Entrepôt de donnée et big-data

Forme normale

1NF : Une table est en première forme normale (1NF) si elle est dotée d'une clé primaire (0NF) et que toutes les colonnes contiennent des valeurs atomiques.

2NF : Un attribut non identifiant ne dépend pas d'une partie de l'identifiant mais de tout l'identifiant

3NF : Un attribut non identifiant ne dépend pas d'un ou plusieurs attributs ne participant pas à l'identifiant

SQL

Créer table :

```sql
-- Table des Photographie
CREATE TABLE Photographie (
    photo_id INT PRIMARY KEY,
    lieu VARCHAR(255),
    date_prise DATE,
    appareil_id INT,
    FOREIGN KEY (appareil_id) REFERENCES AppareilPhoto (appareil_id),
);
```

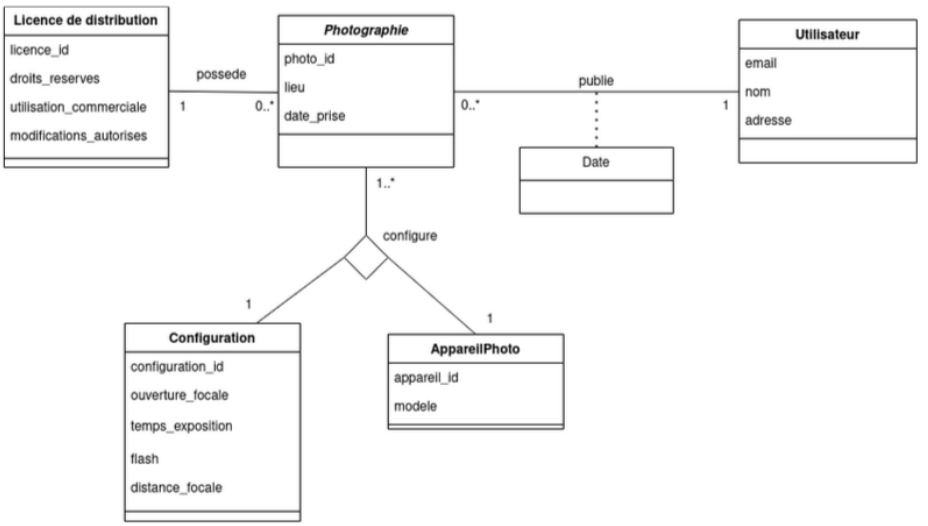
Insertion table :

```sql
INSERT INTO Photographie VALUES(1, 'Montpellier', TO_DATE('2023-09-19', 'YYYY-MM-DD'), 1, 1, 1);
```

Exemple requête :

```sql
-- Sélectionner les photographie de montpellier avec le flash depuis la base de données
SELECT *
FROM Photographie
WHERE lieu = 'Montpellier' AND configuration_id IN (
    SELECT configuration_id
    FROM Configurations
    WHERE flash = 1
);
```

UML

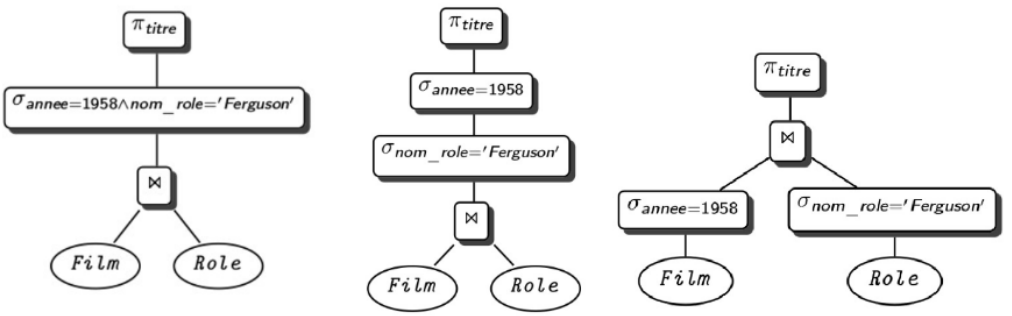


Plan d'exécution logique

Requête

```sql
select titre
from   Film f, Role r
where  nom_role ='Ferguson'
and    f.id = r.id_ilm
and    f.annee = 1958
```

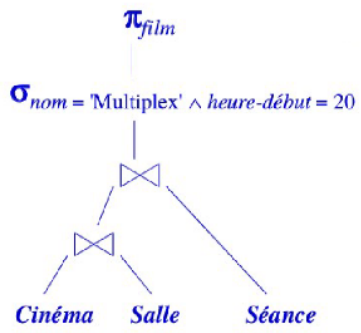
PEL



Écriture algébrique

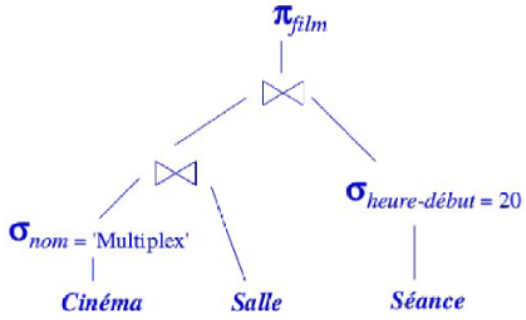
1. **Commutativité des jointures** : $R \bowtie S \equiv S \bowtie R$
2. **Regroupement des sélections** : $\sigma_{A='a' \wedge B='b'}(R) \equiv \sigma_{A='a'}(\sigma_{B='b'}(R))$
3. **Commutativité de σ et de π** : $\pi_{A_1,A_2,\ldots A_p}(\sigma_{A_i='a'}(R)) \equiv \sigma_{A_i='a'}(\pi_{A_1,A_2,\ldots A_p}(R))$
4. **Commutativité de σ et de $\bowtie$** : $\sigma_{A='a'}(R[\ldots A \ldots] \bowtie S) \equiv \sigma_{A='a'}(R) \bowtie S$

$\pi_{\text{film}}(\sigma_{\text{nom = 'Multiplex'} \wedge \text{heure-début=20}}((\text{Cinéma} \bowtie \text{Salle}) \bowtie \text{Séance}))$




```
SELECT Séance.film
FROM Cinéma, Salle, Séance
WHERE Cinéma.nom = 'Multiplex' AND
      Séance.heure-début = 20 AND
      Cinéma.ID-cinéma = Salle.ID-cinéma AND
      Salle.ID-salle = Séance.ID-salle ;
```

$\pi_{\text{film}}(\sigma_{\text{nom = 'Multiplex'} \wedge \text{heure-début=20}}\text{Séance}) \bowtie ((\sigma_{\text{nom = 'Multiplex'}}\text{Cinéma}) \bowtie \text{Salle}))$



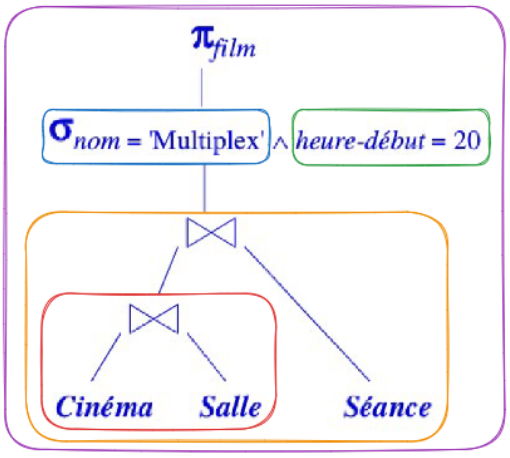
· Calcul des coûts

- **Hypothèses (en nombre de lignes) :**
  - Cinéma : 4 lignes dont 20 % de Multiplex
  - Salle : 6 lignes dont 50 % des salles de Cinéma
  - Séance : 50 lignes et 50 % des séances après 20h

- **Plan 1 :**
  - Jointure : on lit 4 * 6 = 24 lignes
    et on produit 50 % * 6 = 3 lignes
  - Jointure : on lit 3 * 50 = 150 lignes
    et on produit 50 lignes
  - Sélection : on lit 50 lignes
    et on produit 50 % * 50 = 25 lignes
  - Sélection : on lit 25 lignes
    et on produit 20 % * 25 = 5 lignes
  - On laisse de côté la projection (même coût dans les deux cas et même nombre de lignes)

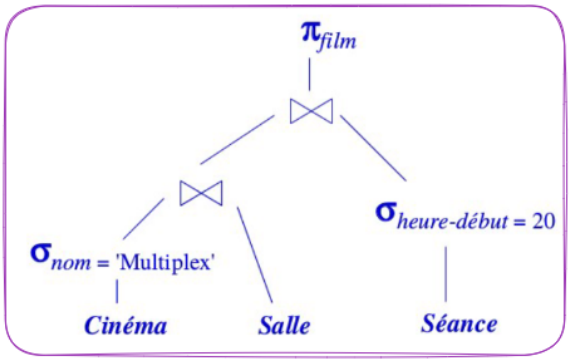
  ➡ Coût (E/S) : 24E + 3S + 150E + 50S +50E + 25S + 25E + 5S = 332 lignes E/S

Gardé la plus grande valeur des deux (50 > 3)
si les tables on au moins une colonne en commun sinon prendre le résultat du produit cartésien



- **Plan 2 :**
  - Sélection : on lit 4 lignes
    et on produit 20 % * 4 = 1 lignes
  - Jointure : on lit 1 * 6 = 6 lignes
    et on produit 50 % * 6 = 3 lignes
  - Sélection : on lit 50 lignes
    et on produit 50 % * 50 = 25 lignes
  - Jointure : on lit 25 * 3 = 75 lignes
    et on produit 25 lignes
  - On laisse de côté la projection (même coût dans les deux cas et même nombre de lignes)

  ➡ Coût (E/S) : 4E + 1S + 6E + 3S + 50E + 25S + 75E + 25S = 189 lignes E/S



Meilleur plan !

· Optimiseur ORACLE

Comment lire un plan d'exécution Oracle ?

- Parcours des étapes de haut en bas jusqu'à en trouver une qui n'a pas de fille (pas d'étape indentée en dessous)
- Traitement de cette étape sans fille ainsi que de ses sœurs (étapes de même indentation)
- Traitement de toutes les étapes mères jusqu'à trouver une étape qui a une sœur
- Traitement de la sœur conformément à l'étape 1.

```
-----------------------------------------------------
|   0 | SELECT STATEMENT
|   1 |  NESTED LOOPS
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP
|*  3 |    INDEX FULL SCAN           | N_DEPT_IDX |
|   4 |   TABLE ACCESS BY INDEX ROWID| DEPT       |
|*  5 |    INDEX UNIQUE SCAN         | DEPT_PK    |
-----------------------------------------------------
```

3 ↦ 2 ↦ 5 ↦ 4 ↦ 1 ↦ 0

SELECT v.nom, d.nom, r.nom
FROM ville v
JOIN departement d ON v.dep = d.id
JOIN region r ON d.reg = r.id
AND r.id = 91;

Le plan est le suivant :

1. Parcours un Index Unique (UNIQUE SCAN) avec la condition "region.id = 91".
2. Accède à la table Région par cet Index (BY INDEX).
3. Parcours une plage de donnée par Index (INDEX RANGE SCAN) de l'index IDX_REG_DEP avec la condition "departement.reg = 91".
4. Accède à la table Departement par cet Index par lots (BY INDEX BATCHED).
5. Joins les tables (NESTED LOOPS).
6. Accède à toute la table Ville (ACCESS FULL).
7. Parcours une plage de donnée par Index (INDEX RANGE SCAN) de l'index IDX_DEP_VILLE avec la condition "ville.dep = departement.id".
8. Joins les tables (NESTED LOOPS).
9. Accède à la table Ville par Index (BY INDEX ROWID).
10. Joins les tables (NESTED LOOPS).
11. Sélectionne les lignes.
    => Cette requête passe donc par tous nos Index ce qui limite son coût.

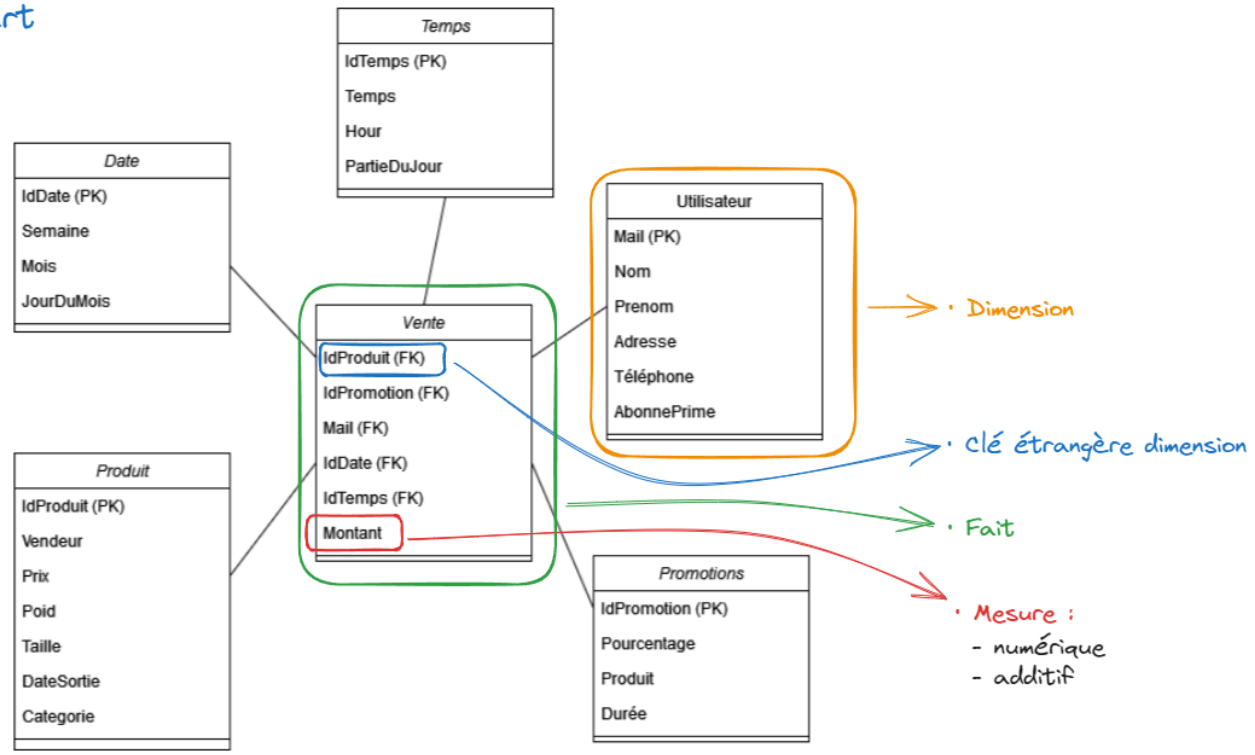
*Il récupère cela en un cout de **21,** en **0.01 seconde**.*
*Il y a eu **1 recursive calls** et **232 consistent gets**.*

OPERATIONS	OPTIONS	SIGNIFICATION
AGGREGATE	GROUP BY	Une recherche d'une seule ligne qui est le résultat de l'application d'une fonction de group à un groupe de lignes sélectionnées.
AND-EQUAL		Une opération qui a en entrée des ensembles de rowids et retourne l'intersection de ces ensembles en éliminant les doublants. Cette opération est utilisée par le chemin d'accès par index.
CONNECT BY		Recherche de ligne dans un ordre hiérarchique
COUNTING		Opération qui compte le nombre de lignes sélectionnées.
FILTER		Accepte un ensemble de ligne, appliqué un filter pour en éliminer quelque unes et retourne le reste.
FIRST ROW		Recherché de le première ligne seulement.
FOR UPDATE		Opération qui recherche et verrouille les lignes pour une mise à jour
INDEX	UNIQUE SCAN	Recherche d'une seule valeur ROWID d'un index.
INDEX	RANGE SCAN	Recherche d'une ou plusieurs valeurs ROWID d'un index. L'index est parcouru dans un ordre croissant.
INDEX	RANGE SCAN DESCENDING	Recherche d'un ou plusieurs ROWID d'un index. L'index est parcouru dans un ordre décroissant.
INTERSECTION		Opération qui accepte deux ensembles et retourne l'intersection en éliminant les doublons.
MARGE JOIN+		Accepte deux ensembles de lignes (chacun est trié selon un critère), combine chaque ligne du premier ensemble avec ses correspondants du deuxième et retourne le résultat.
MARGE JOIN+	OUTER	MARGE JOIN pour effectuer une jointure externe
MINIUS		Différence de deux ensembles de lignes.
NESTED LOOPS		Opération qui accepte deux ensembles, l'un externe et l'autre interne. Oracle compare chaque ligne de l'ensemble externe avec chaque ligne de l'ensemble interne et retourne celle qui satisfont une condition.
NESTED LOOPS	OUTER	Une boucle imbriquée pour effectuer une jointure externe.
PROJECTION		Opération interne
REMOTE		Recherche de données d'une base distante.
SEQUENCE		Opération nécessitant l'accès à des valeurs du séquenceur
SORT	UNIQUE	Tri d'un ensemble de lignes pour éliminer les doublons.
SORT	GROUP BY	Opération qui fait le tri à l'intérieur de groupes
SORT	JOIN	Tri avant la jointure (MERGE-JOIN).
SORT	ORDER BY	Tri pour un ORDER BY.
TABLE ACCESS	FULL	Obtention de toutes lignes d'une table.
TABLE ACCESS	CLUSTER	Obtention des lignes selon la valeur de la clé d'un cluster indexé.
TABLE ACCESS	HASH	Obtention des lignes selon la valeur de la clé d'un hash cluster
TABLE ACCESS	BY ROW ID	Obtention des lignes on se basant sur les valeurs ROWID.

· Entrepôt de donnée

Aspect	Operational DB	DW
User	clerk	manager
Interaction	short (s)	long analyses (min,h)
Type of interaction	Insert, Update, Delete	Read,periodically (bulk) inserts
Type of query	many simple queries	few, but complex queries (typically drill-down, slice...)
Query scope	a few tuples (often 1)	many tuples (range queries)
Concurrency	huge (thousands)	limited (hundreds)
Data source	single DB	multiple independant DB...
Schema	query-independant (3NF)	based on queries
Data	original, detailed, dynamic	derived,consolidated, integrated,historicized,partially aggregated,stable
Size	MB,GB cost of redundancy	TB,PB
Availability	crucial	not so crucial
Architecture	3-tier (ANSI-SPARC)	adapted to data integration

· Data-mart



· Mesure additive : Agrégation par somme valide sur toutes les dimensions.
Exemple : Chiffre d'affaires.

· Mesure semi-additive : L'agrégation varie selon la dimension.
Exemple : Solde bancaire (somme sur certains jours, dernier solde sur d'autres).

· Mesure non-additive : L'agrégation n'a pas de signification sur toutes les dimensions.
Exemple : Taux de change (peut nécessiter une moyenne sur certaines dimensions).

* Les datamart peuvent être en flocon. Un datamart en flocon signifie des les dimension peuvent elles aussi avoir des dimension. Par exemple un produit pourrait avoir une marque représenté par une nouvelle dimension associé.

- évite les redondance
- meilleure maintenance des valeurs des dimensions
- pénalisent la recherche transversale d'attributs (nécessitant des jointures)
- interdit l'utilisation d'index bitmap

* Une dimension dégénérée est une dimension sans table séparée, avec ses détails inclus directement dans la table de faits.

Par exemple, un numéro de commande dans une table de transactions.

* La granularité dans cet exemple se manifeste par le niveau de détail choisi pour l'analyse des données.

Par exemple, pour les ventes, la granularité par type d'article permet une analyse précise de chaque type d'article dans un achat, tandis que la granularité par transaction offre simplement un résumé de l'achat (nombre d'articles, total), présentant une valeur d'analyse limitée. La granularité par article, bien que trop détaillée ici, ne procure aucun avantage significatif.

· Rollup & Cube

```
SELECT   departement, manager, AVG(salaire)
FROM     employees
GROUP BY departement,manager
```

ROLLUP

CUBE

GROUP BY

departement	manager	AVG(salaire)
D1	M1	2000
D1	M3	1280
D2	M2	3400

departement	manager	AVG(salaire)
D1	M1	2000
D1	M3	1280
D1		1640
D2	M2	3400
D2		3400
		2222.6

departement	manager	AVG(salaire)
		2226.6
	M1	2000
	M2	3400
	M3	1280
D1		1640
D1	M1	2000
D1	M3	1280
D2		3400
D2	M2	3400

· Type de requêtes

· Requête transactionnelle :

- Petit volume de donnée
- Requête simple, pour un instant T
- Manipulation de table

· Requête analytique :

- Grand volume de donnée
- Généralement grande période de temps
- Utilisé pour faire des analyses par la suite
- Utilise généralement des fonction d'agrégat (SUM, COUNT, AVG, ...)

· Gestion de données

· Transactionnel :

Le modèle transactionnel enregistre en temps réel chaque transaction effectuée dans une base de données, assurant la cohérence des données.

- scénarios où les données changent fréquemment

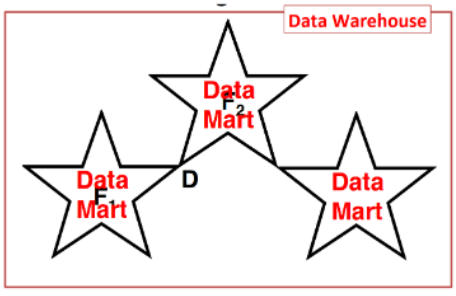
Par exemple, lorsqu'un client effectue un achat en ligne, le modèle transactionnel enregistre immédiatement les détails de cette transaction.

· Snapshot:

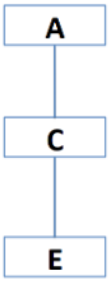
En parallèle, le modèle snapshot capture périodiquement l'état complet des données à des intervalles définis, facilitant l'analyse historique.

Par exemple, un modèle snapshot peut enregistrer chaque jour l'inventaire d'une entreprise, offrant des instantanés pour une analyse rétrospective des niveaux de stock.

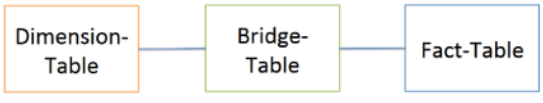
· Constellation



· Bridge



Manager	Employee	Distance	Bottom flag	Top flag
A	A	0		*true*
A	C	1		
A	E	2	(false otherwise)	
C	C	0		
C	E	1		
E	E	0	*true*	



Trouvez le montant total des ventes pour les personnes «en dessous» Manager «A» :

```
SELECT E.EmpID, SUM(F.Amount)
FROM Employee E, Bridge, Sales F
WHERE E.EmpID = Bridge.Manager
AND   Bridge.Employee = F.EmpID
AND   E.Name = 'A'
GROUP BY E.EmpID
```

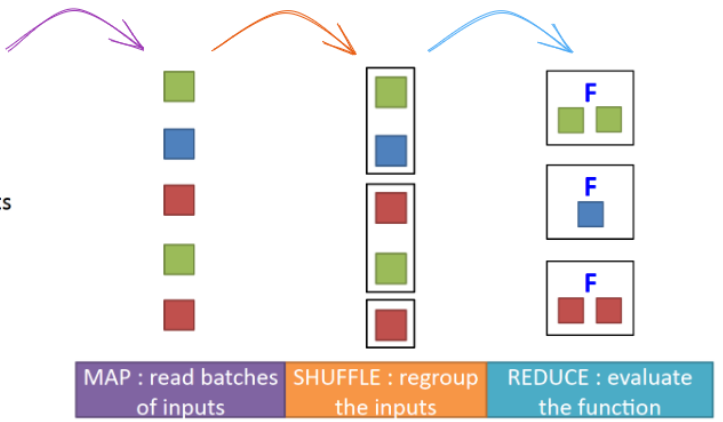
Book	Author	Order	Percentage
Winnie	Milne	1	1
Red House	Milne	1	1
C	Kernigan	1	0.5
C	Ritchie	2	0.5

Hadoop

Hadoop est un framework open-source conçu pour le stockage et le traitement distribué de données massives. Il utilise le Hadoop Distributed File System (HDFS) pour stocker les données sur des clusters de machines, et offre des outils tels que MapReduce, Spark, et Hive pour le traitement et l'analyse de ces données à grande échelle. Hadoop est particulièrement adapté au traitement de données volumineuses et à l'analyse de données distribuées.

Map/Reduce

1. **Read** the inputs
2. **Regroup**-the inputs
3. **Evaluate** a function on the regrouped inputs



Question Bonus

* Quelle est la différence entre une base de données relationnelles et un entrepôt de données ?

Une base de données relationnelle est conçue pour stocker et gérer des données opérationnelles en temps réel, en mettant l'accent sur la rapidité d'accès et la modification des informations. Elle utilise un modèle relationnel basé sur des tables interconnectées. En revanche, un entrepôt de données se concentre sur l'analyse et la prise de décision en agrégeant des données provenant de diverses sources. Il stocke des données historiques et agrégées, optimise la structure des données pour des analyses complexes, souvent au détriment de la vitesse des transactions. Les entrepôts de données utilisent fréquemment des modèles tels que l'étoile ou le flocon pour organiser les données de manière à faciliter les requêtes analytiques approfondies. En résumé, la base de données relationnelle répond aux besoins opérationnels, tandis que l'entrepôt de données fournit une vue consolidée pour des analyses métier approfondies.

* Pourquoi les bases de données relationnelles ne sont pas adaptées à la gestion des données massives ?

Les bases de données relationnelles ne sont pas toujours adaptées à la gestion des données massives en raison de leur schéma fixe, qui exige une définition préalable de la structure des tables, ce qui peut être contraignant pour des données évoluant rapidement. De plus, l'évolutivité horizontale, souvent nécessaire pour gérer des volumes massifs de données, peut être complexe à mettre en œuvre dans le contexte des bases de données relationnelles. Le coût de stockage peut également être élevé en raison de l'optimisation pour l'intégrité des données et la normalisation. Les performances peuvent être impactées par la complexité des requêtes et la nécessité de maintenir des intégrités référentielles. Enfin, les bases de données relationnelles sont mieux adaptées aux données structurées, et la gestion de données massives, souvent non structurées, peut être plus efficacement réalisée avec des solutions comme les bases de données NoSQL ou les systèmes de fichiers distribués tels que Hadoop.

* Pourquoi a-t-on introduit les plateformes de Big-Data ?
Quels sont les avantages et les inconvénients par rapport aux entrepôts de données ?

L'introduction des plateformes de Big Data a été motivée par la nécessité de traiter, stocker et analyser des volumes massifs et variés de données qui dépassent les capacités des entrepôts de données traditionnels. Les avantages des plateformes de Big Data résident dans leur capacité à gérer des données non structurées, à évoluer horizontalement pour gérer des volumes massifs, et à offrir des solutions plus flexibles pour des analyses complexes. Cependant, elles présentent des inconvénients tels que la complexité de mise en œuvre, la nécessité de compétences spécifiques, et des défis en termes de sécurité et de gouvernance des données, ce qui les rend complémentaires plutôt que substituts aux entrepôts de données classiques.

* Pourquoi est-il nécessaire d'optimiser l'évaluation des requêtes dans les bases de données relationnelles ?
Illustrez l'intérêt de l'optimisation avec un exemple

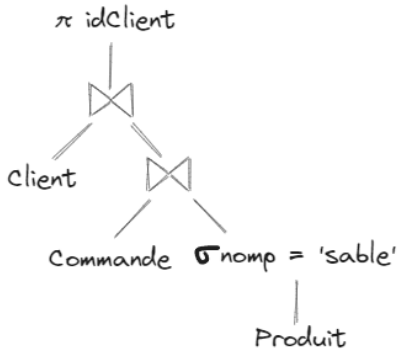
Il est crucial d'optimiser l'évaluation des requêtes dans les bases de données relationnelles pour améliorer les performances des opérations de récupération de données. Une optimisation efficace permet de réduire le temps d'exécution des requêtes, d'optimiser l'utilisation des ressources système et d'offrir une meilleure réactivité aux utilisateurs. Par exemple, dans une base de données contenant des millions d'enregistrements, une requête mal optimisée pourrait nécessiter un balayage complet de la table, entraînant des temps de réponse lents. En optimisant la requête, en utilisant des index appropriés, en ajustant les schémas de données, ou en sélectionnant des plans d'exécution efficaces, on peut significativement accélérer l'accès aux données, ce qui est essentiel pour garantir des performances optimales dans des environnements où la vitesse d'accès aux informations est cruciale.

PARTIEL

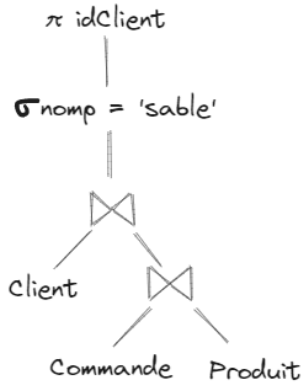
Question 1:

```
SELECT idClient
FROM Client c
JOIN Commande cm ON c.idClient = cm.idClient
JOIN Produit p ON cm.idProduit = p.idProduit
WHERE nomp = 'sable';
```

Question 2:



Question 3:



Question 4: Le premier car la sélection avec 'sable' réduit le nombre de produit

Question Map/Reduce:

Entrée Map :

Chaque ligne du fichier CSV, représentant une course en taxi.

Structure de chaque ligne : (Date/horaire début, Date/horaire fin, Nombre passagers, Prix Total (S))

Fonction Map :

Pour chaque ligne du fichier CSV, émettre une paire clé-valeur où la clé est la date (jour) et la valeur est un tuple contenant le nombre de passagers et le chiffre d'affaires de la course.

Pseudo-code :

```
map(line):
(date, passengers, revenue) = parse_line(line)
emit_intermediate(date, (passengers, revenue))
```

Sortie Map :

Paires clé-valeur émises par la fonction Map.

Entrée Reduce :

Un ensemble de paires clé-valeur regroupées par la clé (date).

Fonction Reduce :

Pour chaque clé (date), agréger les tuples associés en sommant les nombres de passagers et les chiffres d'affaires.

Émettre une paire clé-valeur où la clé est la date et la valeur est un tuple contenant le nombre total de passagers et le chiffre d'affaires total pour cette date.

Pseudo-code :

```
reduce(date, values):
    total_passengers = 0
    total_revenue = 0
for (passengers, revenue) in values:
        total_passengers += passengers
        total_revenue += revenue
emit(date, (total_passengers, total_revenue))
```

Sortie Reduce :

Paires clé-valeur finales représentant le nombre total de passagers et le chiffre d'affaires total pour chaque jour de l'année 2021.

Ce programme Map/Reduce effectue le calcul désiré en parallélisant le traitement des données pour chaque journée, puis agrège les résultats finaux pour obtenir les statistiques totales par jour.