base avec en plus moyenne par produit
-- Exemple : Z = 49 921


SELECT id_produit, id_date, AVG(montant_journalier) FROM ventes_monoprix
GROUP BY cube(id_produit, id_date);
-- Pareil que rollup avec en plus moyenne par date
-- Exemple : date Z = 47 153
```

3.5. Donner le montant total des ventes par ville et par jour.

```
SELECT id_ville, id_date, SUM(montant_journalier) FROM ventes_monoprix
GROUP BY id_ville, id_date;
-- Exemple : Ville E à la date O = 438 836


SELECT id_ville, id_date, SUM(montant_journalier) FROM ventes_monoprix
GROUP BY rollup(id_ville, id_date);
-- Pareil que la requête de base avec en plus total par ville
-- Exemple : ville Z = 20 344 082


SELECT id_ville, id_date, SUM(montant_journalier) FROM ventes_monoprix
GROUP BY cube(id_ville, id_date);
-- Pareil que rollup avec en plus total par date
-- Exemple : date Z = 17 305 200
```

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

```
SELECT id_produit, id_ville, id_date, SUM(montant_journalier)
FROM ventes_monoprix
GROUP BY id_produit, id_ville, id_date;
-- Exemple : Produit N dans la ville G à la date I = 80 212


SELECT id_produit, id_ville, id_date, SUM(montant_journalier)
FROM ventes_monoprix
GROUP BY rollup(id_produit, id_ville, id_date);
-- Pareil que la requête de base avec en plus total par ville / produit
-- et total par produit
-- Exemple : Produit Z dans la ville Z = 740 819


SELECT id_produit, id_ville, id_date, SUM(montant_journalier)
FROM ventes_monoprix
GROUP BY cube(id_produit, id_ville, id_date);
-- Pareil que rollup avec en plus tous les triplets possible, soit total
-- par produit, total par ville, total par date, total par produit / ville
-- total par produit / date, total par ville / date et total par
-- produit / ville / date
```

```
-- Exemple : Produit Z de la ville Z à la date R = 145 361
-- Exemple : Produit Z à la date X = 860 106
```

4) Classification des faits

1. Un fait (j,p,c,m,x) existe lorsqu'un produit p est acheté par un client c le jour j au magasin m. La mesure x correspond au prix total.

Transactionnel et additif

2. Un fait (j,p,m,x) existe lorsqu'un produit p est acheté le jour j au magasin m. La mesure correspond au chiffre d'affaires.

Snapshot et additif

3. Un fait (j,p,m,x) existe pour chaque combinaison de produit p, magasin m et jour j. La mesure x correspond au stock de p en m le jour j.

Snapshot et semi-additif

4. Un fait (j,p,m,x) existe pour chaque combinaison de produit p, magasin m et jour j. La mesure x correspond au nombre de ventes de p en m cumulées depuis le début de l'année jusqu'au jour j.

Snapshot et semi-additif

5. Un fait (c,e,j) existe lorsqu'un appel du client c le jour j est traité par l'employé e. Aucune mesure n'existe.

Transactionnel et void-additif

6. Un fait (c,j,x) existe lorsqu'un client c le jour j laisse une note sur un produit acheté. La mesure x est la note donnée.

Transactionnel et non-additif

7. Un fait (c,e,j,x) existe lorsqu'un appel du client c le jour j est traité par l'employé e. La mesure x est la durée de l'appel en secondes.

Transactionnel et additif

8. Un fait (m,b,j,x) existe lorsque la monnaie m est changée à la banque b le jour j. La mesure x est le montant total échangé en euros.

Snapshot et additif

9. Un fait (m,b,j,x) existe lorsque la monnaie m est changée à la banque b le jour j. La mesure x est le cours de change moyen de m en euros pour toutes les transactions du jour j.

Snapshot et non-additif