LOG660 - Base de données de haute performance

# Chapitre 19: Entrepôts de données

## Solutions:

### QUESTION 1

**a)**

BD opérationnelles VS entrepôts de données:

| **BD opérationnelles** | **Entrepôts de données** |
|---|---|
| Données quotidiennes et récentes | Données d'archive |
| Données volatiles | Données statiques |
| Organisation permettant de traiter rapidement les requêtes | Organisation facilitant l'analyse des données |
| Gestion avancée de la concurrence | Gestion de concurrence rudimentaire |

**b)**

Schéma en étoile:

Le schéma en étoile contient 2 types de tables:

Tables de faits:

- Contiennent des colonnes des faits à analyser (mesures);
- Contiennent des clés étrangères vers les tables de dimension.

Tables de dimension:

- Décrivent les attributs des dimensions de l'analyse;
- Décrivent les niveaux de granularité de ces dimensions.

**c)**

CUBE:

Regroupe successivement les lignes d'une table selon chaque sous-ensemble de Colonnes.

*Exemple*:

`GROUP BY CUBE(a,b,c)` est équivalent à

- `GROUP BY a,b,c`
- `GROUP BY a,b`
- `GROUP BY a,c`
- `GROUP BY b,c`
- `GROUP BY a`
- `GROUP BY b`
- `GROUP BY c`
- aucun `GROUP BY`

ROLLUP:

Regroupe successivement les lignes d'une table selon chaque préfixe de Colonnes.

*Exemple*:

`GROUP BY ROLLUP(a,b,c)` est équivalent à

- `GROUP BY a,b,c`
- `GROUP BY a,b`
- `GROUP BY a`
- aucun `GROUP BY`

**d)**

Une hiérarchie dimensionnelle est formée d'attributs ayant des relations 1 à plusieurs entre eux. On pourrait avoir la hiérarchie suivante :

*catégorie* → *sousCatégorie* → *idProduit*

qui se traduit comme "une catégorie contient plusieurs sous-catégories, chacune d'elle renfermant plusieurs produits. On pourrait également définir des hiérarchies comportant d'autres attributs, par exemple :

*département* → *catégorie* → *sousCatégorie* → *marque* → *idProduit*

Faites attention au point suivant. Une certaine marque peut vendre des produits de différentes (sous) catégories. Donc, si on considère ces attributs de manière indépendante, la relation entre ceux-ci est de type plusieurs à plusieurs. Dans la hiérarchie ci-dessus, il faut plutôt considérer l'attribut *marque* comme étant un sous-élément de *sousCatégorie*. Si on imagine la hiérarchie comme une arborescence de répertoires, l'attribut *marque* dans cette hiérarchie correspondrait au répertoire suivant :

/[*département*]/[*catégorie*]/[*sousCatégorie*]/[*marque*]

**e)**

La pré-agrégation des faits correspond à pré-calculer des agrégations des données (par exemple : SELECT col1, col2, SUM(col3) … GROUP BY col1, col2) de manière à

accélérer les requêtes analytiques nécessitant de faire ces agrégations. Dans le contexte relationnel, ceci est typiquement fait à l'aide de vues matérialisées (i.e., table physique qui stocke et met à jour le résultat du pré-calcul des agrégations).

**f)**

L'architecture en bus de magasins de données est une approche incrémentale (*bottom-up*) de conception qui consiste à bâtir un entrepôt de données d'une entreprise un magasin de données à la fois. Chaque magasin de données est modélisé comme un schéma en étoile (i.e., une table de faits reliée à plusieurs tables de dimensions) et renferme les données d'un seul processus d'affaires (e.g., ventes, appels de services, commandes aux fournisseurs, etc.). L'intégration entre les différents magasins de données se fait avec les dimensions partagées, appelées dimensions conformes (e.g., client, produit, date, etc.)

**g)**

L'architecture fédérée permet d'unifier plusieurs entrepôts de données conçus de manière indépendante, et utilisant possiblement des technologies ou schémas de données différents, de manière transparente aux applications analytiques. Ce type d'architecture se base typiquement sur des métadonnées pour définir le mappage entre un schéma global et le schéma local de chaque entrepôt. Cette architecture est recommandée pour l'intégration rapide et à faible coût d'entrepôts de données hétérogènes pré-existants.

**h)**

La modélisation dimensionnelle facilite la compréhension des données par les utilisateurs d'affaires. Au lieu d'avoir l'information et les relations divisées en plusieurs dizaines de tables, les données sont représentées comme une table de faits refermant les mesures d'un évènement d'affaires (ex : le montant d'une vente, le profit, etc.), et des tables de dimensions donnant le contexte à cet évènement. L'autre avantage est une meilleure performance lors des requêtes analytiques car les tables sont dénormalisées et donc il y a moins de jointures à faire.

**i)**

Prenons l'exemple de la modélisation des ventes avec trois dimensions : Date, Client et Produit. Supposons la requête analytique suivante qui calcule le total des ventes de chaque produit par année:

```
SELECT Produit.code, Date.année, SUM(Ventes.montant) as total
FROM Produit, Date, Ventes
WHERE Produit.id = Ventes.idProduit AND Date.id = Ventes.idDate
GROUP BY Produit.code, Date.année
```

L'opération *slice* consiste à restreindre pour une certaine dimensions les valeurs sur

lesquelles est calculée l'agrégation. Par exemple, on peut limiter les produits à une certaine catégorie :

```
SELECT Produit.code, Date.année, SUM(Ventes.montant) AS total
FROM Produit, Date, Ventes
WHERE Produit.id = Ventes.idProduit AND Date.id =
Ventes.idDate
        AND Produit.catégorie = 'ordinateur'
GROUP BY Produit.code, Date.année
```

Un *roll-up* sert à changer le niveau de détail pour une des dimensions d'analyse, allant d'un niveau plus détaillé vers un niveau moins détaillé. Par exemple, on peut analyser les ventes selon la catégorie de produits au lieu d'analyser par produit :

```
SELECT Produit.catégorie, Date.année, SUM(Ventes.montant) as
total
FROM Produit, Date, Ventes
WHERE Produit.id = Ventes.idProduit AND Date.id =
Ventes.idDate
GROUP BY Produit.catégorie, Date.année
```

Le *drill-down* est l'opération inverse, par exemple si on passe de catégorie à code.

Enfin, le *rotate* consiste à changer une ou plusieurs dimensions d'analyses. Par exemple, on pourrait changer la dimension Date pour Client (niveau ville dans l'exemple):

```
SELECT Produit.code, Client.ville, SUM(Ventes.montant) as
total
FROM Produit, Client, Ventes
WHERE Produit.id = Ventes.idProduit AND Client.id =
Ventes.idClient
GROUP BY Produit.code, Client.ville
```

**j)**

ROLAP qui vient de *Relational OLAP* emploie les bases de données relationnelles pour réaliser les opérations d'analyse multidimensionnelles (OLAP). Comme mentionné à la question *e*), le ROLAP utilise typiquement les vues matérialisées pour accélérer les requêtes nécessitant l'agrégation d'un grand nombre de lignes de la table de faits.

En revanche, MOLAP veut dire *Multidimensional* OLAP. Ce type de BD repose plutôt sur une structure de cube multidimensionnel (tableau à D dimensions) et la compression de tableaux pour stocker et manipuler les données efficacement.

## QUESTION 2

**a)**

```
SELECT genre, annee
       count(*) OVER (
         PARTITION BY genre
         ORDER BY annee ASC
         ROWS UNBOUNDED PRECEDING) AS nbFilms
FROM Film F, GenreFilm GF
WHERE F.idFilm = GF.idFilm
```

**b)**

La requête 1 va présenter les résultats aggrégés par Film, pas la requête 2. Par exemple :

**Requête 1**

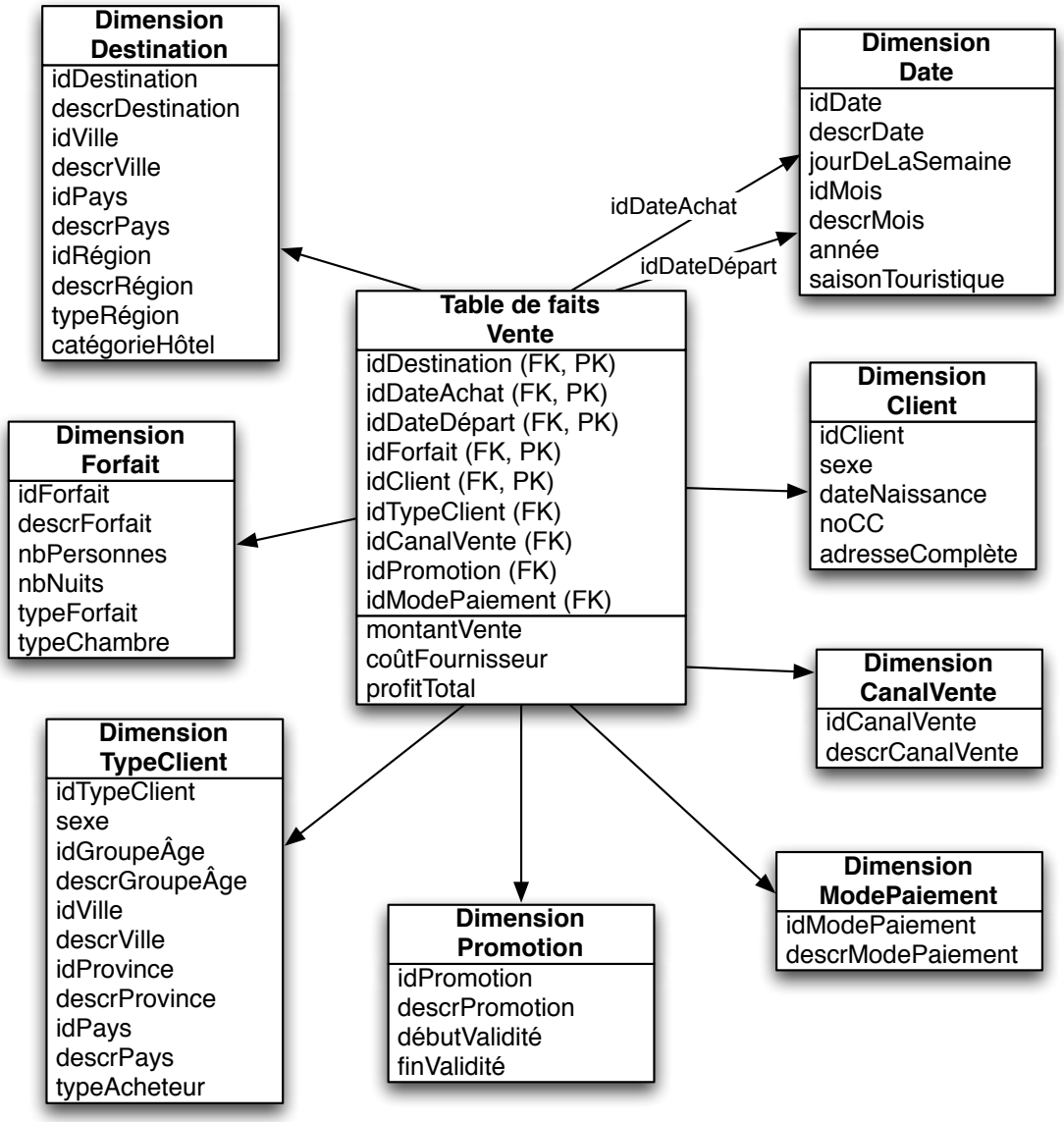
| idFilm | col |
|---|---|
| 1 | 2 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |

**Requête 2**

| idFilm | col |
|---|---|
| 1 | 2 |
| 1 | 2 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |

**QUESTION 3**

**a)**




Notes:

- La plupart des attributs dimensionnels ont un ID ainsi qu'un champ descriptif. Par exemple, dans la table Date, le mot 'Novembre' n'est pas suffisant pour identifier avec précision ce mois, car on le retrouve dans chacune des années. Il faut donc un attribut idMois (ex: '11/2010') ainsi qu'un attribut descriptif descrMois (ex: 'Novembre'). C'est la même chose pour l'attribut ville: le même nom de ville peut se trouver dans plusieurs pays ou même plusieurs fois dans un même pays;
- Nous avons créé une table TypeClient selon la stratégie de mini-dimension. L'avantage est que la table TypeClient peut être pré-générée (toutes les combinaisons possibles de sexe, ville, groupe d'âge, etc.). De même, les tables Destination, Date, Forfait, Promotion et CanalVentes peuvent également être pré-générée et ne sont (presque) jamais modifiées. Seule la

table de dimension Client est modifiée à chaque fois qu'un client s'ajoute au système;

- La clé primaire de la table de faits Vente est une clé composée car il est très rare que l'on accède individuellement les lignes de cette table. En revanche, les clés primaires des tables de dimension sont toujours des clés artificielles simples (ex: NUMBER).

**b)**

Les niveaux d'une hiérarchie doivent avoir une relation 1 à plusieurs: un parent peut avoir plusieurs enfants (ex: une année a plusieurs mois) mais chaque enfant n'a qu'un seul parent (ex: le mois '11/2010' appartient uniquement à l'année 2010).

| **Table de dimension** | **Hiérarchies** |
|---|---|
| Destination | idDestination ← idVille ← idPays ← idRégion ← tous |
| Date | idDate ← idMois ← année ← tous |
| Forfait | idForfait ← tous |
| Client | idClient ← tous |
| TypeClient | idTypeClient ← idVille ← idProvince ← idPays ← tous |
| CanalVente | idCanal ← tous |
| Promotion | idPromotion ← tous |
| ModePaiement | idModePaiement ← tous |

**c)**

Pour définir la stratégie d'agrégation, il faut choisir, pour chaque dimension, un niveau hiérarchique permettant de faciliter l'analyse. L'objectif est d'accélérer les calculs en pré-calculant les agrégations faites dans les requêtes analytiques fréquentes.

Ainsi, on prévoit que les analyses se feront aux niveaux suivants:

| **Dimension** | **Niveau hiérarchique retenu** |
|---|---|
| Destination | idPays |
| Date (achat) | tous |

| | |
|---|---|
| Date (départ) | idMois |
| Forfait | idForfait |
| Client | tous |
| TypeClient | idProvince |
| CanalVente | idCanal |
| Promotion | idPromotion |
| ModePaiement | tous |

Avec ces niveaux d'agrégation, on pourrait analyser les ventes par pays, mois de départ, forfait, province de client, canal de vente et promotion utilisée.

Pour créer la table de faits agrégés, on pourrait employer une vue matérialisée qui serait mise à jour incrémentalement à chaque modification de la table de faits principale:

```
CREATE MATERIALIZED VIEW VentesAgregees
REFRESH FAST ON COMMIT AS
SELECT D.idPays AS idPaysDestination,
               DD.idMois AS idMoisDepart,
               V.idForfait AS idForfait,
               TC.idProvince AS idProvinceClient,
               V.idCanalVente AS idCanalVente,
               SUM(V.montantVente) AS ventesAgregees,
               SUM(V.coutFournisseur) AS coutsAgreges,
               SUM(V.profitTotal) AS profitsAgreges
FROM Ventes V, Destination D, Date DD, TypeClient TC
WHERE V.idDestination = D.idDestination AND
              V.idDateDepart = DD.idDate AND
              V.idTypeClient = TC.idTypeClient
GROUP BY D.idPays, DD.idMois, V.idForfait, TC.idTypeClient,
V.idCanalVente
```

**QUESTION 4**

**Solution pour le datamart #1**

**1) Identifiez le principal évènement d'affaires**

Le principal évènement d'affaires est: « *la réservation par un client d'une chambre d'un certain type dans un hôtel, pour une certain période* ».

**2) Identifiez les attributs associés aux faits**

En se basant sur les questions analytiques, les trois métriques de performance sont :

- Le nombre de réservations
- La durée (moyenne) des séjours
- Les revenus non-attribuables aux restaurants

La dernière métrique s'obtient en multipliant le nombre de chambres (*Room_count*) avec le nombre de jours (*Number_of_days*) et le tarif de la chambre (*Room_standard_rate*)

**3) Identifiez les dimensions et leurs attributs**

En se basant sur les questions analytiques et les sources, les principales dimensions sont *Client*, *Hôtel*, *DateArrivée* et *TypeChambre*.

**Note**: bien que nous ne modélisons que le premier magasin de données, il est important de prévoir la modélisation du second magasin de données afin d'éviter d'avoir des silos de données. Pour ce faire, nous allons tenter de modéliser des dimensions conformes.

Dimension Clients (*DimCustomer*):

Cette dimension se retrouve dans les tables *Guests* et *Customers* des trois sources de données. Les attributs diffèrent par le fait que la table *Guests* comporte une colonne pour le prénom et une autre pour le nom de famille, alors que les autres tables ont une seule colonne pour le nom complet. Puisqu'il est préférable d'avoir des attributs séparés, nous choisissons cette option et laissons l'ETL diviser les noms complets dans les tables *Customers*. De même, on conserve l'attribut courriel de la table *Guests*.

**Note** : On suppose que les numéros de client sont les mêmes pour le système de réservation et de commandes. Si ce n'est pas le cas, il faudrait alors stocker les identifiants des deux sources dans la table de dimension.

Attributs :

- Customer_number
- Customer_first_name
- Customer_last_name
- Customer_address
- Customer_city
- Customer_zipcode
- Customer_email

Dimension Hôtel (*DimHotel*):

On commence par identifier les attributs communs à toutes les sources :

- Hotel_id

- Country_code
- Hotel_name
- Hotel_address
- Hotel_city
- Hotel_zipcode

On analyse ensuite les relations avec les autres tables dans la base de données de réservation. La relation plusieurs-à-un avec *Countries* correspond à une hiérarchie dimensionnelle Hôtel → Pays et on met les attributs de *Countries* dans la dimension :

- Country_currency
- Country_name

La relation un-à-plusieurs avec *HotelRooms* est plus complexe à gérer car il ne s'agît pas d'une hiérarchie dimensionnelle. Une solution consiste à résumer la relation à l'aide de statistiques dans la dimension Hôtel. Par exemple :

- Nb_ rooms_total
- Nb_rooms_type1
- ...
- Nb_rooms_typeN
- Nb_floors
- etc.

La relation plusieurs-à-plusieurs avec *Amenities* peut être dénormalisée en introduisant un attribut binaire (flag) et descriptifs pour chaque service :

- Amenity1_flag
- Amenity1_descr
- ...
- AmenityK_flag
- AmenityK_descr

Finalement, pour pouvoir faire une analyse sur la cote de l'hôtel, il faut rajouter un attribut

- Average_rating

La mise à jour de cet attribut est gérée par l'ETL.

<u>Dimension DateArrivée (*DimCheckinDate*)</u>:

Puisque la date peut jouer différents rôles, nous créons une dimension conforme Date, contenant des attributs standards :

- Day

- Month
- Year
- Day_of_week
- Holiday_flag
- ...

Dimension TypeChambre (*DimRoomType*):

Pour cette dimension, nous utilisons les colonnes de la table *Room types* :

- Room_type_code
- Room_type_standard_rate
- Room_description
- Smoking_YN

**4) Élaborez le schéma en étoile selon les principes vus en classe**