Mustapha NEZZARI

Dans ce compte rendu, vous trouverez les réponses au TP2.

Compte rendu du TP2

SQL OLAP

Sommaire

[Modélisation dimensionnelle 3](#_Toc467357500)

[Schéma en étoile 3](#_Toc467357501)

[L’entrepôt de données 3](#_Toc467357502)

[La dimension Produit 3](#_Toc467357503)

[La dimension Client 3](#_Toc467357504)

[La dimension Lieu 3](#_Toc467357505)

[La dimension Temps 3](#_Toc467357506)

[La table des faits Vente 3](#_Toc467357507)

[Les mesures 3](#_Toc467357508)

[Les vues matérialisées 3](#_Toc467357509)

[Requêtes 3](#_Toc467357510)

[Mode de rafraichissement 3](#_Toc467357511)

[Les clés 3](#_Toc467357512)

[Tests des vues 4](#_Toc467357513)

[Optimisations 4](#_Toc467357514)

[Les index 4](#_Toc467357515)

[Les hiérarchies 4](#_Toc467357516)

# Modélisation dimensionnelle

Dans le cas présenté, l’entreprise cherche à analyser ses ventes selon des mesures différentes. Elle s’intéresse aux **quantités** vendues, au **prix** de ventes (avec éventuellement des **remises**) et aux **chiffres d’affaire**.

Elle souhaite aussi pouvoir avoir des informations sur les produits vendus comme leur **désignation**, leur **catégorie**, leur **sous-catégorie**.

Elle s’intéresse aussi aux caractéristiques de ses clients. Les questions qu’elle se pose montrent un intérêt pour leur **âge**, leur **tranche d’âge**, leur **sexe**. Afin de cibler ses campagnes promotionnelles, elle s’intéresse à des critères géographiques locaux et internationaux comme le **pays**, la **ville** ou encore le **code postal**.

Enfin, elle veut archiver le tout afin de pouvoir effectuer des analyses temporelles sur certaines **années**, **mois**, **semaines** et **jours**.

En tenant compte de toutes ces informations, j’ai réalisé le schéma en étoile en Figure 1.

INSERER SHEMA EN ETOILE

On peut y voir la table de fait Vente et les différentes dimensions qui représentent les axes possibles d’analyse à savoir Client, Lieu, Produit et Temps.

# L’entrepôt de données

Afin de créer cet entrepôt de données, j’ai utilisé les vues matérialisées. Il s’agit ici d’expliquer certains choix ainsi que certaines méthodes utilisées.

## La table des faits Vente

La table des faits Ventes va contenir une entrée par ventes d’un produit en stockant les références du client, du produit vendu, du lieu de vente ainsi que de la date.

### Les mesures

En plus des références précédentes, elle va permettre de stocker différentes mesures qui sont :

* La quantité du produit vendu
* Le prix du produit au moment de la vente
* La remise s’il y en a eu une
* Le prix total de ventes qui correspond à la formule : Quantité x Prix unitaire de vente à la date de vente x remise en pourcentage.
* L’âge du client au moment de la vente. J’ai décidé de ne pas mettre l’âge directement dans la dimension produit car on aurait perdu cette notion d’archivage. De plus je pense qu’il est plus pertinent de connaître l’âge de la personne au moment de la vente car l’achat peut être influencé par l’âge et donc la mentalité et le contexte. Par exemple, on pourrait croire qu’une publicité a plus touché les personnes jeunes plutôt que les personnes âgées et l’on pourrait alors le savoir en mettant cette mesure dans la table des ventes.

### Script SQL de la vue matérialisée



Dans cette requête, la jointure avec la vue Temps se fait en reconstruisant de la même manière qu’il que dans la dimension temps. L’âge quant à lui correspond à la date de la facture – la date de naissance du client.

## La dimension Produit

Cette dimension représente un produit à vendre.

### Script SQL de la vue matérialisée

Explications

Dans le SIO on dispose de la table Produit qui possède trois attributs :

* Sa référence (qui ne changera pas dans le SID)
* Sa désignation qui est composé du nom du produit, de sa catégorie et, s’il y en a une, de sa sous-catégorie.
* Son stock

Il a donc fallu découper la désignation pour séparer le nom, la catégorie et la sous-catégorie grâce aux regex. Ici j’ai utilisé la regex '[^.]+' qui permet de dire « un ou plusieurs caractères excepté le point.

La fonction NVL () a permis d’insérer une valeur nulle lorsqu’il n’y a pas de sous-catégorie.

Dans la dimension produit je n’ai pas mis le stock car je pense que pour les questions qui permettent de savoir comment améliorer les ventes, on n’a pas besoin de connaître le stock du produit dans le temps. On pourrait le rajouter plus tard si l’on veut **par exemple** savoir les tendances de variations de stock selon les saisons.

### Script SQL de la dimension



### Hiérarchie

La hiérarchie choisi (du plus grand au plus petit niveau) est : Catégorie > Sous-catégorie > Nom > id

## La dimension Client

Cette dimension représente un client qui a acheté au moins une fois.

### Script SQL de la vue matérialisée



### Explications

Les champs retenus dans la vue sont :

* La référence du client (la même que dans la table du SIO)
* Le sexe (idem)
* La tranche d’âge calculée à partir de la date maximum de vente pour le client. Il pourrait être plus performant d’utiliser la du jour (SYSDATE) plutôt que la date maximum des factures du client.

Le reste des champs de la table Client du SIO ont été utilisé dans la dimension Lieu car la localité est un axe d’analyse.

### Script SQL de la dimension



### Hiérarchie

La hiérarchie choisi (du plus grand au plus petit niveau) est : sexe > tranche d’âge > id

## La dimension Lieu

Cette dimension représente le lieu où une (ou plusieurs) vente a été effectuée.

### Script SQL de la vue matérialisée



### Explications

Cette vue utilise les informations de la table Client du SIO. Elle est composée :

* D’un id qui correspond à la référence client car chaque client a une référence différente et est associé à une adresse. Même en utilisant l’adresse complète pour l’id, on n’aurait pas pu garantir l’unicité des clés car si deux clients sont deux personnes mariées et habitent à la même adresse, il y aurait des conflits de clés. C’est pourquoi il est plus judicieux de prendre l’id du client comme id ici.
* Un code d’état
* Le pays
* La ville
* Le code postal

Il a fallu découper l’adresse pour séparer les trois derniers champs de la même manière que la désignation des produits. Utiliser DISTINCT ?

### Script SQL de la dimension



### Hiérarchie

La hiérarchie choisi (du plus grand au plus petit niveau) est : Pays > Ville > Code postal > id

## La dimension Temps

Cette dimension est importante car elle permet l’historisation des données.

### Script SQL de la vue matérialisée



### Explications

Cette vue est constitué des champs suivants :

* L’id du temps qui est la concaténation du numéro du jour de l’année et de l’année (Par exemple : 365-2017)
* Du numéro du jour de l’année (de 1 à 365)
* Du mois (de 1 à 12)
* De l’année
* Du trimestre (de 1 à 4)
* Du libellé (Du Lundi au Dimanche)

Pour générer cette table, on utilise la sous requête avec la clause CONNECT BY qui permet de générer des dates entre deux dates. Le minimum et le maximum sont les dates de la plus ancienne et la plus récentes facture.

### Script SQL de la dimension



### Hiérarchie

La hiérarchie choisi (du plus grand au plus petit niveau) est : Année > Trimestre > Mois > Semaine > Jour de l’année > Libellé > id

## Les vues matérialisées

### Mode de rafraichissement

Selon moi, il n’est pas nécessaire de rafraichir les vues à chaque fois qu’il y a une entrée de fait dans le SIO car les données du SID seront utilisées ponctuellement et non de façon continue. J’ai donc décidé d’utiliser la méthode REFRESH FORCE ON DEMAND sur toutes mes vues, qui me permet de manuellement rafraîchir mes vues de façon incrémentale (si c’est possible, sinon toute la vue est rafraîchie).

Il faut donc utiliser les méthodes suivantes pour rafraîchir les vues à chaque fois que l’on veut s’en servir pour prendre des décisions :



Figure 1

### Tests

Afin de tester si mes vues fonctionnent, j’ai créé une procédure qui permet d’insérer 4 ventes dans les tables du SIO. Le script de cette procédure est disponible dans le fichier *procedure\_insert.sql*.

Il suffit d’appeler la procédure avec la commande « exec INSERER\_DONNEES; » et de rafraichir les vues grâce au commande de la Figure 1.

# Les clés primaires et étrangères

Dans notre cas, nous utilisons des vues matérialisée, il n’est donc pas nécessaires de créer des clés primaires puisqu’elles utilisent les clés des tables du SIO à part pour la table Temps où l’on génère nous même une clé. La création d’un index pourra apporter cette notion d’unicité (si l’on utilise un index unique).

# Optimisations

## Les index

### Script SQL



### Explications

Les index sont en général utilisés pour indexer des données afin de les retrouver plus rapidement à l’aide de hashmaps par exemple. On les utilise par exemple dans le cas où l’on a des jointures sur plusieurs tables sur lesquelles on a besoin de moins de 15% des données.

# VERIFIER QUE ORACLE A BIEN CR2ER DES INDEX AUTOMATIQUEMENT

# Questions d’exploitation

## Question 1

### Script SQL

### Résultat

### Explications

## Question 2

### Script SQL

### Résultat

### Explications

## Question 3

### Script SQL

### Résultat

### Explications

## Question 4

### Script SQL

### Résultat

### Explications