TP 3-ENTREPOT DE DONNEES ET BIG DATA

- **Master Informatique**

- **Membres** :

  - 21908889: Adam DAIA **PARCOURS GL**

  - 22014701 : Mohamed DAFAOUI **PARCOURS GL**

2023/2024

Table des matières

PARTIE 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 : Cette requête vise à vérifier la disponibilité des billets pour une séance de film spécifique à un moment donné dans une ville Elle sert à interroger la base de données pour obtenir des informations sur les billets vendus pour une séance en temps réel.

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

Requête Analytique : : Cette requête vise à analyser les performances et les tendances liées à la sortie du dernier film de Star Wars. Elle ne concerne pas une opération transactionnelle en temps réel, mais plutôt une réflexion sur des décisions futures en se basant sur des données historiques.

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

Requête Analytique : Cette requête est axée sur l'analyse des données vis a vis des ventes de billet de cinema, elle récupère le nombre de place vendus pour chaque film et pour chaque cinema sur chaque mois

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

Requête Analytique : Cette requete est conçue pour étudier les créneaux horaires associés aux ventes de billets

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

Requete Transactionnel : On insere seulement des informations dans la table ventes, on ne cherche pas à analyser une donnée mais plus tot a modifier la table vente

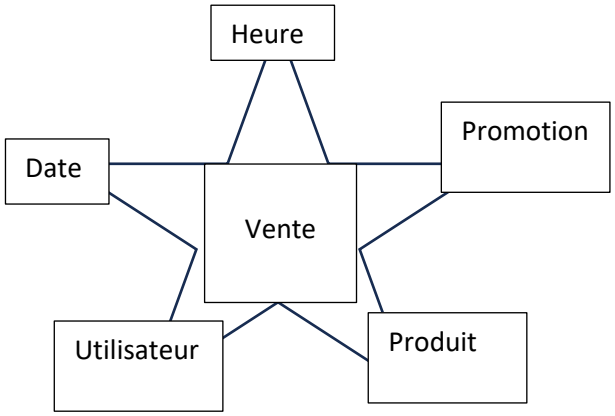
2) Un entrepôt de données pour Amazon

1) ***Proposez un modèle en étoile pour un entrepôt de données permettant l'analyse des ventes dans Amazon. Le modèle doit pouvoir permettre une analyse temporelle des ventes en fonction des produits, desutilisateurs, ainsi que des promotions mises en place***

Réflexion :

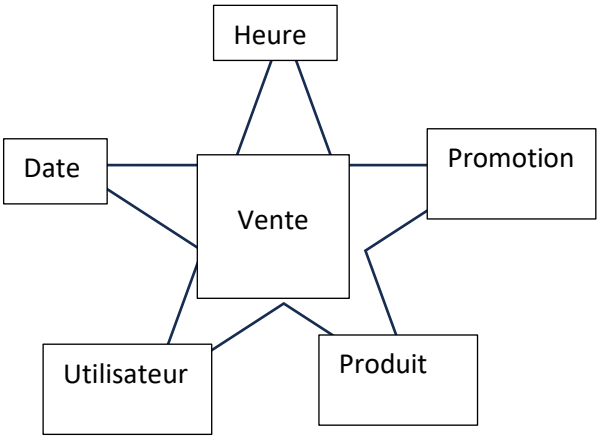
A quelle moment (date, temps) a été faites la vente d'un produit X est faite ?
Quelles sont le nombre de ventes par utilisateurs ?
Est ce qu'une promotion a été soumise à une vente ?

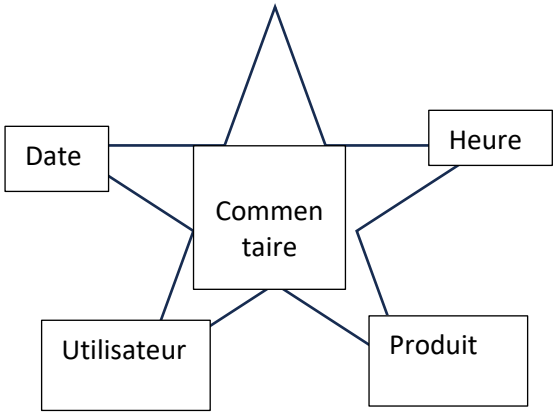
Après s'être questionné sur les éléments clés de notre structure nous avons réalisé le modèle suivant



2) ***Comment feriez-vous pour intégrer les commentaires client dans ce modèle ?***

Notre première réflexion a été de créer une deuxième table de fait pour le commentaire comme nous l'avons fait pour la vente





Mais nous en avons observé que cela revient a faire le produit cartésien de 2 tables de faits, ce qui produira énormément de ligne. Nous avons donc conclu qu'il serait préférable de ne pas ajouter de commentaire a ce modèle.

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

- *Combien de ventes ont été effectué à la date du 5/10/2023 ?*

- *Quelles sont les promotions qui ont réalisés le plus de ventes ?*

- *SELECT PRODUIT.CODE, COUT(U.NOM) FROM VENTES, PRODUIT, UTILISATEUR U*
  WHERE VENTES.CODE=PRODUIT.CODE AND VENTES,NOM=U.NOM
  GROUP BY PRODUIT.CODE , U.NOM

- *SELECT ID, SUM(vente) AS TOTAL_ACHATS FROM VENTE*
  GROUP BY ID
  ORDER BY TOTAL_ACHATS DESC LIMIT 5;-

- *SELECT CATEGORIE, PRODUIT.CODE, COUNT(PRODUIT.CODE) AS QUANTITE_VENDUE FROM VENTE*
  GROUP BY CATEGORIE, PRODUIT.CODE
  ORDER BY CATEGORIE, QUANTITE_VENDUE DESC;

3) Requêtes Analytiques

```
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 );
```

1. Donner le montant total des ventes par produit

```
SELECT id_produit, SUM(montant_journalier) AS MontantJournalier
        FROM ventes_monoprix
        GROUP BY id_produit;
```

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

```
SELECT id_produit, id_ville, SUM(montant_journalier) AS MontantJournalier
        FROM ventes_monoprix
        GROUP BY id_produit  , id_ville ;
```

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

```
SELECT id_produit, id_date, SUM(montant_journalier) AS MontantJournalier
        FROM ventes_monoprix
        GROUP BY id_date  , id_ produit ;
```

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

```
SELECT id_date, id_magasin, AVG (montant_journalier) AS MoyenneJournalier
        FROM ventes_monoprix
        GROUP BY id_date, id_magasin;
```

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

```
SELECT id_ville, id_date, SUM (montant_journalier) AS MontantJournalier
        FROM ventes_monoprix
        GROUP BY id_ville, id_date;
```

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

```
SELECT id_produit ,id_ville, id_date, SUM(montant_journalier) AS MontantJournalier
        FROM ventes_monoprix
        GROUP BY id_produit , id_ville, id_date;
```

Réécriture des requêtes précédentes avec ROLLUP :

1. *SELECT id_produit, SUM (montant_journalier) AS MontantJournalier*
   FROM ventes_monoprix
   GROUP BY ROLLUP (id_produit);

2. *SELECT id_produit, id_ville, SUM (montant_journalier) AS MontantJournalier*
   FROM ventes_monoprix
   GROUP BY ROLLUP (id_produit, id_ville);

3. *SELECT id_produit, id_date, SUM (montant_journalier) AS MontantJournalier*
   FROM ventes_monoprix
   GROUP BY ROLLUP (id_produit, id_date);

4. *SELECT id_date, id_magasin, AVG (montant_journalier) AS MoyenneJournalier*
   FROM ventes_monoprix
   GROUP BY ROLLUP (id_date, id_magasin);

5. *SELECT id_ville, id_date, SUM (montant_journalier) AS MontantJournalier*
   FROM ventes_monoprix
   GROUP BY ROLLUP (id_ville, id_date);

6. *SELECT id_produit, id_ville, id_date, SUM (montant_journalier) AS MontantJournalier*
   FROM ventes_monoprix
   GROUP BY ROLLUP (id_produit, id_ville, id_date);

Réécriture des requêtes précédentes avec CUBE :

1. *SELECT id_produit, SUM (montant_journalier) AS MontantJournalier*
   FROM ventes_monoprix
   GROUP BY CUBE *(id_produit);*

2. *SELECT id_produit, id_ville, SUM (montant_journalier) AS MontantJournalier*
   FROM ventes_monoprix
   GROUP BY CUBE *(id_produit, id_ville);*

3. *SELECT id_produit, id_date, SUM (montant_journalier) AS MontantJournalier*
   FROM ventes_monoprix
   GROUP BY CUBE *(id_produit, id_date);*

4. *SELECT id_date, id_magasin, AVG (montant_journalier) AS MoyenneJournalier*
   FROM ventes_monoprix
   GROUP BY CUBE *(id_date, id_magasin);*

5. *SELECT id_ville, id_date, SUM (montant_journalier) AS MontantJournalier*
   FROM ventes_monoprix
   GROUP BY CUBE *(id_ville, id_date);*

6. *SELECT id_produit, id_ville, id_date, SUM (montant_journalier) AS MontantJournalier*
   FROM ventes_monoprix
   GROUP BY CUBE *(id_produit, id_ville, id_date);*

Avec ROLLUP

```
ID_PRODUIT MONTANTJOURNALIER
---------- -----------------

A     19620355,7
B     19406517,8
C     19618576
D     20035040,4
E     17807074,1
F     18586375,4
G     16861680,1
H     18490601,9
I     17806841,1
J     19158948
K     17750387,6

ID_PRODUIT MONTANTJOURNALIER
---------- -----------------
L     20174639,5
M     20003909,6
N     22176027
O     19707949,9
P     18501655,8
Q     20161393,7
R     18597457,5
S     18457617,5
T     19811453,4
U     20010973,5
V     19115978

ID_PRODUIT MONTANTJOURNALIER
---------- -----------------
W     20044239,4
X     18460292,8
Y     19055782,2
Z     19796575,9
    499218344

27 lignes sélectionnées.
```

Avec CUBE

```
ID_PRODUIT MONTANTJOURNALIER
---------- -----------------
    499218344
A     19620355,7
B     19406517,8
C     19618576
D     20035040,4
E     17807074,1
F     18586375,4
G     16861680,1
H     18490601,9
I     17806841,1
J     19158948

ID_PRODUIT MONTANTJOURNALIER
---------- -----------------
K     17750387,6
L     20174639,5
M     20003909,6
N     22176027
O     19707949,9
P     18501655,8
Q     20161393,7
R     18597457,5
S     18457617,5
T     19811453,4
U     20010973,5

ID_PRODUIT MONTANTJOURNALIER
---------- -----------------
V     19115978
W     20044239,4
X     18460292,8
Y     19055782,2
Z     19796575,9

27 lignes sélectionnées.
```

Avec **ROLLUP**, la somme totale apparaît à la fin des résultats, ce qui crée une agrégation hiérarchique, où vous avez d'abord les totaux pour chaque niveau d'agrégation, suivi du total global.

Avec **CUBE**, la somme totale apparaît au début des résultats, créant une vue d'ensemble exhaustive de toutes les combinaisons possibles.

On peut regrouper les interrogations en utilisant les options ROLLUP et CUBE. Ces deux options permettent de regrouper et d'agréger des données de manière plus souple, ce qui facilite l'analyse de données. En les utilisant, on peut obtenir des résumés de données à différents niveaux d'agrégation dans une seule requête SQL.

PARTIE 2

4) Classification des faits

a) Précisez s'il s'agit d'un fait transactionnel ou d'un snapshot. Justifiez votre réponse.
b) Précisez si la mesure correspondante est additive, semi-additive ou non-additive

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.

- Il s'agit d'un fait transactionnel. La raison en est que ce fait enregistre les achats au fur et à mesure qu'ils se produisent en temps réel. Chaque transaction est capturée lorsqu'un client effectue un achat particulier à une date donnée.
- La mesure x, qui correspond au prix total, est additive. Une mesure additive peut être sommée ou agrégée.

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.

- Il s'agit en réalité d'un fait transactionnel, il enregistre toujours des transactions spécifiques au fur et à mesure qu'elles se produisent. Chaque transaction est capturée lorsque quelqu'un achète un produit donné à une date précise dans un magasin spécifique.
- La mesure x, qui correspond au chiffre d'affaires, est additive. On peut sommer ou agréger de manière significative les chiffres d'affaires pour obtenir des totaux, des moyennes

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.

- Il s'agit d'un snapshot, car il capture l'état du stock des produits dans les magasins pour chaque jour précis. Les données ne changent pas en fonction des transactions passées, nous avons une vue fixe à un moment précis.
- La mesure x, qui correspond au stock du produit p dans le magasin m le jour j, est non-additive. Le stock n'est pas une mesure que l'on peut simplement additionner, car il ne s'agit pas d'une quantité cumulative. Le stock est une mesure de l'état à un moment précis, ce qui le rend non-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.

- Il s'agit d'un fait transactionnel. Ce fait enregistre les ventes cumulées au jour j pour chaque produit p dans le magasin m. Les données évoluent au fur et à mesure que de nouvelles ventes sont enregistrées, ce qui les classe comme des données transactionnelles.
- La mesure x, qui correspond au nombre de ventes cumulées depuis le début de l'année jusqu'au jour j, est additive. On peut additionner ces valeurs de ventes cumulées pour obtenir le total de ventes cumulées à une date spécifique.

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.

- Il s'agit d'un fait transactionnel. Ce fait enregistre les appels des clients au fur et à mesure qu'ils se produisent en temps réel. Chaque entrée représente le traitement d'un appel par un employé à une date donnée.
- Dans ce cas, aucune mesure n'est associée au fait. Par conséquent, il n'y a pas de mesure à classer.

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 par le client.

- Il s'agit d'un fait transactionnel. Le fait enregistre les notes laissées par les clients au fur et à mesure qu'ils se produisent en temps réel. Chaque entrée représente une note donnée par un client à une date donnée.
- La mesure x, qui correspond à la note donnée par le client, est additive. Les notes peuvent généralement être additionnées ou agrégées pour obtenir des statistiques, telles que la moyenne des notes données par les clients. Par conséquent, la mesure x est additive.

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.

- Il s'agit d'un fait transactionnel. Le fait enregistre la durée des appels traités par les employés au fur et à mesure qu'ils se produisent en temps réel. Chaque entrée représente la durée d'un appel entre un client, un employé et à une date donnée.
- La mesure x, qui correspond à la durée de l'appel en secondes, est additive. On peut additionner ces valeurs de durée des appels pour obtenir le total de la durée des appels traités par un employé à une date donnée, ou pour calculer la moyenne de la durée des appels. La mesure x est donc considérée comme additive.

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 de la monnaie changée en euros.

- Il s'agit d'un snapshot. Le fait capture l'état du montant total de la monnaie changée en euros pour une journée j. Les données ne changent pas en fonction des transactions passées, elles représentent une vue fixe à un moment précis.
- La mesure x, qui correspond au montant total de la monnaie changée en euros, est additive. On peut additionner ces valeurs de montants totaux de monnaie changée pour obtenir le total du montant en euros échangé à la banque pour une journée donnée. La mesure x est donc considérée comme additive.

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.

- Il s'agit d'un fait transactionnel. Le fait enregistre les transactions de change de monnaie au fur et à mesure qu'ils se produisent en temps réel. Chaque entrée représente une transaction de change de monnaie entre une monnaie m, une banque b, et à une date donnée.
- La mesure x, qui correspond au cours de change moyen de m en euros pour toutes les transactions du jour j, est semi-additive. Un cours de change moyen peut être additionné sur plusieurs transactions de la journée, mais il ne peut pas être additionné sur une période plus longue cela pourrait donner des résultats incorrects. Le cours de change moyen dépend du nombre de transactions de change effectuées au cours de la journée, ce qui en fait une mesure semi-additive.