L'informatique des entrepôts de données

Daniel Lemire

SEMAINE 8
Introduction à OLAP

8.1. Présentation de la semaine

Le modèle OLAP (Online Analytical Processing) est un modèle quasi-omniprésent en intelligence d'affaires [3]. Il est une alternative au modèle relationnel, qui est mieux adapté à l'analyse des données multi-dimensionnelles. En utilisant notamment la matérialisation, les bases de données multidimensionnelles support OLAP permettent souvent un temps de réponse constant, indépendant de la taille de la base de données. Cette semaine nous ferons un survol du modèle OLAP en préparation pour les semaines prochaines où nous approfondiront le concept.

8.2. Vers une définition

Le modèle OLAP fut proposé par Codd [1], l'inventeur du modèle relationnel. Il n'existe pas de véritable définition du modèle OLAP. Cependant, plusieurs des termes et des concepts fréquemment associés au modèle OLAP sont faciles à définir. Considérons un exemple. Imaginons que votre entreprise fasse affaire dans plusieurs villes, avec plusieurs produits. Vous pourriez alors vouloir étudier des données ayant trait aux ventes :

Ville	Produit	Mois	Ventes ($)
Montréal	iPod	Janvier	421.32
Paris	Vélo	Février	100.12

Ce type de table est appelé "table de faits". Chacune des trois premières colonnes correspond à une dimension (ville, produit, mois). Dans certaines applications, il peut y avoir 10, 20 ou 30 dimensions. Les dimensions peuvent être sélectionnées de manière indépendante.

Ainsi, on peut se limiter aux données ayant trait à Montréal, sans pour autant devoir spécifier ou se limiter aux produits ou aux mois. Dans notre exemple, il s'agit de dimensions simples. Dans la pratique, les dimensions disposent souvent d'une hiérarchie. On pourrait donc remplacer la ville par une notion de lieu qui comprend la rue, le quartier, la ville, la province, le pays, le continent, etc. On peut remplacer le mois par une notion de temps comprenant la seconde, la minute, l'heure, le jour, la semaine, le mois, l'année, etc. Une même dimension peut avoir plus d'une hiérarchie simultanément : (année, trimestre, mois) et (année, semaine, jour). On associe normalement à chaque dimension une valeur spéciale (souvent notée $\top$) qui correspond à l'ensemble des valeurs (ainsi, tous les enregistrements satisfont à la condition Ville= $\top$). Il n'y a qu'un enregistrement dans la table de faits par valeur possible des dimensions. Ainsi, cette table ne serait pas une table de faits légitime :

Ville	Produit	Mois	Ventes ($)
Montréal	iPod	Janvier	321.00
Montréal	iPod	Janvier	120.32
Paris	Vélo	Février	100.12

La dernière colonne représente une mesure (les ventes). Il peut y avoir une ou plusieurs mesures. La mesure est associée implicitement ou explicitement à une opération d'agrégation. Ainsi, dans notre exemple,

il s'agit du total des ventes : nous avons effectué une somme. Si nous avions choisi un exemple scolaire avec comme mesure la note obtenue par les étudiants, la note moyenne aurait été plus appropriée.

8.3. Schémas en étoile, en flocon et en galaxie

Il existe différentes façons de mettre en œuvre OLAP. La stratégie MOLAP représente les valeurs au sein d'un tableau multidimensionnel. Dans notre exemple, cela donnerait[1] :

Ville	iPod	Vélo
Montréal	421.32	0
Paris	0	100.12

Cette approche peut s'avérer efficace quand les données sont denses : il y a peu de valeurs distinctes et peu de dimensions. En effet, les calculs se font directement sur un tableau de mesures. L'outil open source Palo utilise cette approche[2]. Malheureusement, il est rarement possible d'utiliser ce type d'approche dans les gros entrepôts de données. On utilise donc la stratégie ROLAP qui consiste à stocker la table de faits dans une base de données relationnelle. On bénéficie alors de la maturité des bases de données relationnelles. (Il existe aussi d'autres variantes, comme l'approche HOLAP qui est un hybride entre ROLAP et MOLAP.)

En ROLAP, on pourrait alors stocker la table de faits telle quelle. Mais, en pratique, on souhaite souvent normaliser la table. On crée donc une table pour chaque dimension. La stratégie de normalisation la plus simple est le **schéma en étoile**. Dans ce cas, on remplace tout simplement dans la table de faits chaque valeur par une clé (ou un identifiant) qui fait référence à une valeur dans la table de la dimension correspondante. Dans notre exemple, la table de faits est remplacée par une nouvelle table de faits et trois tables de dimension :

Ville ID	Produit ID	Mois ID	Ventes ($)
codeville1	codeproduit1	codemois1	421.32
codeville2	codeproduit2	codemois2	100.12

[1] On se limite à deux dimensions pour éviter d'avoir à faire un diagramme tridimensionnel.

[2] Voir http://www.palo.net/en/.

Ville ID	Ville
codeville1	Montréal
codeville2	Paris

Produit ID	Produit
codeproduit1	iPod
codeproduit2	Vélo

Mois ID	Mois
codemois1	Janvier
codemois2	Février

Parmi les avantages de cette normalisation, citons une certaine compression (puisque les valeurs ne sont pas inutilement répétées) et des mises à jour parfois plus faciles (lorsque l'on doit, par exemple, renommer un produit). L'inconvénient de la normalisation est la nécessité de procéder à des jointures pour traiter les requêtes.

Dans le cas où les valeurs sont hiérarchiques, on peut normaliser les tables de dimension à leur tour. On crée alors un schéma en flocon. Considérons par exemple une table de dimension hypothétique portant sur les lieux :

Lieu ID	Ville	Pays	Continent
codelieu1	Montréal	Canada	Amérique du Nord
codelieu2	Paris	France	Europe
codelieu3	Trois-Rivières	Canada	Amérique du Nord

On voit qu'on répète (inutilement) la description du pays Canada. Pour éviter cette duplication, on peut alors la remplacer par deux tables, comme ceci :

Lieu ID	Ville	Pays ID
codelieu1	Montréal	codepays1
codelieu2	Paris	codepays2
codelieu3	Trois-Rivières	codepays1

Pays ID	Pays	Continent
codepays1	Canada	Amérique du Nord
codepays2	France	Europe

Le schéma en flocon procure moins de bénéfices que le schéma en étoile, et il est en conséquence moins populaire. En effet, il ne compresse pas mieux les données que le schéma en étoile puisque les tables de dimension sont généralement petites en comparaison avec la table de faits, qui domine le coût de stockage. Par ailleurs, le schéma en flocon nous force à utiliser encore plus de jointures, ce qui peut réduire les performances.

Lorsqu'il y a plus d'une table de faits, on peut souvent réutiliser les mêmes tables de dimension. On dit alors qu'on a un schéma en galaxie.

8.4. Le cube de données

Jusqu'à présent, nous avons défini la table de faits et ses schémas, mais nous n'avons pas encore traité du problème de l'accélération des calculs. En effet, si la table de faits est volumineuse, sans indexation, le temps de réponse des requêtes risque d'être trop long. Une stratégie populaire d'accélération est le cube de données [2]. Celui consiste en un treillis de cuboïdes[3]. Chaque cuboïde correspond à un GROUP BY sur un sous-ensemble des dimensions. Avec notre exemple à trois dimensions, nous pourrions matérialiser un regroupement sur les dimensions Ville, Produit, Mois, Ville et Produit, Produit et Mois, Ville et Mois. Le résultat final est illustré à la Fig. 1. L'ensemble forme le cube OLAP. Une fois matérialisé, un cube OLAP permet d'effectuer plusieurs calculs très rapidement : il suffit de sélectionner le bon aggrégat.

[3]Un treillis est une relation d'ordre où pour chaque paire d'éléments a, b, il existe une borne supérieure et une borne inférieure à l'ensemble $\{a, b\}$. Pour plus de détails, voir l'article correspondant sur wikipédia.

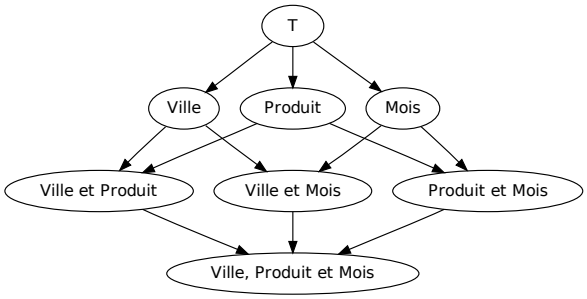

Figure 1. Treillis de cuboïdes formant un cube de données

Exemple 1: Exemple de cube de données

Un utilisateur souhaite obtenir un compte-rendu des ventes, regroupées par villes, pour le mois de février. Dans un tel cas, faire le calcul sur l'aggrégat Ville et Mois permettra de répondre à la requête beaucoup plus rapidement que si on devait travailler à partir de la table de faits. En effet, l'aggrégat Ville et Mois sera généralement beaucoup plus petit que la table de faits. Par exemple, s'il y a 100 produits différents vendus chaque mois dans chaque ville, le ratio pourrait être de 100:1. On peut donc imaginer une accélération par un facteur de 100 dans un tel scénario, ce qui est considérable.

La matérialisation du cube de données peut être difficile lorsqu'il y a beaucoup plus de trois dimensions. Le nombre d'aggrégats à calculer, et l'espace de stockage correspondant, deviennent trop grands. On procède donc le plus souvent à une matérialisation partielle du cube de données. Par exemple, on ne retiendra que les plus petits cuboïd (ceux portant sur peu de dimensions).

Un exemple de matérialisation partielle du cube de données est mise en œuvre dans MySQL avec l'instruction ROLLUP[4]. On matérialise d'abord le cuboïde du sommet ($\top$), puis le cuboïde correspondant à la première dimension (par exemple Ville), puis celui correspondant aux deux premières dimensions (Ville et Produit). Et ainsi de suite. La commande correspondante est :

[4]Voir http://dev.mysql.com/doc/refman/5.0/en/group-by-modifiers.html.

Table 1. Exemple de cube de données partiellement matérialisé avec l'instruction GROUP BY/WITH ROLLUP

Ville	Produit	Mois	Ventes ($)
Montréal	iPod	Janvier	321.00
Montréal	iPod	Janvier	120.32
Paris	Vélo	Février	100.12
Montréal	iPod	⊤	441.32
Paris	Vélo	⊤	100.12
Montréal	⊤	⊤	441.32
Paris	⊤	⊤	100.12
⊤	⊤	⊤	541.44

```
SELECT Ville, Produit, Mois, SUM(Ventes) FROM
tabledefaits GROUP BY Ville, Produit, Mois WITH
ROLLUP.
```

Le résultat est illustré par la Table 1. Dans la pratique, la valeur ⊤ est rendue par NULL dans MySQL. Contrairement au cube de données dont le volume peut croître de manière exponentielle avec le nombre de dimensions, la table du cube partiel matérialisé avec l'instruction ROLLUP croît de manière linéaire, dans le pire des cas, avec le nombre de dimensions.

8.5. Questions d'approfondissement

(a) Combien y a-t-il d'aggrégats distincts dans un cube portant sur 12 dimensions?
(b) Comment puis-je calculer l'aggrégat Ville et Mois à partir de la table de faits dans l'exemple de la Fig. 1 en SQL?
(c) Comment puis-je calculer efficacement l'aggrégat Ville dans l'exemple de la Fig. 1 en SQL?

8.6. Réponses suggérées

(a) On choisit donc souvent de ne matérialiser qu'une partie des aggrégats.
(b) Il y en a $2^{12} = 4096$. On voit que les cubes de données peuvent devenir trop coûteux dans la pratique s'il y a trop de dimensions.
SELECT Ville, Mois, SUM(Ventes) FROM tabledefaits GROUP BY Ville, Mois.

Soit aggregat1 l'aggrégat de la question précédente, il suffit alors de faire : SELECT Ville, SUM(Ventes) FROM aggregat1 GROUP BY Ville. Il serait inefficace de faire le calcul à partir de la table de faits. (c)

Votre avis compte !

Chers étudiants,
Si vous repérez une erreur dans ce document ou si vous avez une suggestion pour l'améliorer, nous vous invitons à remplir notre **formulaire anonyme**.
Votre contribution est précieuse pour nous !

Cliquez ici pour accéder au formulaire

BIBLIOGRAPHIE

1. E. F. Codd. Providing OLAP (on-line analytical processing) to user-analysis: an IT mandate. Technical report, E. F. Codd and Associates, 1993.
2. J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In *ICDE '96*, pages 152–159, 1996.
3. I. Kovacic, C. G. Schuetz, B. Neumayr, and M. Schrefl. OLAP Patterns: A pattern-based approach to multidimensional data analysis. *Data & knowledge engineering*, 138:101948, 2022.