Systèmes d'Information Décisionnels

Support de Cours Présenté Par

ABDALLAH BENSALLOUA Charef

Université Abdelhamid Ibn Badis - Mostaganem
Faculté des Sciences Exactes et de l'Informatique
Département de Mathématiques et Informatique

Juillet 2019

Avant-propos

Le cours de Systèmes d'Information Décisionnels est destiné aux étudiants de Master en Informatique, option Ingénierie des Systèmes d'Information (ISI), dans le Semestre 3. Les systèmes d'information décisionnels désignent l'ensemble des méthodes et des techniques permettant de collecter, consolider, modéliser et présenter les données d'une organisation dans le but d'aider les décideurs dans le processus de prise de décision. L'objectif de ce cours est de présenter aux étudiants les concepts et de les faire maitriser les compétences théoriques et techniques nécessaires pour effectuer l'analyse des situations, la conception et le déploiement de solutions de systèmes d'information décisionnel. Pour cela, la bonne exploitation du contenu de ce cours nécessite un pré-requis devant être maitrisé par les étudiants. Ça concerne en particulier : la conception et l'implémentation de Bases des Données Relationnelles et des notions sur la modélisation UML. Le contenu du cours peut se résumer dans les points suivant :

— SID : Définitions et Architecture

— Modélisation multidimensionnelle

— Dimension spatiale dans les Systèmes d'information décisionnels

— Qualité d'analyse dans les systèmes décisionnels

— Optimisation des requêtes dans les EDD

— Entrepôt de données NoSqQL

— Annexe A : panorama des différentes solutions logicielles pour la mise en oeuvre.

— Annexe B : Série d'exercices

à
Mon cher père et ma chère mère
à ma famille
Je dédie ce travail

Charef Abdallah Bensalloua

Sommaire

Liste des figures

Liste des tableaux

Liste des abréviations

BI : business intelligence

CI (IC) : Contrainte d'Intégrité (Integrity Constraint)

DM : Data Mart

ED (DW) : Entrepôts de données (Data Warehouse)

EDS (SDW) : Entrepôts de données Spatiales (Spatial Data Warehouse)

ETL : Extract - Transform - Load

NoSQL : Not-only SQL

MD : MultiDimensionnel

OLAP : On-Line Analytical Processing

OCL : Object Constraint Language

SDSS : Spatial Decision Support System

SID : Système d'Information Décisionnel

SIG : Système d'Information Géographique

SOLAP : Spatial On-Line Analytical Processing

SQL : Structured Query Language

UML : Unified Modeling Language

Introduction Générale

Le système d'information décisionnel est devenu très essentiel pour les entreprises du fait qu'il s'agit de moyen de gérer leurs performances et de définir leur stratégie de développement. Toutefois, l'introduction de systèmes décisionnels présente des caractéristiques spécifiques à ne pas négliger, à savoir la gestion des données historiques, notamment en termes de modèle de données, de référence, de qualité et de protection des données. Dans les entreprises, la prise de décision est encore souvent un domaine moins solide du système d'information. Ceci peut être traduit par le fait que les dirigeants ne le considèrent pas, depuis longtemps, au sommet de leurs priorités.

Le système d'information décisionnel doit permettre de développer la capacité de l'entreprise à penser et à agir. A ce titre, l'information nécessaire, élaborée par un processus de transformation itératif, est basée sur des données élémentaires relatives aux différents acteurs et événements (marketing, ventes, facturation, comptes clients, ... etc.). Pour que les systèmes décisionnels créent de la valeur et fournissent un avantage concurrentiel, il est nécessaire de veiller à ce qu'ils soient alignés sur les besoins de l'entreprise. Pour ce faire, il est nécessaire de définir un système d'information décisionnel et de mettre en place les moyens nécessaires à son implémentation et optimisation. Généralement, la planification et la création d'un système décisionnel doivent intégrer les contraintes nécessaires similaires à celles liées à la réalisation des autres composants majeurs du système d'information. Il s'agit donc, de se focaliser sur certaines inquiétudes, telles que les méthodes de développement, le choix de solutions technologiques et les conditions favorables pour l'exploitation de solutions.

Ce cours est présenté en six chapitres suivis par deux annexes organisés comme suit : i) le premier chapitre est consacré pour présenter les définitions et l'architecture des systèmes d'information décisionnels ; ii) dans le deuxième chapitre nous étudierons les différentes techniques de modélisation multidimensionnelle ; iii) dans le troisième chapitre, nous présenterons la dimension spatiale dans les systèmes décisionnels. iv) la qualité d'analyse dans les systèmes décisionnels et présentée dans le quatrième chapitre. v) Les techniques d'optimisation des requêtes dans les entrepôts de données sont présentées dans le cinquième chapitre. vi) le sixième chapitre est consacré pour la présentation des entrepôts de données NoSqQL. En Annexe A, un panorama des différentes solutions logicielles pour la mise en oeuvre est présenté. En fin, une série d'exercices est proposée dans Annexe B. Le présent document s'achève par une conclusion où nous récapitulons les apports du présent cours dans l'informatique décisionnelle.

Chapitre 1

SID : Définitions et Architecture

Dans ce chapitre, nous présentons des définitions relatives aux systèmes d'information décisionnels, ainsi que leurs architectures.

Sommaire

1.1 Introduction (Système d'information)

Le système d'information désigne l'ensemble des méthodes et moyens de collecte, de contrôle et de diffusion des informations nécessaires à l'activité d'une organisation. Sa fonction est de produire et de stocker les informations, puis de représenter l'activité du système d'exploitation (système opérationnel), puis de les mettre à la disposition du système de contrôle. Cependant, ces systèmes ne conviennent pas à l'analyse de données complexes et ne préparent pas les données pour la prise de décision. Pour assurer une plus grande réactivité et une plus grande compétitivité, les approches traditionnelles sont donc insuffisantes et les décideurs ont besoin de systèmes basés sur la technologie d'analyse en ligne, ce qui facilite leur processus de prise de décision, d'où la naissance de cette catégorie de système : Système d'Information Décisionnel.

1.2 Définitions

Définition1.1 *«Un système d'information décisionnel, SID, est un système qui réalise la collecte, la transformation des données brutes issues de sources de données et le stockage dans d'autres espaces ainsi que la caractérisation des données résumées en vue de faciliter le processus de prise de décision ».*

Définition1.2 *« Le système d'information décisionnel est un ensemble de données organisées de façon spécifiques, facilement accessibles et appropriées à la prise de décision [...].*

La finalité d'un système décisionnel est le pilotage d'entreprise. Les systèmes de gestion sont dédiés aux métiers de l'entreprise [...]. Les systèmes décisionnels sont dédiés au management de l'entreprise [...]. » (Goglin, 2001)

Le système d'information décisionnel doit remplir trois fonctions :

— L'extraction de données.

— Le stockage.

— La restitution des données sous une forme exploitable.

1.3 Enjeux du décisionnel

La prise de décisions stratégiques dans une organisation nécessite le recours et le croisement de multiples informations qui concernent tous les départements : production, ressources humaines, achats, ventes, marketing, service après-vente, maintenance,... etc. Or ces données sont généralement :

— éparpillées au sein des départements et non connectées entre elles

— hétérogènes dans leurs formats techniques et leurs organisations structurelles, voire leurs sémantiques

— implémentées pour l'action (par construction) et non pour l'analyse

— volatiles, au sens où leur mise à jour peut conduire à oublier des informations obsolètes.

Exemple 1.1

Un catalogue de produits sera conçu pour permettre de trouver facilement un produit en fonction de caractéristiques précises, de faire des mises à jour rapides et fiables, de gérer des stocks...

Mais un système décisionnel souhaitera :

— connaître l'organisation des produits selon certaines caractéristiques et regroupements qui ne sont pas forcément premiers dans la gestion quotidienne ;

— croiser le catalogue avec les ventes...

Résultat

L'enjeu des systèmes décisionnels est de donner accès aux données existantes dans l'organisation, sous une forme intégrée, afin de faciliter leur interrogation croisée et massive.

1.4 Exploitation des données

Les données agrégées dans un système décisionnel servent à trois grandes catégories d'usage :

— La production de rapports récurrents (reporting)

— L'exploration manuelle

— L'analyse de données (descriptive ou prédictive)

1.4.1 Reporting

Le principe du reporting est d'agréger et de synthétiser des données nombreuses et complexes sous forme d'indicateurs, de tableaux, de graphiques permettant d'en avoir une appréhension globale et simplifiée.
Le reporting s'appuie principalement sur les agrégats (GROUP BY en SQL par exemple) afin de faire apparaître des comptages, sommes ou moyennes en fonction de critères d'analyses.
Le reporting est généralement récurrent, c.à.d. le même rapport sera produit à intervalles réguliers pour contrôler les variations des indicateurs.

1.4.2 Exploration

Une autre exploitation de données en contexte décisionnel consiste à pouvoir explorer les données de façon peu dirigée (heuristique) afin de trouver des réponses à des questions que l'on ne s'est pas posées. L'idée générale est plutôt que les réponses aux premières questions que l'on se pose conduiront à se poser de nouvelles questions.
L'exploration de données s'appuie sur des outils permettant de manipuler et de visualiser les données grâce à des interfaces selon des requêtes dynamiquement produites par des utilisateurs experts du domaine.

1.4.3 Analyse de données

L'analyse de données est une technique qui permet de mettre en évidence des tendances des données ou corrélations entre les données non évidentes a priori.
Dans le cas de l'analyse descriptive, elle cherche une information statistique "cachée" que l'on ne connaît pas a priori.
L'approche prédictive consiste à réaliser un modèle statistique des corrélations entre les données à partir d'échantillons d'apprentissage, puis à appliquer le modèle à des données nouvelles pour prédire leur comportement, avec des raisonnements du type "si ... alors".
Une autre technique consiste à classifier des données (tel objet caractérisé par telles données appartient-il à telle classe ?). Les résultats sont généralement qualifiés par une probabilité d'occurrence.

1.5 Limites des systèmes décisionnels

1. **Rationalisation excessive et processus complexes**
   Les systèmes décisionnels produisent des indicateurs ou s'appuient sur des modèles dont l'objectif est de simplifier la réalité pour aider à la prise de décision. Mais la décision doit bien réintégrer des évaluations humaines qui la replacent dans sa réalité, qui est restée complexe.
   — Le modèle ou l'indicateur n'est pas la réalité, c'est une représentation.
   — La décision ne s'applique pas à une représentation, mais à la réalité.
2. **Sélectivité des données et organisations humaines**
   Les systèmes décisionnels s'appuient sur les données que l'on est en mesure de produire, mais ces données ne peuvent pas intégrer toutes les dimensions d'une organisation et de son environnement, en particulier les dimensions humaines.
   Or ces dimensions cachées au système décisionnel déterminent de nombreux fonctionnements de l'organisation, et doivent continuer d'être prises en compte.
3. **L'interprétation est humaine**
   Un système informatique produit des indicateurs qui nécessitent des interprétations humaines, expertes dans le cas du décisionnel. Un système informatique ne produit pas des directives qu'une organisation humaine doit suivre.
4. **L'erreur est informatique**
   Les résultats produits par les systèmes décisionnels sont le résultat de conceptions informatiques et mathématiques complexes, qui peuvent receler des erreurs. Par ailleurs, les résultats sont souvent statistiques, donc non déterministes. La possibilité d'une erreur ou d'une approximation inadaptée devra toujours être prise en compte dans les décisions.

1.6 Architecture d'un système décisionnel

Il n'existe pas de consensus sur une typologie d'architectures pour les SID. L'architecture type d'un système décisionnel basé sur l'entrepôt de données et l'OLAP peut être représentée dans la figure 1.1 (Naoum, 2006). Elle est divisée en quatre niveaux :

1.6.1 Sources de données

Les systèmes décisionnels peuvent avoir besoin de données issues de différentes sources et de différents formats de stockage. Ceci regroupe les fichiers texte, les rapports, les fichiers de base de données issus de différents systèmes de gestion de bases de données (SGBD), ... etc.

Pour ce fait ces données doivent passer par le processus d'Extraction, Transformation et chargement (ETL) pour qu'elles soient exploitables. Ce processus peut être résumé comme suit :

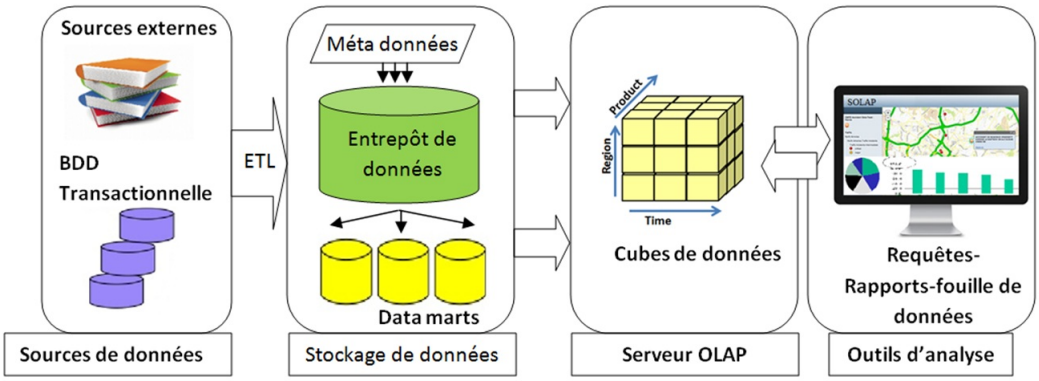


FIGURE 1.1 – Architecture d'un système décisionnel

1. *Extract (E)* : Dans la phase d'extraction seules les données destinées à l'exploitation pour l'analyse qui sont gardées en connectant aux différentes applications ou bases en production.
2. *Transform (T)* : Dans la phase de transformation, la mise au format des données, la fusion ou l'éclatement des informations et l'agrégation des données peuvent être effectuées.
3. *Load (L)* : Enfin, dans la phase de chargement, les informations sont stockées dans les entrepôts de données.

1.6.2 Entrepôts et magasins de données

1. ***Entrepôts de données***

   Un Entrepôt de Données (ED) est défini comme étant « une collection de données orientées sujet, intégrées, non volatiles, historisées disponibles pour le support du processus de prise de décision » (Inmon, 2005). Cette définition peut être expliquée comme suit :

   — Orientées sujet : les données sont organisées par sujet d'analyse suivant les besoins analytiques des entreprises.

   — Intégrées : les données hétérogènes venues de différentes sources feront l'objet d'une intégration dans un seul espace de stockage.

   — Non volatiles : les données ne sont ni modifiables, ni supprimable.

   — Historisées : du fait que les données sont non volatiles, des intervalles de temps les sont associés.

   Pour gérer l'ensemble des données entreposées, l'entrepôt de données doit disposer des "informations sur les données", à savoir les « métadonnées ». Celles-ci doivent permettre de répondre aux questions telles que : Comment extraire les données ? Quelles transformations effectuer ? . . . etc. Les métadonnées doivent également spécifier les droits d'accès et d'utilisation associés à ces données.

   Les données entreposées dans les entrepôts de données sont souvent stockées sous forme de vues matérialisées. Il s'agit de tables contenant les résultats de requêtes.

Elles améliorent l'exécution des requêtes en pré-calculant les opérations les plus coûteuses comme la jointure et l'agrégation.

2. ***Magasins de données***
   Un magasin de données (DM : data mart) est un sous-ensemble de l'entrepôt de données concernant un secteur particulier de l'entreprise. Il s'agit de données extraites, adaptées à une classe particulière de décideurs. Un magasin de données peut être considéré comme un petit entrepôt correspondant à un sujet précis. Ceci permet de réduire le temps de réponse aux requêtes (Naoum, 2006).

1.6.3 Serveurs OLAP

Dans cette section nous allons définir le concept de système OLAP et les opérateurs assurés par son serveur.

1. ***Système OLAP***
   Sur la base de ce type de structure (ED), les systèmes OLAP sont utilisés pour définir un modèle analytique transformant les données entreposées en informations d'aide à la décision.
   Selon (Codd et al., 1993 ; Kimball et Ross, 2002), un système OLAP est *« ... une catégorie d'outils qui permet exploration interactive suivant une approche multidimensionnelle à plusieurs niveaux d'agrégation ... »*.

   Ceci peut se réalisé en exploitant un entrepôt de données. Il a pour but d'évaluer l'activité et aide à la décision au sein d'une organisation.

   E. F. Codd (1993) définit un cahier des charges comprenant douze règles que doivent satisfaire les modèles OLAP :

   (a) la structure multidimensionnelle : le modèle OLAP est multidimensionnel,
   (b) la transparence : le serveur OLAP est transparent pour l'utilisateur,
   (c) l'accessibilité : l'utilisateur OLAP dispose de l'accessibilité à toutes les données nécessaires à ses analyses,
   (d) la stabilité : le système reste stable quelque soit le nombre de dimensions,
   (e) architecture client-serveur : le serveur OLAP s'intègre dans une architecture client serveur,
   (f) le dimensionnement : le dimensionnement est générique pour assurer les analyses,
   (g) la gestion complète : le serveur OLAP assure la gestion des données clairsemées,
   (h) les multiutilisateurs : le serveur OLAP offre un support multiutilisateur (gestion des mises à jour, intégrité, sécurité),
   (i) l'inter-dimension : le serveur OLAP permet la réalisation d'opérations inter dimensions sans restriction,

(j) l'aspect intuitif : le serveur OLAP permet une manipulation intuitive des données,

(k) la flexibilité : la souplesse de l'édition des rapports est intrinsèque au modèle,

(l) l'analyse sans limites : le nombre de dimensions et de niveaux d'agrégation possibles est suffisant pour autoriser les analyses les plus poussées.

A la différence des systèmes transactionnels (On-Line Transactional Processing (OLTP)) qui permettent la gestion des activités opérationnelles quotidiennes de l'entreprise, la technologie OLAP est proposée pour fournir un support pour la prise de décision. Le tableau 1.1 résume quelques différences entre ces deux types de systèmes.

TABLE 1.1 – Différences entre systèmes OLTP et OLAP (Boulil, 2012)

	OLTP	OLAP
Vocation (objectif)	Gestion des activités opérationnelles quotidiennes	évaluation de l'activité et aide à la décision
Utilisateurs	nombreux (des milliers), agents opérationnels	moins nombreux (des centaines), analystes et décideurs
Pattern d'utilisation	régulier, prédictible, et fréquent	irrégulier, non prédictible, et moins fréquent
Modèle de données	Normalisé (3FN) et optimisé pour les exigences de performance des traitements transactionnels	dénormalisé et optimisé pour les performances des traitements analytiques
Données	Opérationnelles orientées transaction, détaillées, courantes (un horizon temporel moyen de 60 à 90 jours), non redondantes et moins volumineuses	Orientées analyse, moins détaillées, historiques (un horizon temporel moyen de 5 à 10 ans), redondantes et volumineuses
Mode d'accès	lecture, rajout, modification et suppression	lecture et rajout
Type de traitements	Transactionnel : accès à des centaines d'enregistrements	analytique : agrégation et accès à des millions d'enregistrements
Type technologies	Optimisé pour le traitement transactionnel (temps de réponse et gestion d'accès concurrents)	Optimisé pour le traitement analytique (historisation et analyse en ligne)

2. ***Opérateurs OLAP***

Le serveur OLAP doit assurer un ensemble d'opérateurs permettant la navigation à travers les données entreposées. Voici les opérateurs typiques :

— ***Roll-up*** : cet opérateur permet d'agréger les valeurs de mesure en montant dans une hiérarchie de dimension.

— ***Drill-down*** : cet opérateur augmente le niveau de détail de la mesure en descendant dans la hiérarchie de dimension inversement à Roll-up,

— ***Slice*** : cet opérateur effectue une sélection des cellules de l'hypercube en utilisant une condition définie sur les membres d'une dimension. En résultat nous obtiendrons un sous hypercube.

— ***Dice*** :cet opérateur est utilisé pour réaliser une sélection en utilisant une condition définie sur deux dimensions ou plus.

— ***Projection*** : cet opérateur permet de sélectionner un sous-ensemble de mesures de l'hypercube.

— ***Pivot*** : cet opérateur effectue un pivotage des axes de l'hypercube.

1.6.4 Les outils d'analyse

L'outil d'analyse (client OLAP) représente l'élément le plus important pour l'utilisateur final qui permet d'exploiter les données stockées. C'est l'élément qui correspond à la partie visible du système par rapport au décideur.

En ce qui concerne les techniques de traitements des donnees, on distingue trois solutions :

— ***SQL*** : SQL est utilise pour effectuer les differents traitements sur les données. On réalise les opérations de traitement (forage vers le haut, vers le bas, etc.) en utilisant des requêtes en général très complexes et tres exigeantes en terme de ressources et de temps d'exécution. Il s'agit de l'alternative relationnelle (ROLAP).

— ***Serveur de traitement OLAP*** : il s'agit de l'approche la plus adaptée aux traitements de données. Un serveur, conjointement avec la base de données, est alors dédié a effectuer les différents traitements de données. Dans ce cas, les performances sont généralement très bonnes. Il s'agit de l'alternative multidimensionnelle pure (MOLAP).

— ***Client de traitement OLAP*** : un nombre limite de traitements OLAP se font sur le poste client de l'utilisateur. Il s'agit de l'alternative Bureau, en local (DOLAP).

En se basant sur ces deux critères, on peut alors différencier aisément les alternatives en combinant une technologie de stockage et une technique de traitement. Les combinaisons sont regroupées dans le Tableau 1.2.

Table 1.2 – Alternatives de mise en oeuvre OLAP

Stratégie	Stockage des données	Traitement des données
MOLAP	Base de données dimensionnelle	Serveur de traitement OLAP
ROLAP	Base de données relationnelle	SQL avancé
DOLAP	Fichier sur le poste client	Client de traitement OLAP
HOLAP	MOLAP pour données sommaires & ROLAP pour données détaillées	

1.7 Conclusion

L'analyse de données dans un système décisionnel repose sur des outils d'analyse statistiques dont le concept d'OLAP se base. Il représente la technologie qui offre une analyse multidimensionnelle avec un affichage des résultats sous forme de graphiques ou des tableaux.

Dans le chapitre suivant, nous allons explorer les différentes techniques de modélisation multidimensionnelles. Cette nomenclature se base sur le mode de stockage et de traitement des données. Le vocable exploitant l'acronyme OLAP ne cesse de croitre aujourd'hui (peut-être parfois parce qu'il est commercialement porteur). Il ne faut pas alors chercher à positionner ces termes les uns par rapport aux autres puisqu'ils référencent des concepts incomparables. Ainsi, on voit aujourd'hui apparaitre un terme comme JOLAP (Java OLAP) qui constitue en fait une API (Application Programming Interface) Java qui permet de se connecter a des applications et des serveurs OLAP, tentant de normaliser l'accès aux bases de données multidimensionnelles. On parle de SOLAP (Spatial OLAP) lorsqu'il s'agit de traiter des données spatiales. D'ailleurs, le SOLAP constitue à présent un domaine a part entière de recherche. On parle également d'OOLAP (Object OLAP), faisant référence à l'utilisation du paradigme objet. Néanmoins, à notre connaissance, cette technologie n'apparait pas dans les solutions commerciales.

Chapitre 2

Modélisation multidimensionnelle

Dans ce chapitre, nous allons présenter les principes de base de la Modélisation Multidimensionnelle.

Sommaire

2.1 Introduction

La Modélisation multidimensionnelle d'après (Teste, 2000), consiste à « *considérer un sujet analysé comme un point dans un espace à plusieurs dimensions. Les données sont organisées de manière à mettre en évidence le sujet analysé et les différentes perspectives de l'analyse*».

L'objectif d'une modélisation multidimensionnelle est de permettre aux analystes d'avoir une vision des données qui supporte et aide leur processus de prise de décision.

Les donnée sont alors, stockées dans une structure multidimensionnelle d'où à l'intersection de plusieurs dimensions se trouvent des valeurs nommées indicateurs ou variables. Ces valeurs sont calculées par le moteur OLAP à travers des opérations mathématiques ou statistiques plus ou moins complexes.

Nous distinguons trois types de modélisation multidimensionnelle à savoir *conceptuelle, logique et physique* décrites ci-dessous.

2.2 Modélisation conceptuelle

La modélisation conceptuelle fait référence aux concepts définis dans les travaux de Ravat (2005) et qui sont utilisés dans différents types de schémas de données (Ravat et al. 2005 ; Naoum, 2006 ; Kimball, 1996).

2.2.1 Concepts de base

Un ensemble de concepts sont utilisés pour concevoir un modèle multidimensionnel de données.

— ***Fait*** : c'est le concept qui modélise le sujet de l'analyse. Il est formé des informations de l'activité analysée.

— ***Mesure*** : C'est la valeur d'attribut quantitatif ou qualitatif qui évalue un fait. Voir Figure 2.1.

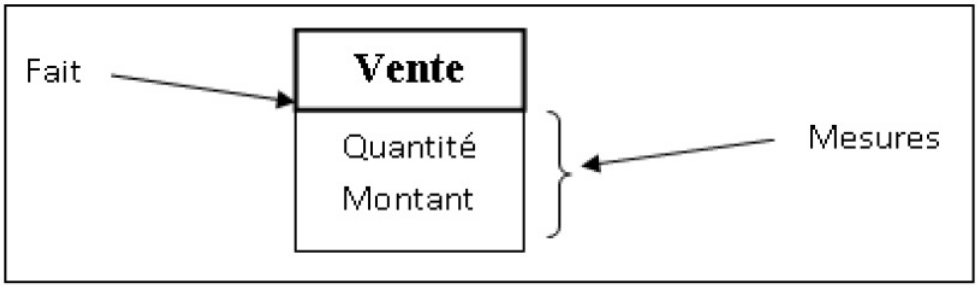


FIGURE 2.1 – Exemple de table de fait

Nous distinguons trois types de mesures :

1. *mesures de type « flux »* : mesures qui peuvent être sommées selon toutes les dimensions (e.g. "montant de vente") ;
2. *mesures de type « stock »* : mesures qui ne peuvent pas être sommées selon certaines dimensions (e.g. "population"), et
3. *mesures de type « valeur par unité »* : mesures qui ne peuvent jamais être sommées (e.g. "température").

— ***Dimension*** : ensemble de paramètres qui peuvent faire varier les mesures. Elle modélise une perspective de l'analyse voir Figure 2.2.

— ***Agrégation des mesures*** : c'est une opération qui permet de calculer différents indicateurs d'analyse à différents niveaux de détail. Elle utilise trois éléments clefs du modèle multidimensionnel à savoir la mesure, la fonction d'agrégation et la hiérarchie de dimension. Elle dépend des conditions d'agrégeabilité. Dans la littérature on distingue trois conditions d'agrégeabilité (Lenz et Shoshani, 1997),

1. *Disjonction* : pour permettre d'éviter le comptage en double des valeurs de mesure.
2. *Complétude* : pour le but d'éviter le problème d'agrégats incomplets.

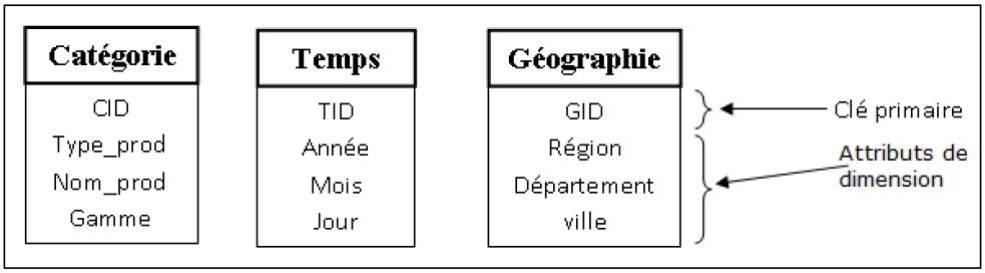


FIGURE 2.2 – Exemple de Tables de dimension

3. *Comptabilité de type* : cette condition vérifie que les natures des trois éléments, mesure, dimension et fonction d'agrégation, sont compatibles.

— ***Hiérarchie*** : Une hiérarchie permet d'organiser les membres d'une dimension selon leur niveau de détail. les hiérarchies peuvent être classées en deux catégories (Mazón et al, 2009) :

1. *Hiérarchies régulières* : qui satisfont les conditions de la complétude et la disjonction. Ici nous distinguons :

   - Hiérarchies strictes : spécifient pour chaque membre feuille un seul chemin d'agrégation vers le membre racine de la hiérarchie.
   - Hiérarchies onto : ici, tout membre non feuille possède au moins un membre fils et tous les membres feuille se trouvent au même niveau qui est le niveau d'agrégation feuille de la hiérarchie.
   - Hiérarchies covering : une hiérarchie est dite covering si elle ne présente pas de raccourcis ou de sauts dans les liens d'agrégation.

2. *Hiérarchies irrégulières* : ces hiérarchies ne satisfont pas l'une ou les deux conditions d'agrégeabilité citées ci-dessus (disjonction et complétude).

— ***Cube de données (Hypercube)*** : En plus de ces concepts, la notion du cube de données (hypercube) a été proposée par (Gray et al., 1997) et est défini comme *« un ensemble de données organisées selon des dimensions. On appelle mesure la valeur contenue dans une cellule du cube, associée aux valeurs prises sur les dimensions composant le cube ».*

La Figure 2.3 ci-dessous représente un cube à trois dimensions : produits, localisations et temps. Ici la mesure du cube est constituée des résultats des ventes des produits pour différentes villes à différents mois.

2.2.2 Schémas de données

Les concepts définis dans la section précédente peuvent être représentés en utilisant différents schémas à savoir : en étoile, en flocon de neige, en constellation et mixte.

1. **Modèle en étoile***(star schema)* : comme illustré dans la Figure 2.4, le schéma est constitué du fait central et des dimensions représentées de manière dénormalisée.

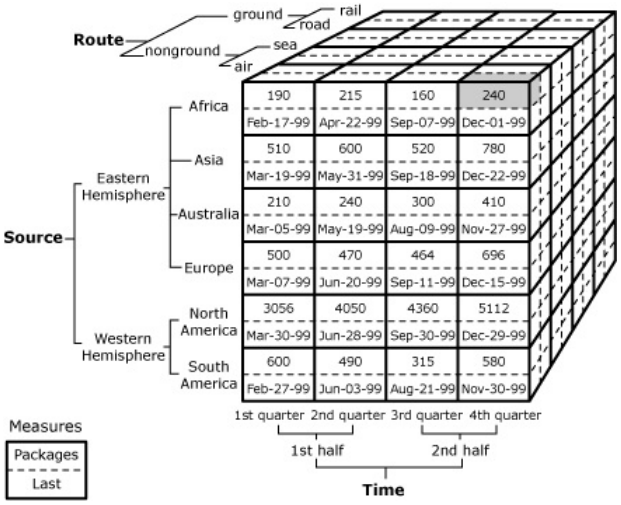


FIGURE 2.3 – Un cube de données à trois dimensions

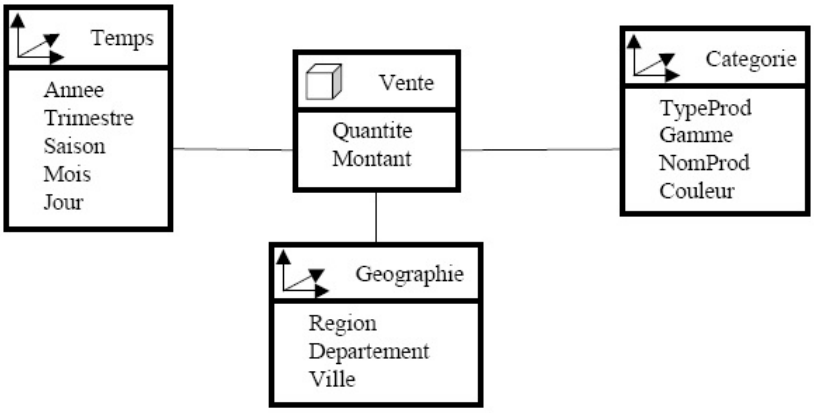


FIGURE 2.4 – Modèle en étoile

2. **Modèle en flocon** *(snowflake)* : dans ce schéma la table de faits est conservée et les dimensions sont éclatées conformément à sa hiérarchie. Ce modèle permet d'éviter le problème de redondance qu'on peut trouver dans le modèle en étoile. (voir Figure 2.5)
3. **Modèle en constellation** : cette technique consiste à fusionner plusieurs modèles en étoile. Il correspond donc à plusieurs tables de faits qui partagent des dimensions communes. (voir Figure 2.6)
4. **Modèle mixte** : cette technique consiste à fusionner plusieurs modèles en étoile et en flocon de neige.

2.3 Modélisation logique

La modélisation logique représente la technique du stockage physique des données selon le système de gestion de base de données (SGBD) utilisé ce qui donne naissance à trois modèle logiques :

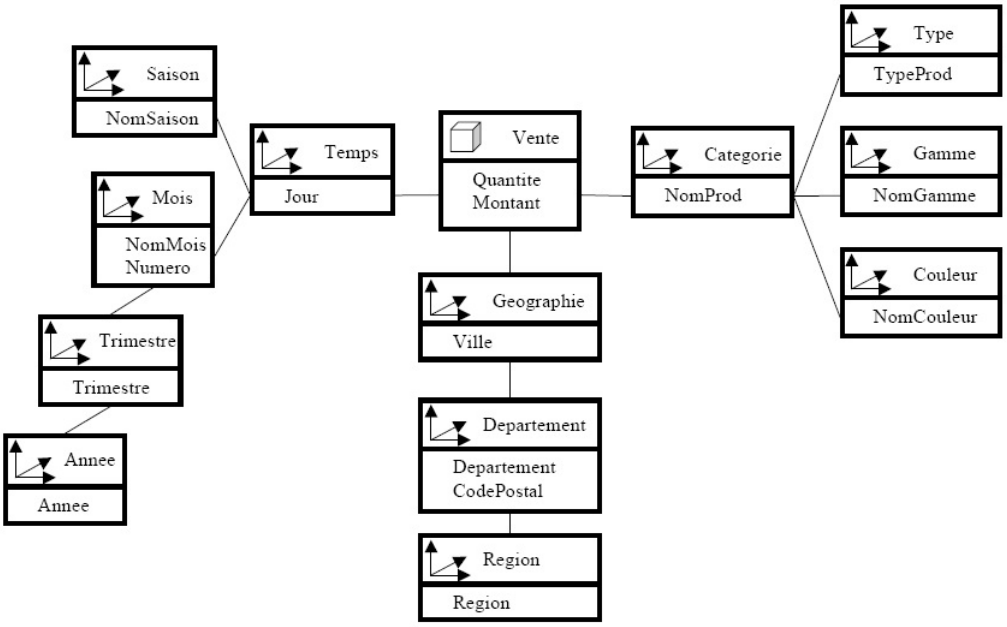


FIGURE 2.5 – Modèle en flocon de neige

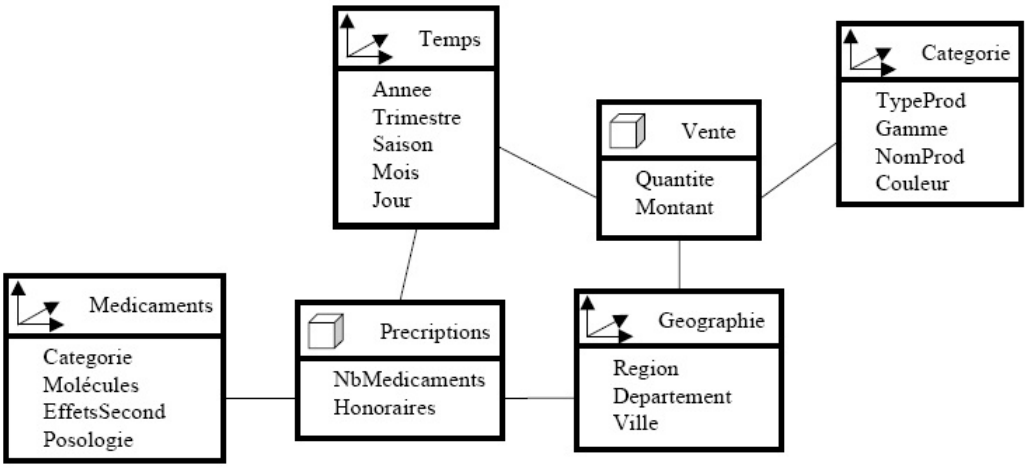


FIGURE 2.6 – Modèle en constellation

1. ***OLAP Relationnel (ROLAP)***

   Dans ce type d'implémentation un Système de Gestion de Bases de Données Relationnelles (SGBDR) est utilisé pour le stockage des données. Le serveur ROLAP permet de simuler la vue multidimensionnelle et de communiquer entre le SGBDR et les outils client OLAP. L'architecture ROLAP permet une grande souplesse dans la gestion de gros volumes de données et une meilleure administration des données. La figure 2.7 représente l'architecture ROLAP.

2. ***OLAP Multidimensionnel (MOLAP)***

   Dans ce type d'implémentation un Système de Gestion de Base de Données Multidimensionnelles (SGBDM) natifs est utilisé. Dans ce cas les données décisionnelles sont maintenues dans des structures multidimensionnelles. Les cellules des différents hy-

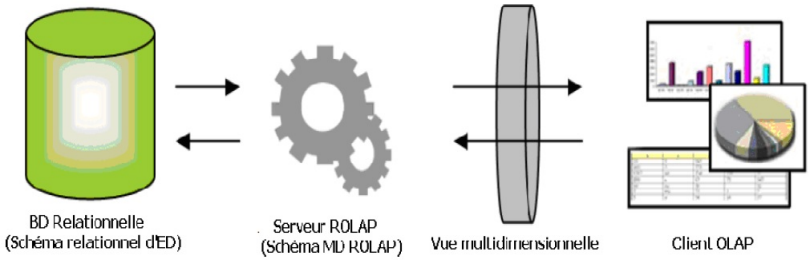


FIGURE 2.7 – Architecture ROLAP

percubes sont pré-calculées à partir de l'ED ou des bases sources et stockées dans ces structures. Par conséquent, Les opérateurs OLAP permettent d'explorer ces cellules d'une façon simple et rapide.

L'architecture MOLAP offre moins de performances que la précédente concernant l'administration de données à cause du pré-calcul des cellules. Voir Figure 2.8.

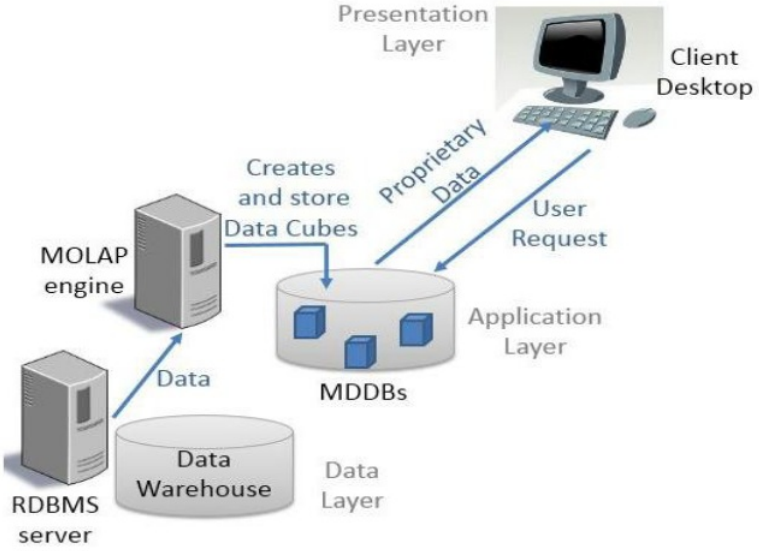


FIGURE 2.8 – Architecture MOLAP

3. ***OLAP hybride (HOLAP)***

   Ce type d'implémentation combine les deux technologies ROLAP et MOLAP. En effet, une partie des données dont les utilisateurs accèdent le plus fréquemment est stockée dans des structures multidimensionnelles gérées par un SGBDM. Une autre partie est stockée dans des structures relationnelles gérées par un SGBDR.

   Les systèmes HOLAP offrent un compromis des avantages et inconvénients des systèmes ROLAP et MOLAP. Voir Figure 2.9.

2.4 Modélisation physique

Après avoir effectué les deux premières modélisations conceptuelle et logique, nous devons passer à la modélisation physique de l'ED. Cette phase a pour objet d'appliquer

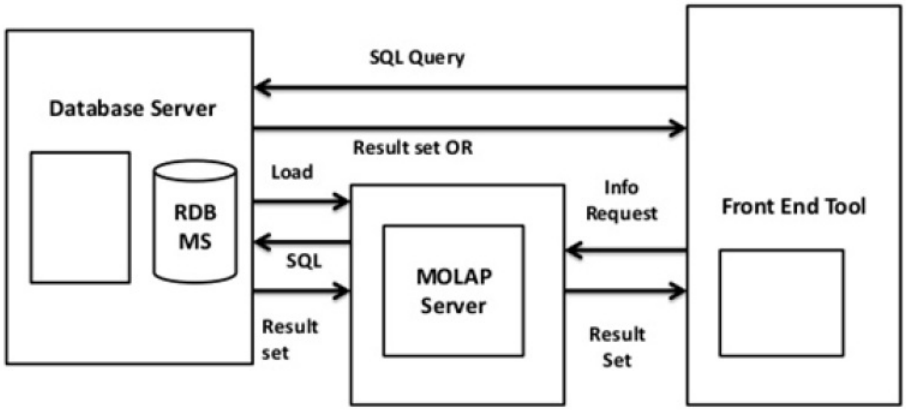


FIGURE 2.9 – Architecture MOLAP

techniques adéquates pour stocker efficacement les données afin d'en assurer un accès rapide. Ceci correspond à la traduction du modèle logique dans le langage de la plateforme d'implémentation (e.g. Oracle, MS SQLServer,... etc). (Mazon et Trujillo, 2008). Ensuite, des techniques d'optimisation peuvent être appliquées en vue d'optimiser les performances :

— La matérialisation de vues : certaines requêtes les plus utilisées sont calculées et stockées physiquement pour améliorer le temps de réponse. Néanmoins, certains problèmes peuvent se présentent tel que le choix des requêtes et leurs mise à jour.

— l'indexation : c'est une technique qui vise à assurer un accès direct aux données. On distingue les indexes binaires dans des tables simples ; et les indexes de jointure entre deux tables.

la fragmentation : c'est une technique qui consiste à diviser les tables volumineuses en plusieurs fragments pour accélérer l'accès aux données.

2.5 Limite des systèmes OLAP

Les systèmes OLAP ont montré leur efficacité notamment pour assurer l'analyse multidimensionnelle des données entreposée, non volatiles et à différents niveaux de granularité. Ils sont considérés comme outils d'aide à la décision. Néanmoins, comme nous l'avons indiqué dans la section précédente, ils présentent aussi des difficultés remarquables aux différents stades de modélisation notamment conceptuelle, logique et physique. En effet, les systèmes OLAP ne font pas la mise à jour des données, mais seulement l'archivage des versions. On se retrouve par fois devant des situations de gestion de Big data. Ceci est relatif à la périodicité de l'application du processus ETL. Les techniques et méthodes proposés pour résoudre ces problèmes sont efficaces dans quelques cas. Dans les autres cas, ces propositions font complexifier encore ces outils destinés au départ pour l'analyse rapide et facile des données. En plus, d'autres problèmes se posent relatives à la qualité des analyse effectuées par les utilisateurs. Cette qualité dépond de plusieurs facteurs notamment la consistance des données entreposées, la consistance des agrégations et la consistance de l'exploitation c'est-à-dire la formulation des requêtes.

2.6 Conclusion

Dans ce chapitre, nous avons exploré les différentes techniques de modélisation multidimensionnelle, notamment la modélisation conceptuelle, logique et physique des entrepôts de données. Néanmoins, il est à noter que les entrepôts de données traditionnelles ne permettent pas la gestion des données géographiques. Or, la grande partie des données sont géo-référencées portant des attributs géométriques.

A cet effet, le recours aux outils permettant la manipulation de ce type de données est incontournable, pour la mise en place de systèmes d'information décisionnels spatiaux. Nous parlons ici de l'ED Spatial et du spatial OLAP qui feront l'objet du chapitre suivant.

Chapitre 3

Dimension spatiale et Systèmes d'Information Décisionnels

Ce chapitre est consacré pour la présentation de la dimension spatiale dans les systèmes d'information décisionnels, notamment par la définition des Entrepôts de Données Spatiales et le Spatial-OLAP.

Sommaire

3.1 Introduction

L'utilisation intensive de nouvelles technologies et applications spatiales, a conduit à une production de gros volume de données qui ont un caractère spatial d'une importance remarquable. Ceci a conduit à l'intégration de ces données dans les processus d'aide à la décision en particulier dans l'analyse multidimensionnelle qui profite du concept de l'Entrepôts de Données Spatiales (EDS).

3.2 ED Spatial et Spatial-OLAP

Un Entrepôt de Données Spatiales est considéré comme est une collection de données spatiales et non spatiales orientées sujet, intégrées, variables dans le temps et non volatiles, utilisée pour supporter le processus décisionnel. (Stefanovic et al., 2000).

Dans le même contexte, les systèmes OLAP sont étendus en présence de la dimension spatiale, pour donner naissance au concept de Spatial OLAP (SOLAP). Ceci permet une exploration interactive des EDS en se basant sur un modèle spatio-multidimensionnelle. (Bédard et al., 2007).

3.2.1 Modèle spatio-multidimensionnel

Le modèle spatio-multidimensionnelle est basé sur un ensemble de concepts étendus du modèle multidimensionnelle. Voir figure 3.1.

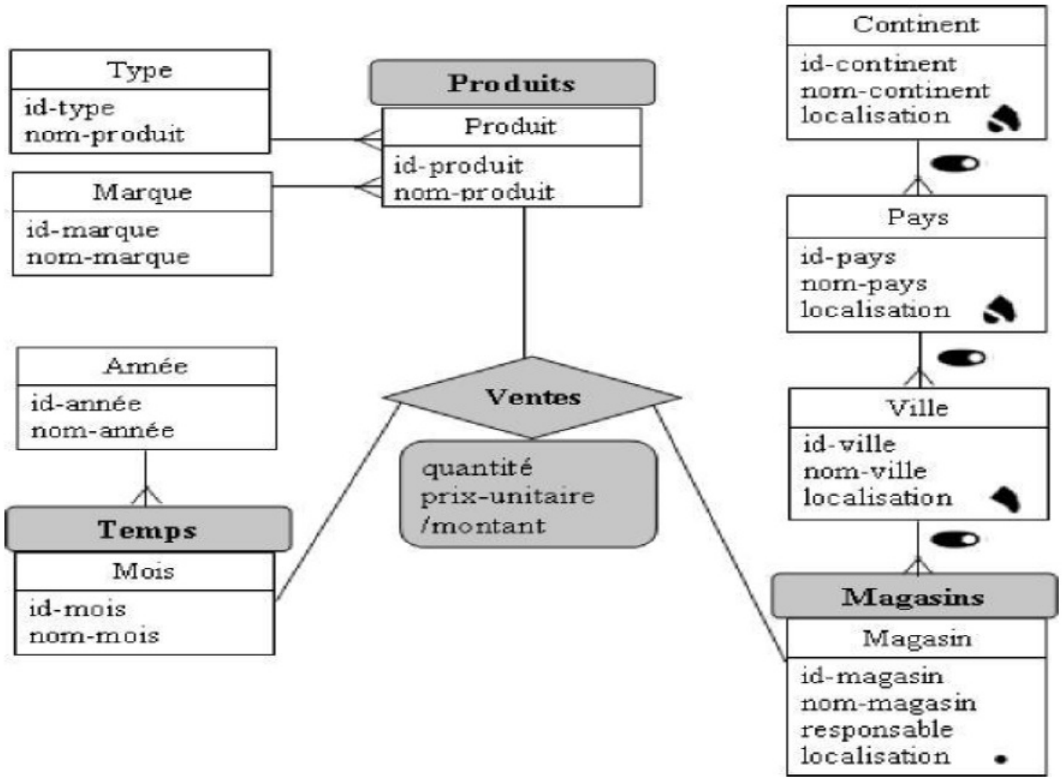


FIGURE 3.1 – Modèle spatio-multidimensionnel -"Analyse des ventes"

1. **Mesure spatiale** :

   Une mesure spatiale est une mesure dont le type de données est géométrique et/ou numérique. Par exemple la surface d'une zone géographique de forêt incendiée, la circonférence d'une zone de forêt protégée, la distance entre régions spatiales, etc. Trois types de mesures spatiales sont reconnus dans les différents travaux liés à la géomatique [1].

   A *mesure spatiale géométrique* : qui est définie par les entités géométriques représentant des objets spatiaux, tels que des "foyers d'incendie".

   B *mesure spatiales numériques* : qui sont des valeurs numériques résultant des opérateurs spatiaux telles que : distance ou superficie.

1. Géomatique : l'ensemble des méthodes qui permettent l'acquisition, la représentation et l'analyse des données à référence spatiale

C *mesure spatiale complète* : introduite par (Bédard et Han, 2008). Elle est spécifique aux cubes de données raster. Il s'agit d'une combinaison d'une mesure spatiale numérique et géométrique relative à une position de cellule raster et de sa valeur associée. (Rivest et al., 2005).

Exemple 3.1 : un exemple de mesure spatiale pour une "zone géographique incendié" est montré dans la Figure 3.2.

FIGURE 3.2 – Exemple de mesure spatiale

2. **Niveau d'agrégation spatial** :
   Un niveau d'agrégation spatial est un niveau d'agrégat : 4 représentant un ensemble de membres. Ceux-ci possèdent des attributs spatiaux qui leur permettent de représenter les données sous forme cartographique. (Bimonte, 2007 ; Malinowski et Zimányi, 2008).
   Un niveau d'agrégation spatial peut être géométrique ou non géométrique relatif à une description textuelle.
3. **Dimension spatiale** :
   La dimension spatiale peut être définie comme une dimension avec au moins un niveau d'agrégation spatial. Elle permet de représenter l'information spatiale des faits en axes d'analyse. (Bédard et Han, 2009).
4. **Hiérarchie de dimension spatiale** :
   Une hiérarchie de dimension spatiale est définie comme *une hiérarchie qui contient au moins un niveau d'agrégation spatial.*
5. **Fait spatial** :
   Un fait spatial est défini comme étant un fait contenant au moins une mesure spatiale ou un niveau d'agrégation spatial. (Salehi et al., 2010). Divers autres définitions peuvent exister dans la littérature notamment celles présenté par (Malinowski et Zimányi, 2008) et celle de (Boulil, 2012).
6. **Hypercube spatial** :
   Un hypercube (ou data cube) spatial peut être défini comme un hypercube possédant au moins une mesure spatiale ou une dimension spatiale. (Salehi et al., 2010).

Il existe aussi d'autres définitions dans la littérature. Dans cette thèse nous allons considérer qu'un hypercube doit posséder au moins une mesure spatiale et une dimension spatiale pour être qualifié de spatial.

7. **Entrepôt de données Spatial (EDS)** :
   Un Entrepôt de données Spatial est un modèle multidimensionnel contenant au moins un hypercube spatial. (Boulil, 2012). Voir Figure 3.3.

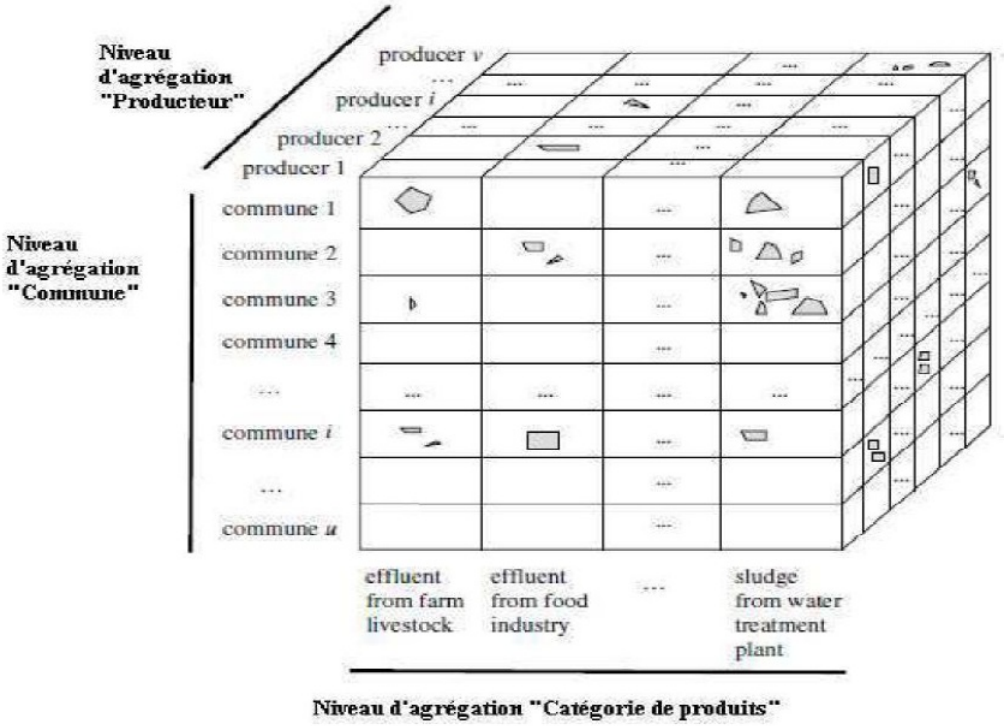


FIGURE 3.3 – Exemple d'instance d'hypercube spatiale

3.2.2 Opérateurs de navigation SOLAP

Pour permettre l'exploration des hypercubes de données spatiales, un ensemble d'opérateurs de navigation doivent être définis. Ces opérateurs sont l'extension des opérateurs OLAP. (Bédard et al., 2007) :

— **Spatial Roll-up :** Cet opérateur est utilisé pour la navigation dans une hiérarchie de dimension d'un niveau d'agrégation spatial vers un autre moins détaillé.

— **Spatial Drill-down :** Cet opérateur est utilisé pour la navigation dans une hiérarchie de dimension d'un niveau d'agrégation spatial vers un autre moins détaillé.

— **Spatial Slice :** cet opérateur est utilisé pour la sélection d'un sous-ensemble des cellules de l'hypercube spatial par l'application d'un prédicat spatial sur les membres d'une dimension spatiale.

— **Dice spatial :** cet opérateur est utilisé pour la sélection d'un sous-ensemble des données par l'application des prédicats spatiaux à des membres spatiaux de plusieurs dimensions spatiales. (Boulil, 2012)

3.2.3 Fonctions d'Agrégation dans le SOLAP

L'application des fonctions d'agrégation spatiale sur des mesures spatiales est une opération fondamentale pour l'analyse spatio-multidimensionnelle. Ce qui permet de retourner un ensemble d'objets géométriques simple ou complexes. Pour garantir la pertinence de cette application, nous devons vérifier les paramètres suivant (Pedersen et Tryfona, 2001) :

— L'additivité de la mesure (stock, flux ou valeur par unité),

— La structuration de la hiérarchie (stricte ou non stricte),

— Le type de fonction d'agrégation,

— La distributivité de la fonction d'agrégation,

— Les relations spatiales topologiques entre les membres spatiaux.

Au sein du SOLAP, nous pouvons appliquer différentes fonctions d'agrégation applicables dans OLAP traditionnel. En plus, concernant les données géométriques, des fonctions d'agrégation spatiale de type géométrique peuvent être appliquées telles que l'intersection, l'union, centroïde ... etc.

3.3 Architecture d'un système décisionnel basée sur SOLAP

En plus des données sources, l'architecture d'un système décisionnel basée sur SOLAP peut être résumée en quatre couches logicielles : ETL spatial, ED spatiales, Serveur SOLAP et Client SOLAP. Voir figure 3.4.

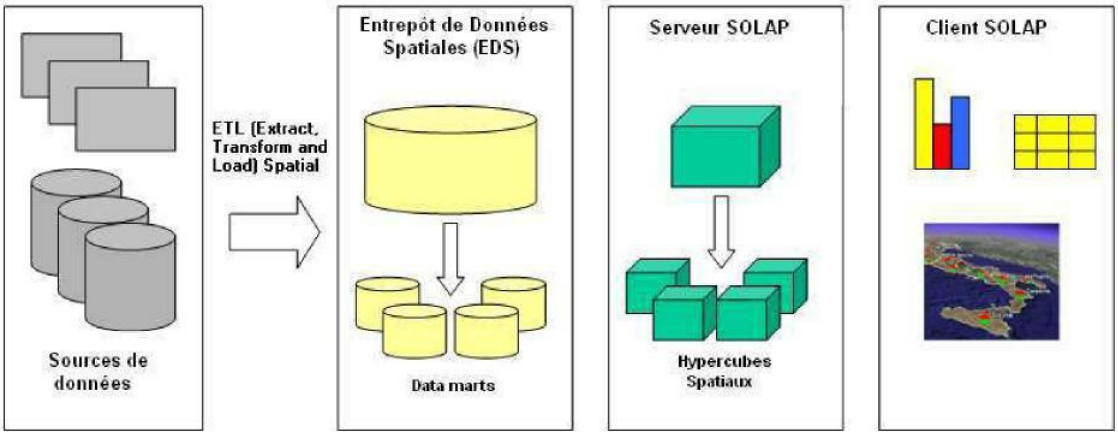


FIGURE 3.4 – Architecture typique d'un système SOLAP

1. **Couche ETL spatial :** L'intégration de données spatiales dans l'EDS nécessite des outils ETL (Extract-Transform-Load) spatiaux (par exemple Spatial Data Integrator [2]), qui permettent le traitement et la transformation des données spatiales pour résoudre des problèmes d'hétérogénéité de données obtenues de différents systèmes de référence spatiale à différentes échelles géographiques, etc. (Xi-Qian et al.,

2. http ://www.spatialdataintegrator.com

2004 ; Bédard et Han, 2009). Cette opération peut être effectuée par changement de projection cartographique, application du géocodage, . . . etc.

2. **Couche EDS :** Cette couche se base sur un modèle spatio-multidimensionnel. Elle englobe l'entrepôt de données spatiales et non spatiale, les métadonnées et éventuellement magasins de données. Elle permet d'historiser et de gérer de très gros volumes de données nécessaires à l'analyse SOLAP tels qu'Oracle Spatial et SQLServer.
3. **Couche Serveur SOLAP :** Grace à cette couche nous pouvons calculer l'hypercube et implémenter les opérateurs de navigation pour permettre l'exploration spatio-multidimensionnelle dans le SOLAP.
4. **Couche Client SOLAP :** Cette couche permet de fournir aux utilisateurs les interfaces de navigation interactive facile et rapide au sein de l'hypercube de données spatiales et non spatiales.

Ces interfaces doivent fournir différents modes d'affichages interactifs (histogrammes, tables de pivot et cartes) d'une part ; et d'appliquer facilement les différents opérateurs de navigation d'une autre part. Cette couche permet l'exploitation du produit final de l'analyse spatio-multidimensionnelle. Voir Figure 3.5.

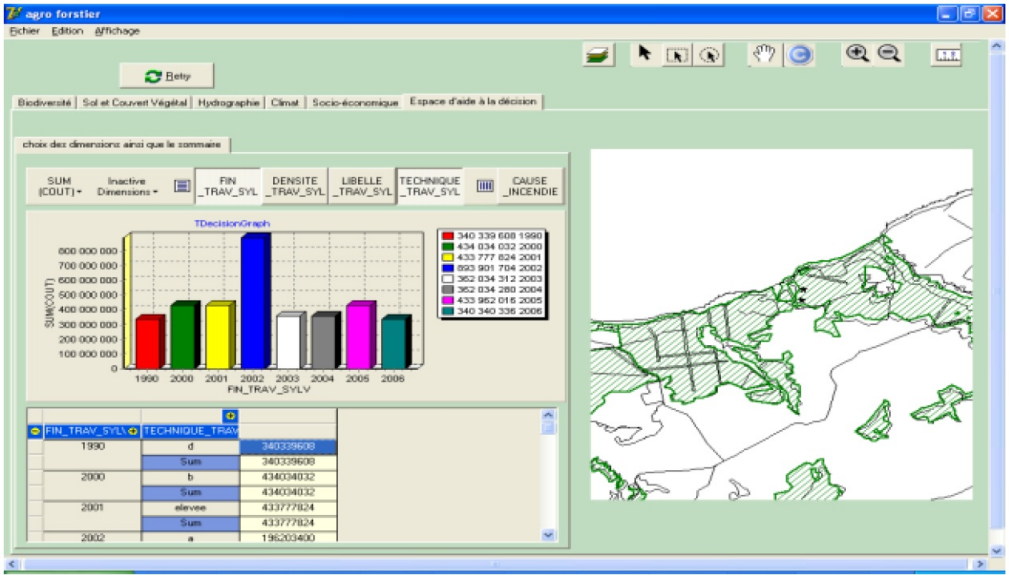


FIGURE 3.5 – Exemple de Client SOLAP

3.3.1 Modes d'intégration SIG-OLAP

Dans un système SOLAP, nous cherchons à utiliser les techniques d'analyse multi-dimensionnelle des systèmes OLAP en intégration des fonctionnalités de représentation cartographiques et l'analyse spatiale des Systèmes d'Information Géographique. Il existe trois familles de solutions technologiques pour le développement et l'implantation d'une application SOLAP :

1. **OLAP dominant** Ce type d'application est développé autour d'un serveur OLAP. Les fonctions OLAP sont dominantes alors que les fonctions SIG sont minimales.

Ex. zoom, déplacement, sélection, gestion des couches actives. Parfois, forage spatial minimal. Il a l'avantage de supporter l'exploration et la visualisation OLAP en utilisant toutes les capacités d'un serveur OLAP. Néanmoins, il fournit des fonctionnalités d'exploration et de visualisation cartographiques simples.

2. **SIG dominant** Ce type d'application est développé autour d'un SIG et d'un SGBD. Il fournit toutes les fonctions de la cartographie et de l'analyse spatiale des SIG. Néanmoins il ne permet pas toutes les capacités d'un serveur OLAP.
3. **SOLAP intégré** Ce type d'application permet un haut niveau de fonctionnalités pour les vues et données spatiales et non-spatiales. Il intègre et synchronise toutes les fonctions OLAP et SIG.

3.3.2 Modélisation spatio-multidimensionnelle

Dans la littérature, beaucoup de travaux ont entamé la problématique liée à la modélisation spatio-multidimensionnelle. En effet, ils visent définir de nouveaux concepts par l'extension des concepts classiques de la modélisation multidimensionnelle traditionnelle.

Ces travaux peuvent être classés en deux catégories notamment : i) celle basée sur des standards tels que Unified Modeling Language (UML) et Entité-Relation (ER) ; et ii) celle basée sur des formalismes ad hoc. Pour plus de détails sur ces aspects nous pouvons nous référer à (Kimball et Ross, 2002 ; Malinowski et Zimányi, 2008 ; Salehi, 2009 ; da Silva et al., 2010).

Modèles basés sur des standards

1. **Modèles UML**

   Un profil UML pour la modélisation multidimensionnelle est présenté dans (Lujan-Mora et al., 2006). Ce profil est étendu par les auteurs dans (Glorio et Trujillo, 2008), en définissant de nouveaux stéréotypes notamment « SpatialMeasure » et « SpatialLevel » pour représenter les mesures spatiales et les niveaux d'agrégation spatiale consécutive.

   Dans ce modèle les hypercubes peuvent partager des dimensions, des hiérarchies et des niveaux d'agrégation. En effet, l'utilisation des paquetages UML permet l'organisation des éléments multidimensionnels selon des niveaux de détail, ce qui permet une lecture simple des modèles complexes.

   Dans (Pinet et Schneider, 2010), les auteurs étendent le modèle UML présenté dans (Pinet et Schneider, 2009) par le concept de « mesure spatiale » qui signifie un attribut UML dont le type de données est géométrique ; et le concept de « classe identifiée spatiale » qui signifie une classe ayant un attribut géométrique. De plus les auteurs unifient les représentations des faits et des niveaux d'agrégation sous en « classes identifiées spatiales » ; et les représentations des relations d'agrégation et des relations (fait-dimension) en « associations d'agrégation ».

   Dans le travail présenté dans (Pinet et al., 2010), les auteurs propose un profil UML pour SOLAP, dans lequel ils représentent les hypercubes par des paquetages

UML, avec les niveaux d'agrégation représentés par classes reliés par des relations d'agrégation UML. Les mesures sont regroupées dans un stéréotypée « Measure » ; alors que les dimensions sont spécialisées en thématiques, spatiales et temporelles.

Pour le meilleur de nos connaissances (Boulil, 2012) a fait une transition remarquable dans le domaine de la modélisation spatio-multidimensionnelle dans ces dernières années. En effet, dans (Boulil et al., 2015) les auteurs proposent un nouveau profil UML pour modéliser des EDS et des agrégations spatiales complexes. Ils proposent aussi un outil pour la mise en oeuvre automatique de cubes de données spatiales en utilisant ce profil. Cette solution permet de générer différentes représentations logiques du modèle de l'EDS (schéma en étoile et schéma en flocon de neige) et d'implémenter des indicateurs d'analyse SOLAP complexes en utilisant le langage MDX (MultiDimensional eXpression).

2. **Modèles ER**

   Dans la littérature, nous trouvons peu de travaux qui ont entammé la modélisation spatio-multidimensionnelle de type (Entité-Relation). Nous pouvons citer à titre d'exemple le modèle présenté dans (Malinowski et Zimányi, 2008) qui étende le modèle multidimensionnel présenté (Malinowski et Zimányi, 2006) pour proposer un modèle spatio-multidimensionnel ER. Ce dernier fournit un support spatial pour les éléments multidimensionnels (mesures, faits, niveaux d'agrégation, hiérarchies et dimensions). En effet, la mesure spatiale est définie comme une mesure dont le type de données est géométrique.

Modèles ad hoc

La plupart des propositions de la modélisation spatio-multidimensionnelle traitent des problématiques particulières. Par exemple dans (Jensen et al., 2004) les auteurs traitent le problème de comptage en double des valeurs de mesure dans les hiérarchies de type non strictes. Il s'agit d'une extension du modèle multidimensionnel proposé dans (Pedersen et al., 2001). Dans (Bimonte et al., 2005), les auteurs traitent le problème de prise en compte de la dépendance entre les agrégations numérique et spatiale. Les auteurs dans (Damiani et Spaccapietra, 2006), proposent une solution pour représenter les mesures spatiales à différentes échelles géographiques.

Dans la plus part des modèles de type ad hoc, ne fournissent pas de notations graphiques ce qui rend leur compréhension et utilisation très difficile. (Boulil, 2012).

3.4 Limite des systèmes SOLAP

Comme tout type d'outils, les systèmes SOLAP présentent quelques limites. Ces limites sont des difficultés qui résident dans plusieurs niveaux. En effet, lors de l'exploitation des systèmes SOLAP, quelques problèmes liés à la consistance des résultats, sont posés. Ils s'agissent de problèmes liés à la qualité des données entreposées, la qualité d'agrégation et à la qualité de l'exploration effectué par les utilisateurs qui formulent des requêtes invalides.

Des études ont proposé de résoudre ces problèmes à un niveau bien avancé notamment au stade de modélisation. Néanmoins, ces efforts, ont contribué à la complexification de la conception des entrepôts de données spatiales. En plus, l'application de ces techniques, ajoute beaucoup d'informations artificielles, qui rendent les résultats d'analyse loin de la réalité.

3.5 Conclusion

Dans ce chapitre nous avons présenté la composante spatiale des données entreposées dans le SID. Cela a donné naissance aux nouveaux concepts d'EDS et SOLAP. Dans le chapitre suivant nous allons traiter les solutions permettant d'améliorer la qualité de l'analyse (spatio)-multidimensionnelle.

Chapitre 4

Qualité d'analyse dans les systèmes décisionnels

Dans ce chapitre, nous allons discuter la qualité des données et les contraintes d'intégrité (CI) dans l'analyse spatio-multidimensionnelle et les Entrepôts de Données.

Sommaire

4.1 Introduction

L'analyse multidimensionnelle à travers les systèmes, peut se faire en se basant sur l'entrepôt de données spatiales. Pour ce fait, en exploitant la structure multidimensionnelle, l'utilisateur cherche à explorer les mesures stockées ou calculées selon les différentes dimensions, en appliquant des règles d'agrégation.

4.2 Qualité d'analyse multidimensionnelle

La qualité d'analyse, dépond de trois paramètres notamment : les données, la façon d'agrégation des mesures et la façon de l'exploration des données.

4.2.1 Selon la qualité de données

Pour déterminer la qualité d'analyse (S)OLAP vis-à-vis les données (spatiales) entreposées dans l'ED(S), plusieurs études ont traité la façon et les critères dont on doit examiner. En général, ces études sont basées sur deux points de vue.

Le premier, concerne la qualité de production des données telles que la précision, l'exhaustivité, la cohérence logique, ... etc.

Le deuxième point de vue, concerne l'adéquation des données pour l'utilisation, c'est-à-dire, est-ce qu'elles répondent bien aux besoins des utilisateurs.

4.2.2 Selon la façon d'agrégation des mesures

La façon dont les données sont agrégées représente un facteur d'une importance irréprochable pour assurer une bonne analyse dans les structures multidimensionnelles.

En effet, en plus de la qualité des données dans l'ED(S), les indicateurs d'aide à la décision, dépendent aussi de la manière dont les hiérarchies, les fonctions d'agrégation et le type de mesures sont utilisés dans la manipulation du système (S)OLAP.

Ça correspond à la possibilité d'effectuer des agrégations correctes lors du passage d'un niveau de détail à un autre. (Boulil, 2012)

Pour ce fait, dans (Lenz et Shoshani, 1997), les auteurs définissent trois conditions notamment : la disjonction, la complétude et la compatibilité de types.

a) *La condition de disjonction* : elle permet de vérifier le comptage en double des mesures ; et que les membres de dimension se trouvent aux mêmes niveaux.

b) *La condition de complétude* : elle permet de vérifier que tous membres de dimension et instances de table de faits soient présents pour éviter les agrégats incomplets.

c) *La condition de comptabilité de types* : elle permet de vérifier que l'application de la fonction d'agrégation à une mesure à travers toute la dimension soit valide. Pour cela on doit considérer le type de la mesure, le type de dimension et le type de fonction d'agrégation.

4.2.3 Selon l'exploration des données

L'exploration interactive des données représente la propriété caractéristique d'une application (S)OLAP. Cette exploration est multidimensionnelle et différents niveaux de détail.

Le (S)OLAP est donc utilisé pour des fins d'aide à la décision, par différents utilisateurs par fois décideurs, qui peuvent ignorer les caractéristiques de l'analyse multidimensionnelle, et de la consistance des résultats obtenus suite à leurs requêtes.

En effet, ces requêtes multidimensionnelles sont formulées par des combinaisons d'indicateurs et des membres de dimension de façon invalide. (Boulil et al., 2012a).

Par exemple, une requête invalide peut être formulé par un utilisateur "Quelles sont les surfaces reboisées à l'union soviétique entre les années 2000 et 2010 ?".

Cette requête est correcte syntaxiquement, et les agrégats sont correctement appliqués. Néanmoins, l'exécution de cette requête retourne des résultats nuls. Ça peut être compris par l'utilisateur qu'il n'avait pas de reboisement à l'union soviétique entre ces deux dates ou bien il y a une absence de donnée dans l'EDS. Or, l'union soviétique n'existe plus après 1991.

La formulation de cette requête va conduire à des interprétations erronées qui peuvent diriger à prendre de mauvaises décisions.

Ce problème se pose dans la plus part des applications (S)OLAP. Il correspond à une incohérence sémantique des requêtes des utilisateurs.

Il est pratiquement difficile de trouver des mécanismes automatiques qui examinent le sens des requêtes et leurs résultats.

Néanmoins, des approches ont été proposées pour éviter ces problèmes. Elles sont basées généralement sur l'interdiction de l'exécution, ou bien la proposition de la reformulation de ces requêtes. (Boulil et al., 2012, Boulil, 2012)

4.3 Contraintes d'intégrité dans les cubes de données (spatiales)

Dans les applications de type (S)OLAP, la qualité de décision dépend de la qualité des données entreposées et de la façon dont elles sont explorées. Dans ce contexte, les contraintes d'intégrité (IC) sont définies dans (Boulil et al., 2012) comme des assertions, généralement définies dans le modèle conceptuel, qui visent à empêcher l'apparition de données incorrectes dans l'entrepôt de données. Ici, une CI peut être utilisée pour :

1. identifier les données erronées stockées,
2. identifier les requêtes analytiques incorrectes et
3. définir des règles d'agrégation correctes.

Dans un (S)OLAP, une spécification de contrainte d'intégrité doit contenir une sémantique spécifique prenant en charge les concepts liés au modèle multidimensionnel.

Au niveau conceptuel, les CI des EDS définissent les conditions que les données, les métadonnées ou les requêtes analytiques sont censées satisfaire.

Dans (Boulil et al., 2011), les auteurs proposent que la définition des CI au niveau conceptuel permette de considérer les problèmes de qualité dans les premiers stades du développement. En effet, ils proposent deux nouvelles classifications des CI pour les EDS :

1. La première classification catégorise les CI en fonction des éléments multidimensionnels qu'elles impliquent.
2. La deuxième classification classe les CI en fonction des niveaux de mise en oeuvre dans l'architecture (S)OLAP.

En effet, l'expression conceptuelle de toute CI dépend des concepts EDS concernés ; alors que sa traduction physique dépend du niveau SOLAP où il est implémenté.

En fait, il révèle que les CI peuvent être spécifiés en utilisant OCL et Spatial OCL au niveau conceptuel. Cependant, Spatial OCL et UML ne permettent pas de spécifier tous les CI proposés dans cette classification. Cela concerne en particulier les IC des métadonnées, les CI d'agrégation et les CI de visualisation. L'auteur dans (Boulil, 2012), propose une extension de la classification précédente en introduisant une nouvelle classe liée à la requête d'analyse. Ceci vise à éviter les interprétations erronées des résultats. Voir la Figure 4.1.

La solution puisse être au stade de modélisation, en ajoutant dans le profil UML :

1. plus de valeurs étiquetées (tagged values) représentant les métadonnées principales,

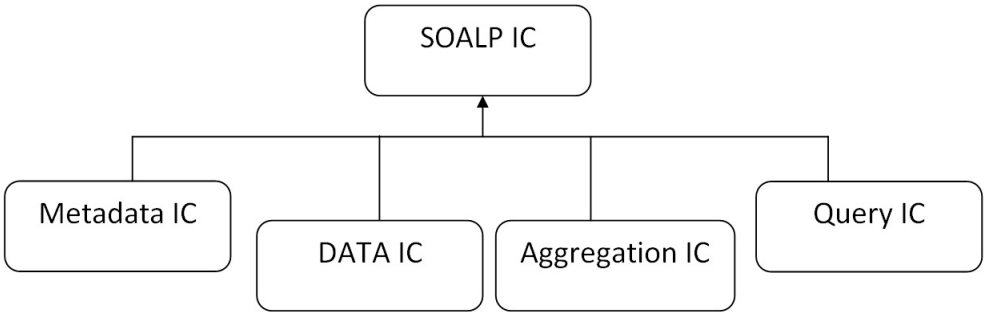


FIGURE 4.1 – Classification des CI SOLAP

2. plus de stéréotypes permettant aux utilisateurs de définir leurs propres CI d'agrégation,
3. formalisation des profils des utilisateurs pour spécifier les CI de visualisation en fonction de leurs préférences.

Etude de cas :

Soit un entrepôt de données dont la mesure est la valeur de la température.

En utilisant cet ED, les décideurs peuvent répondre à des questions OLAP telles que :

— "Quelle est la température minimale par année et par pays ?"

— "Quelle est la température moyenne par mois et par pays ?".

Pour répondre à ces questions, les décideurs utilisez les fonctions d'agrégation (min et moyenne) pour agréger les valeurs de température.

Exemple 4.1 : «la géométrie de chaque ville doit être topologique inclus dans la géométrie de sa région »ou

Exemple 4.2 : « aucun fait (par exemple, les valeurs de température) ne devrait exister pour l'URSS après le 26 décembre 1991 ».

Exemple 4.3 : dans un ED spatial, les membres spatiaux et les mesures doivent être définis avec la même échelle géographique.

Ces contraintes peuvent être définies pour tous les éléments de l'ED(S), tels que les faits, les membres, etc.

La CI d'agrégation garantit des agrégations correctes et significatives de mesures.

Les contraintes sémantiques concernent en particulier le problème de l'applicabilité des fonctions agrégées aux mesures en fonction de la nature sémantique et du type de mesures, des fonctions agrégées et des dimensions. Par exemple :

Exemple 4.4 : «La somme des valeurs de température n'a pas de sens dans certaines applications».

Les contraintes de schéma sont des conditions qui doivent être satisfaites par les hiérarchies de dimensions et les relations dimension-fait pour éviter les doubles comptages et les agrégats incomplets.

Exemple 4.5 : les dimensions et les faits devraient être liés par des relations un-à-plusieurs.

CI de requête fait référence à des conditions garantissant la validité des requêtes (S)OLAP, en ce sens que leurs résultats ne sont pas toujours vides afin d'éviter les problèmes d'interprétation erronée.

Exemple 4.6 : la requête SOLAP "Quelles sont les températures moyennes en URSS en 2010 ?" Renvoie un résultat vide, aucune valeur de température n'étant stockée pour l'URSS après le 26 décembre 1991. Même si ce CI est implémenté en tant que CI de données, les outils OLAP classiques permettent aux décideurs de formuler cette requête en combinant ces deux membres (URSS et 2010) renvoyant une valeur vide. Cela pose un problème d'interprétation : cette valeur vide peut être perçue comme si aucune valeur de température n'avait été enregistrée pour l'URSS en 2010, au lieu de se rendre compte que cette combinaison de membres (URSS et 2010) est invalide.

Par conséquent, pour éviter cette erreur d'interprétation, définissons la contrainte de requête suivante :

Exemple 4.7 : "Il est incorrect de combiner l'URSS avec des jours après le 26 décembre 1991 dans une requête OLAP".

Bien que cet exemple de requête puisse être résolu en utilisant des structures de données spatio-multidimensionnelles particulières telles que les structures de gestion de versions DW, CI de requête permet aux concepteurs de modéliser toute requête non valide pouvant être indépendante des aspects liés à la gestion du temps (par exemple, certains produits ne peuvent pas être vendus dans certains magasins).

4.4 Le cadre conceptuel

Nous présentons les concepts principaux d'un profil UML et d'une OCL spatiale.

Rappel :

— Les profils UML permettent de personnaliser UML pour des domaines ou des plates-formes particuliers en étendant ses métaclasses (classe, propriété, etc.).

— Un profil est défini à l'aide de trois mécanismes d'extension : les stéréotypes, les valeurs étiquetées et les contraintes.

— Un stéréotype est une extension d'une métaclasse UML.

— Les valeurs marquées sont des propriétés de stéréotypes.

— Une contrainte OCL précise la sémantique d'application de chaque stéréotype.

— OCL fournit une méthode indépendante de la plate-forme pour modéliser les contraintes. Il peut être interprété par les générateurs de code pour générer le code automatiquement.

— Les contraintes OCL peuvent être définies au niveau du méta-modèle (profil UML, par exemple), mais également au niveau du modèle (instance de profil).

— Spatial OCL est une extension de OCL qui prend en charge les relations topologiques spatiales (intérieur, intersection, etc.).

Afin de définir les données SOLAP, l'agrégation et le CI d'interrogation au niveau conceptuel, nous proposons un cadre basé sur un profil UML et une OCL spatiale.

L'idée principale est de disposer d'un profil UML unique définissant 3 modèles interconnectés à représenter de manière conceptuelle :

1. structures de données SDW (modèle SDW),
2. comment les mesures sont agrégées pour répondre aux exigences d'analyse (modèle d'agrégation), et
3. Modèle de requête IC.

Puis définissons les CIs avec Spatial OCL en utilisant ces modèles.

— Data IC est défini par les concepteurs utilisant Spatial OCL en haut de l'instance du modèle SDW.

— Aggregation IC est défini comme les contraintes de stéréotypes du modèle d'agrégation utilisant OCL et

— Query IC est défini à l'aide du modèle Query IC et de Spatial OCL.

Soit le modèle de l'EDS (« SDWModel »). Il est considéré comme un ensemble fini d'hypercubes non vides (« Hypercube ») pour lequel au moins un est spatial.

Les hypercubes représentent des sujets d'analyse. Chaque hypercube est constitué d'un ensemble fini de mesures («measure») regroupées en une classe de faits («fact») et d'un ensemble fini non vides de dimensions («dimension»). Il est considérée comme spatial (isSpatial = true) s'il contient au moins une dimension spatiale ou au moins une mesure spatiale (voir Figure 4.2). (Boulil, 2012)

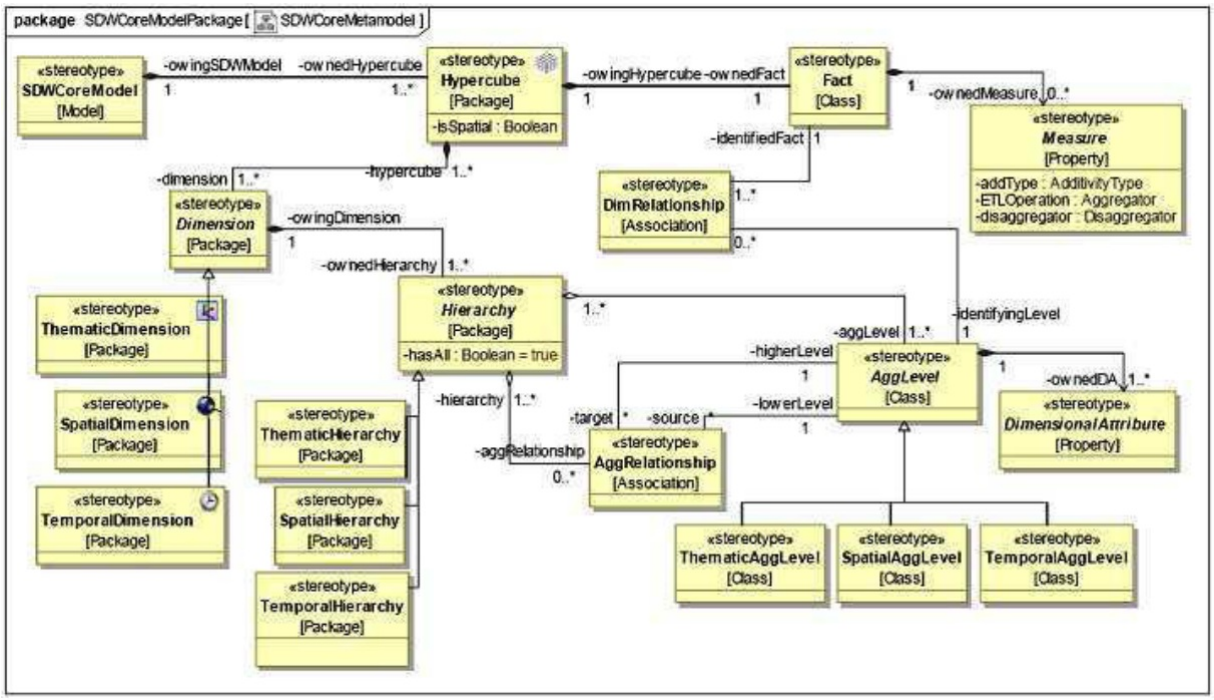


FIGURE 4.2 – Exemple de Le Profile d'EDS

Exemple 4.8 : Un cas du modèle EDS est présentée à la figure 4.3. Cette instance de modèle SDW contient deux dimensions :

— une dimension spatiale composée de 3 niveaux spatiaux (stéréotypés « SpatialAggLevel »), Ville, Région et Pays ; et

— une dimension temporelle composée de trois niveaux temporels jour, mois et année.

La mesure numérique de température (stéréotype « NumericalMeasure ») est définie comme l'attribut de classe de faits Temperature (stéréotype « Fact »).

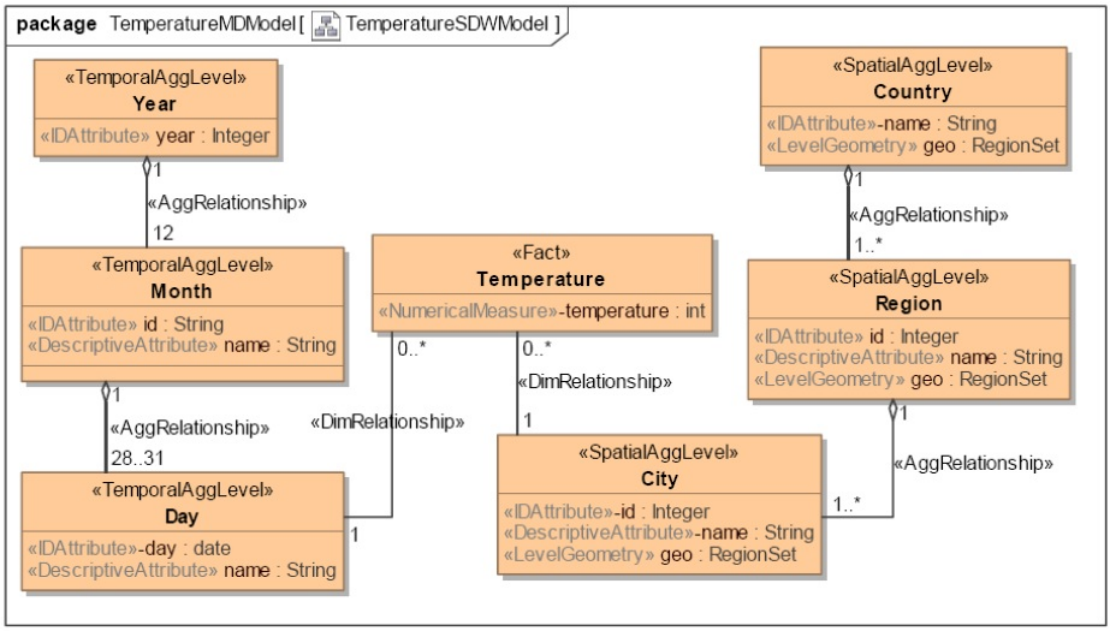


FIGURE 4.3 – Instance du model d'EDS

Une fois l'instance de modèle EDS a été définie, les contraintes d'intégrité des données peuvent être exprimées à l'aide de OCL. Par exemple, la CI de données de l'exemple 1 est exprimé comme suit :

$ContextRegioninvDataIC1$:
$Self.geo.isInside(country.geo) or$
$Self.geo.coveredBy(country.geo)$

La CI de données de l'exemple 4.2 est exprimé en utilisant OCL de la manière suivante :

$ContextTemperatureinvDataIC2$:
$Not($
$Self.day.day >= `1991-12-26' and$
$Self.city.region.country.name =' URSS'$
)

Le modèle d'agrégation montre comment les mesures sont agrégées selon des dimensions en fonction des besoins des décideurs en matière d'analyse. L'instance de modèle d'agrégation de notre étude de cas, qui indique que la mesure de la température (valeur étiquetée aggregatedAttribute) est agrégée le long de toutes les dimensions à l'aide de la fonction d'agrégation moyenne (agrégateur = valeur étiquetée Avg), est illustrée à la figure 4.4.

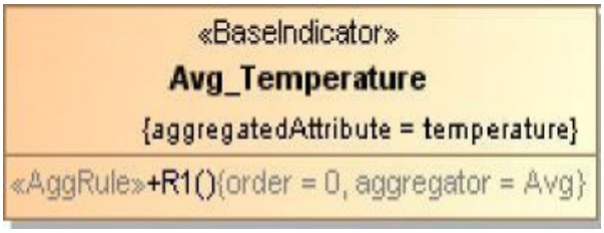


FIGURE 4.4 – Instance du model d'EDS

Un ensemble de contraintes d'agrégation doit être identifié permettant d'obtenir des agrégations significatives de mesures. Ces contraintes sont valables pour toutes les applications SOLAP. Elles peuvent être implémentées en tant que contraintes OCL dans le package de modèle d'agrégation du profil. Ils sont vérifiés par l'outil CASE (ou bien USE) au stade de la conception lors de la validation du modèle conceptuel.

Exemple 4.9 Afin de forcer l'utilisateur à ne pas agréger de mesures non additives (ou de valeur par unité), par exemple la température, à l'aide de la fonction d'agrégation somme, l'instruction OCL suivante est définie dans le profil :

$ContextAggRuleinvNotSumValuePerUnitMeasure$:
$if($
$baseIndicator.aggregateAttrubute.OclIsKindOf(Measure)$
$andbaseIndicator.aggregatedAttribute.addType = `ValuePerUnit'$
$)thenaggregator.name <> `Sum'$

Enfin, les concepteurs peuvent exprimer des requêtes IC sur (S)OLAP en utilisant le modèle Query IC. En règle générale, une requête (S)OLAP est une combinaison de mesures et de membres de différentes dimensions. Ainsi, le modèle IC de requête peut être utilisé par exemple pour définir des combinaisons non valides d'ensembles de membres. Ces ensembles de membres sont spécifiés en tant qu'attributs avec le stéréotype « MemberSet ». Le domaine de valeurs d'un attribut « MemberSet » est un sous-ensemble de membres d'un niveau de dimension, dont la définition est spécifiée avec la valeur tagged condition, qui est une instruction OCL définie sur le contexte du niveau de dimension pour sélectionner un sous-ensemble de ses membres.

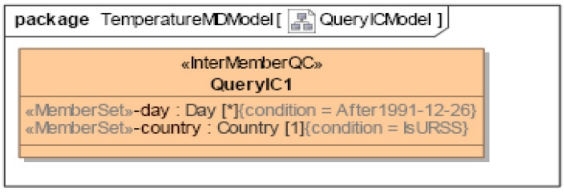


FIGURE 4.5 – Instance du model CI de Requête

Un exemple d'instance du modèle Query IC est illustré à la figure 4.5, où l'utilisateur déclare que la combinaison des jours (« MemberSet »>) après le 26 décembre 1991 (condition = After1991-12-26, avec l'URSS (pays « MemberSet ») n'a pas de sens dans

une requête SOLAP. L'expression OCL est comme suit :

$Context Day inv After 1991 - 12 - 26 :$
$Self.day >= `1991 - 12 - 26'$

4.5 Conclusion

L'assurance de la qualité des données et leur cohérence, et par conséquent, la fiabilité d'un système d'information décisionnel, peuvent être atteintes grâce au contrôle des CI de différents types. Celles-ci peuvent être spécifiées au stade de la conception de modèles ; ce qui permet de rajouter la sémantique aux modèles.

Pour ce fait, des solutions pour la description des contraintes d'intégrité sont étudiées. Elles sont destinées aux structures multidimensionnelles notamment les EDS. Différents outils et méthodes existent permettant la spécification des contraintes sur les données. Nous parlons ici de l'OCL, le langage de spécification des contraintes d'intégrité objet.

Dans ce chapitre, nous avons traité les problèmes de contrôle de la qualité d'analyse dans les systèmes (S)OLAP. Nous avons montré que cette qualité dépend de trois aspects : la qualité des données entreposées, la qualité d'agrégation et la qualité d'exploration.

Dans le chapitre suivant, nous allons traiter les techniques d'optimisation des requêtes dans les entrepôts de données (spatiales).

Chapitre 5

Optimisation des requêtes dans les ED

Dans ce chapitre, nous allons présenter les différentes approches qui permettent l'optimisation des requêtes dans les Entrepôts de Données.

Sommaire

5.1 Introduction

Vu la taille importante des données pouvant être stockées dans un entrepôt de données, qui rend leur interrogation lente mesurée par le temps de réponse d'une part, et d'autre part la complexité des requêtes décisionnelles qu'il l'exploite. Cette complexité est due aux opérations de jointure et d'agrégation utilisées par les requêtes, qui détériorent de manière significative les performances de l'entrepôt. De ce fait, il apparaît donc nécessaire de concevoir des techniques pour l'optimisation des performances des requêtes d'entrepôts de données.

5.2 Techniques d'optimisation des requêtes

Dans la littérature il existe plusieurs techniques pour optimiser les requêtes analytiques. Elles peuvent être classées en deux catégories : redondantes et non redondantes illustrées dans la Figure 5.1 ci après.

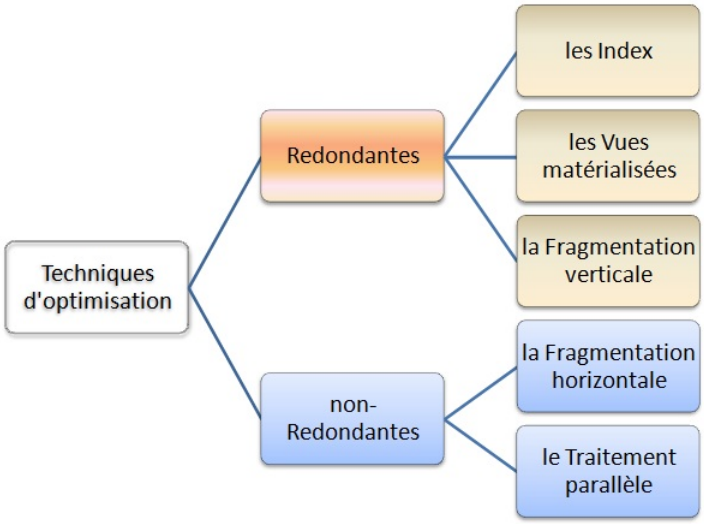


FIGURE 5.1 – Solutions d'optimisation

1. ***Redondantes*** :
   les techniques redondantes sont des techniques d'optimisation des requêtes qui nécessitent un stockage physique des données et une maintenance dont l'utilisation produit une duplication des données. on distingue : les indexes, les vues matérialisées et la fragmentation verticale.
2. ***non-Redondantes*** :
   les techniques non redondantes sont des techniques d'optimisation des requêtes qui ne nécessitent pas un stockage physique des données. ces techniques ne dupliquent pas les données mais réorganisent leur représentation physiques. On distingue la fragmentation horizontale et traitement parallèle.

Les méthodes de traitement parallèle permettent d'exécuter les opérations en parallèle pour assurer un gain de temps de traitement. dans ce qui suit : Nous présentons trois solutions d'optimisation, à savoir les vues matérialisées, les index et la fragmentation.

5.3 Les vues matérialisées

Une vue matérialisée est une table contenant les résultats d'une requête. Les vues améliorent l'exécution des requêtes en pré calculant les opérations les plus coûteuses comme les jointures et les agrégations, en stockant leurs résultats dans la base. En conséquence, certaines requêtes nécessitent seulement l'accès aux vues matérialisées et sont ainsi exécutées plus rapidement. Cependant, la mise à jour des données implique systématiquement celle des vues matérialisées calculées à partir de ces données afin de conserver la cohérence et

l'intégrité des données. Cela induit une surcharge du système liée au coût de maintenance des vues matérialisées. De plus, la matérialisation des vues requiert un espace de stockage additionnel que l'administrateur alloue à ces vues. On note deux grands problèmes liés à la sélection et à la maintenance de vue matérialisée.

5.3.1 Problème de sélection de vue matérialisée

Il s'agit de déterminer l'ensemble de vues à matérialiser en tenant compte d'un certain nombre de paramètres comme les requêtes les plus fréquentes, l'espace de stockage et le coût de maintenance.

5.3.2 Problème de maintenance de vue matérialisée

Le coût d'exécution des requêtes est en conflit avec le coût de maintenance des vues car la matérialisation favorise l'optimisation de requêtes mais en contre partie elle entraîne un sur coût de maintenance des données en cas de mise à jour des données sources (Theodoratos et Sellis, 1998 ; Bellatreche, 2000).

De nombreux travaux traitent ces problématiques, nous pouvons distinguer deux axes principaux de recherche :

1. La maintenance incrémentale des vues matérialisées qui se propose de répercuter les mises à jour survenues au niveau des données sources sans recalculer complètement les vues ;
2. La sélection des vues à matérialiser qui propose des algorithmes permettant de déterminer une configuration de vues à matérialiser dans l'entrepôt de données de telle sorte que le coût d'exécution des requêtes soit optimal.

Après la sélection des vues matérialisées, toutes les requêtes définies sur l'entrepôt doivent être réécrites en fonction des vues disponibles. Ce processus est appelé réécriture des requêtes en fonction des vues. La réécriture des requêtes a attiré l'attention de nombreux chercheurs car elle est en relation avec plusieurs problèmes de gestion de données : l'optimisation de requêtes, l'intégration des données, la conception des entrepôts de données, etc. Le processus de réécriture des requêtes a été utilisé comme technique d'optimisation pour réduire le coût d'évaluation d'une requête.

Exemple 5.1 :

À partir d'ORACLE, le processus de réécriture de requête incorporé transforme une commande SQL de telle façon qu'elle puisse accéder aux vues matérialisées. Cet outil de réécriture permet de réduire significativement le temps de réponse pour des requêtes d'agrégation ou de jointure dans les grandes tables des entrepôts. Quand une requête cible une ou plusieurs tables de base pour calculer un agrégat (ou pour réaliser une jointure) et qu'une vue matérialisée contient les données requises, l'optimiseur d'oracle peut réécrire la requête d'une manière transparente pour exploiter la vue, et procurer ainsi un temps de réponse plus court.

La matérialisation des vues est une approche embarrassante du fait qu'elle nécessite une anticipation des requêtes à matérialiser. Or, les requêtes dans les environnements des entrepôts sont souvent ad hoc et ne peuvent pas toujours être anticipées.

5.4 Les index

Dans les systèmes de gestion de bases de données (SGBD), l'accès aux données est d'autant plus lent que la base de données est volumineuse. Un parcours séquentiel des données est une opération lente et pénalisante pour l'exécution des requêtes, notamment dans le cas des opérations de jointure où ce parcours doit souvent être effectué de façon répétitive. La création d'un index permet d'améliorer considérablement le temps d'accès aux données en créant des chemins d'accès directs.
Un index est (généralement) une table ordonnée contenant les couples utilisée pour accélérer la recherche des enregistrements d'un fichier. Il existe deux types d'index :

1. **Les index primaires (clustered ou index groupants)**
   Si le champ clé ne contient de valeurs en double, l'index est alors « primaire ». Les adresses contenues dans cet index sont triées suivant le placement physique sur disque des n-uplets composant la table indexée. Peu de blocs disques sont parcourus et les requêtes de recherche sont ainsi résolues de manière efficace. Néanmoins, cette méthode souffre d'un coût de maintenance très élevé car il faut maintenir l'ordre du tri, dans une table, il peut y avoir au plus un index primaire.
2. **Les index secondaires (non-clustered ou index non-groupants)**
   Pour améliorer les recherches basées sur des champs non clés (appelés aussi clés secondaires), on peut construire des index secondaires sur ces champs. Les adresses contenues dans un index secondaires sont moins efficaces que les index primaires, mais moins coûteux au niveau de la maintenance, index secondaires sur une table donnée sont possibles.

En plus, dans les entrepôts de données, nous devons faire la différence entre Les techniques d'indexation, et la sélection des indexes

5.4.1 Techniques d'indexation

Dans ce qui suit, nous explorons les principales techniques d'indexation utilisées dans les SGBD relationnels et les entrepôts de données :

1. **Index en B-arbre**
   Un B-arbre est une liste chaînée de noeuds dont la valeur est celle de l'index. Les feuilles de l'arbre font référence à :
   — Une seule valeur, si cet index est construit sur un attribut clé des n-uplets de la table indexée
   — Plusieurs valeurs, si cet index est construit sur un attribut non-clé des n-uplets de la table indexée

Cette référence spécifie l'emplacement physique du n-uplet sur le disque. Un B-arbre offre un excellent compromis pour les opérations de recherche par clé et par intervalle, ainsi que pour les mises à jour. Ces qualités expliquent le fait que les B-arbres et leurs variantes soient systématiquement intégrés dans la plupart des SGBD.

La Figure 5.2 montre un exemple de B-arbre construit sur la table "Film" définie par le schéma Film (F_id, F_titre, F_realisateur ,...).

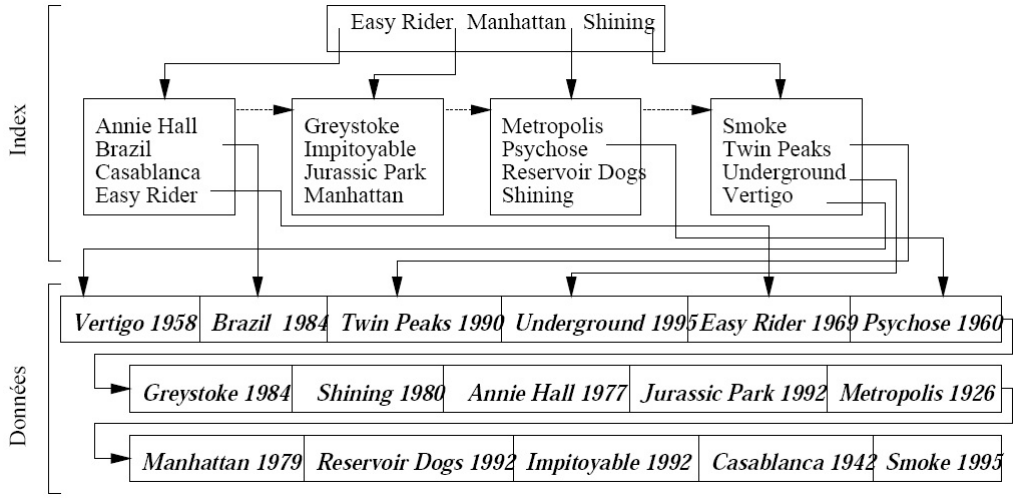


FIGURE 5.2 – Index en B-arbre construit sur l'attribut Personne "Titre"

2. **Index de hachage**

   Les tables de hachage sont des structures de données très couramment utilisées en mémoire centrale pour organiser des ensembles et fournir un accès performant à leurs éléments.

   L'idée de base du hachage est d'organiser un ensemble d'éléments d'après une clé et d'utiliser une fonction, dite de hachage, qui, pour chaque valeur de clé c, donne l'adresse f (c) d'un espace de stockage où l'élément doit être placé. En mémoire principale, cet espace de stockage est en général une liste chaînée et, en mémoire secondaire, un ou plusieurs blocs sur le disque.

   La Figure 5.3 montre un exemple d'index de hachage construit sur la table "Film". La fonction de hachage est H (Titre) = rang (Titre [0]) mod 5, où Nom [0] désigne la première lettre du Titre d'un Film.

   Une fonction de hachage mal conçue affecte tous les n-uplets à la même adresse et la structure dégénère vers un simple fichier séquentiel. Cela peut être le cas, avec notre fonction basée sur la première lettre du titre, pour tous les Films dont le titre commence par la lettre l.

3. **Index bitmap**

   Un index bitmap repose sur un principe très différent de celui des index en B-arbre.

   Alors que dans ces derniers, on trouve, pour chaque attribut indexé, les mêmes valeurs dans l'index et dans la table, un index bitmap considère toutes les valeurs possibles de l'attribut indexé, que la valeur soit présente ou non dans la table. Pour

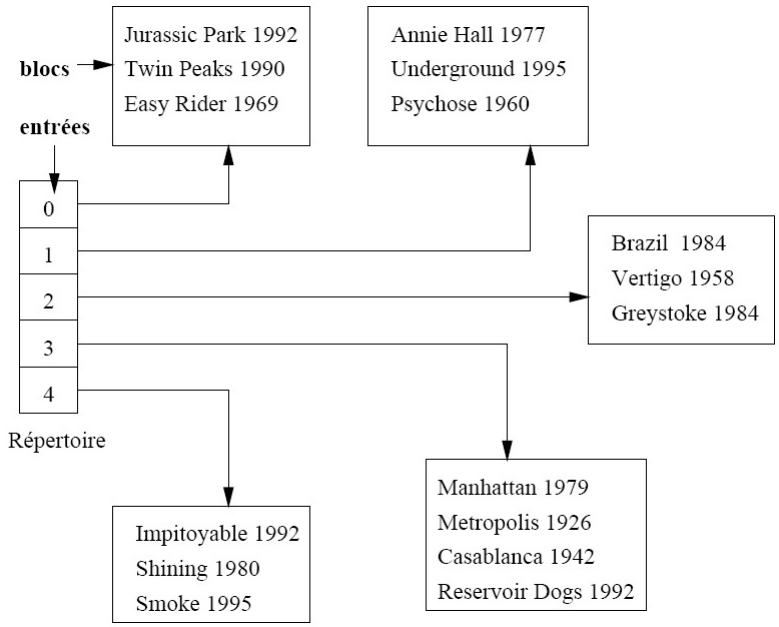


FIGURE 5.3 – Index de hachage construit sur l'attribut Nom

chacune de ces valeurs possibles, un tableau de bits, dit bitmap, est stocké. Ce bitmap est composé d'autant de bits qu'il y a de n-uplets dans la table indexée. Notons par A l'attribut indexé et v la valeur définissant le bitmap. Chaque bit associé à un n-uplet a alors la signification suivante :

si le bit est mis à 1, l'attribut A a pour valeur v pour ce n-uplet ; sinon, le bit est mis à 0.

Lorsque les n-uplets dont la valeur est y sont recherchés, il suffit donc de prendre le bitmap associé à y , de chercher tous les bits à 1 et d'accéder ensuite aux n-uplets correspondants.

Un index bitmap est très efficace si le nombre de valeurs possibles de l'attribut indexé est relativement faible. Un index bitmap est de très petite taille comparé à un B-arbre construit sur le même attribut. Il est donc très utile dans des applications de type entrepôt de données gérant de gros volumes de données et classant les informations par des attributs catégoriels définis sur de petits domaines de valeurs. Certaines requêtes peuvent alors être exécutées très efficacement, parfois sans même recourir à la table contenant les données.

Prenons l'exemple de la table Client et créons un index bitmap sur le sexe des personnes. La Figure 5.4 montre l'index bitmap pour les valeurs Féminin et masculin.

4. **Index de jointure**

   L'opération de jointure est très coûteuse en terme de temps de calcul lorsque les tables concernées sont grandes. Plusieurs méthodes ont été proposées pour accélérer ces opérations. Ces méthodes incluent les boucles imbriquées, le hachage, la fusion, etc.

   Valduriez (Valduriez, 1987) a proposé des index spécialisés appelés index de jointure,

FIGURE 5.4 – Index bitmap construit sur le sexe des clients

pour préjoindre des relations. Un index de jointure matérialise les liens entre deux relations par le biais d'une table à deux colonnes, contenant les RID (identifiant de n-uplet) des n-uplets joints deux par deux. Ce genre d'index est souhaité pour les requêtes des systèmes OLTP car elles possèdent souvent des jointures entre deux tables.

Par contre, pour les entrepôts de données modélisés par un schéma en étoile, ces index sont limités. En effet les requêtes décisionnelles définies sur un schéma en étoile possèdent plusieurs jointures. Il faut alors subdiviser la requête en fonction des jointures. Or le nombre de jointures possibles est de l'ordre de N !, N étant le nombre de tables à joindre (problème d'ordonnancement de jointure).

Pour résoudre ce problème, Redbrick (Redbrick, 97) a proposé un nouvel index appelé index de jointure en étoile, adapté aux requêtes définies sur un schéma en étoile. Un index de jointure en étoile peut contenir toute combinaison de clés étrangères de la table des faits.

Ce type d'index est dit complet s'il est construit en joignant toutes les tables de dimensions avec la table des faits. Un index de jointure partiel est construit en joignant certaines des tables de dimensions avec la table des faits. En conséquence, l'index complet est bénéfique pour n'importe quelle requête posée sur le schéma en étoile. Il exige cependant beaucoup d'espace pour son stockage.

5.4.2 Sélection d'index

A partir d'un ensemble de requêtes décisionnelles et la contrainte d'une ressource donnée (l'espace, le temps de maintenance, etc.) on sélectionne un ensemble d'index afin de minimiser le coût d'exécution des requêtes.

Le groupe base de données de Microsoft a développé un outil pour sélectionner des index avec Microsoft SQL Server. L'architecture de l'outil de sélection des index proposé est illustrée ci-dessous.

L'outil prend un ensemble de requêtes définies sur un schéma de base de données. Le traitement est itératif. Durant la première itération, il choisit les index sur une colonne

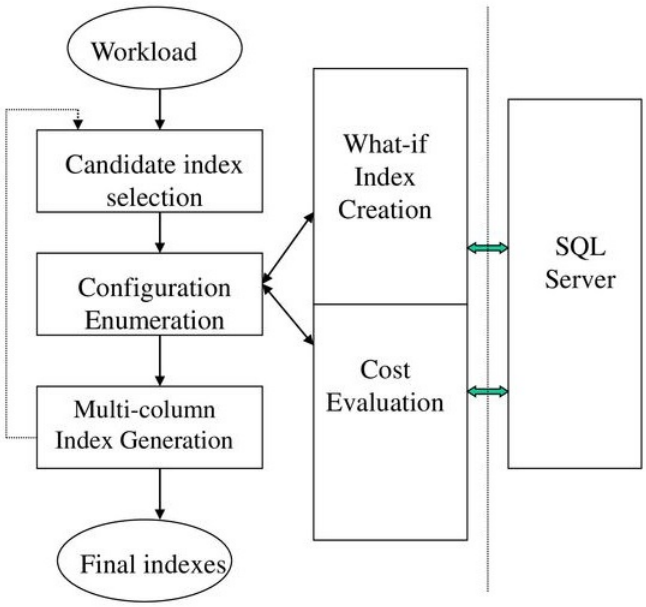


FIGURE 5.5 – L'architecture de l'outil de sélection d'index

(mono index) ; dans la deuxième les index sur deux colonnes et ainsi de suite. L'algorithme de recherche d'index est testé en fonction de ces trois modules :

— La sélection des index candidats,

— L'énumération des configurations,

— La génération des multi index.

Le module de sélection des index candidats permet de déterminer la meilleure configuration pour chaque requête d'une manière indépendante. Finalement, il fait l'union de ces configurations.

S'il existe n index candidats, et que l'outil doit sélectionner k parmi n index, le module d'énumération doit énumérer toutes les configurations, et à l'aide d'un modèle de coût sélectionner le meilleur ensemble de configurations garantissant un coût minimal.

Cet algorithme de sélection des index prend une requête à un moment donné et sélectionne tous les index possibles. Cependant, l'ensemble des index utilisant cette méthodologie pourra exiger beaucoup d'espace de stockage et des coûts de maintenance élevés.

Dans le but de minimiser les coûts de stockage et de maintenance, Chaudhuri (Chaudhuri et al., 1999) ont proposé une technique appelée fusion d'index (index merging). Elle prend un ensemble d'index ayant une capacité d'espace S et fournit un nouvel ensemble d'index ayant une capacité d'espace SO inférieure à celle de départ (SO < S). L'opération de fusion est guidée par un modèle de coût : la fusion est appliquée s'il y a une réduction dans le coût d'exécution des requêtes. La technique de fusion d'un ensemble d'index ressemble à la reconstruction des fragments verticaux d'une relation donnée.

— Tous les algorithmes proposés pour résoudre ces problèmes sont dirigés par un modèle de coût. Ce dernier permet non seulement de dire si une vue (ou index) est plus

bénéfique qu'une autre vue (ou index), mais également d'orienter ces algorithmes dans leur sélection. En conséquence il faut prévoir un modèle de coût des requêtes pour mieux les optimiser. Le modèle de coût accepte en paramètre le plan d'exécution d'une requête et retourne son coût. Le coût d'un plan d'exécution est évalué en cumulant le coût des opérations élémentaires (sélection, jointure, etc.). Ces modèles de coûts contiennent, d'une part, des statistiques sur les données et, d'autre part, des formules pour évaluer le coût. Ces coûts sont mesurés en unités de temps si l'objectif est de réduire le temps de réponse des requêtes, le nombre d'entrées-sorties ou le temps de maintenance des vues et des index.

— L'optimisation par index se fait d'une manière séquentielle ; c'est-à-dire, d'abord la sélection des vues matérialisées et ensuite la sélection des index. Cette façon de procéder ne prend pas en compte l'interaction entre les vues et les index et pose un problème de gestion de ressources. Par exemple, considérons que ces deux problèmes soient contraints par la capacité d'espace. Il s'agit alors de savoir comment distribuer l'espace entre les vues et les index afin de garantir une meilleure performance des requêtes ?

5.5 La fragmentation

La fragmentation est une technique, permettant l'optimisation de performances des requêtes et d'éviter le balayage de grandes tables, consiste à diviser un schéma de données en plusieurs fragments (sous schémas) de telle façon que la combinaison de ces fragments produit l'intégralité des données sources, sans perte ou ajout d'information. Le but est de réduire le temps d'exécution des requêtes.

5.5.1 La fragmentation verticale

C'est une relation qui est divisée en sous relations appelées fragments verticaux qui sont des projections appliquées à la relation (Figure 5.6). La fragmentation verticale favorise naturellement le traitement des requêtes de projection portant sur les attributs utilisés dans le processus de la fragmentation, en limitant le nombre de fragments à accéder. Son inconvénient est qu'elle requiert des jointures supplémentaires lorsqu'une requête accède à plusieurs fragments.

Les auteurs dans (DATTA et al., 1999) exploitent la fragmentation verticale pour construire un index nommé "Cuio" dans un entrepôt modélisé par un schéma en étoile Cuio permet d'accélérer l'accès aux données et optimise l'espace de stockage en matérialisant les fragments au lieu des attributs indexés.

Afin de minimiser le temps de réponse des requêtes on utilise la fragmentation verticale pour partitionner des vues définies sur un entrepôt. Cette fragmentation est basée sur une charge de requêtes et un modèle de coût. Selon les auteurs, la fragmentation verticale désigne deux opérations : d'une part le partitionnement d'une vue en plusieurs fragments et, d'autre part, l'unification en une seule vue de deux ou plusieurs vues ayant une clé commune.

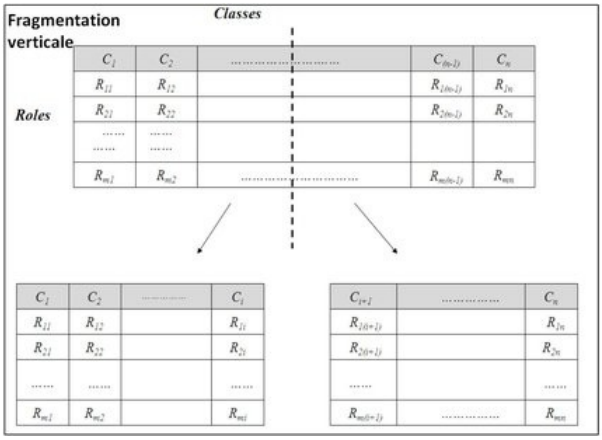


FIGURE 5.6 – Fragmentation verticale

L'unification respecte la règle de reconstruction d'une table fragmentée à partir de ses fragments verticaux et vise à réduire la redondance des vues. Les auteurs supposent que leur approche peut être bénéfique pour la distribution de l'entrepôt sur une architecture parallèle et proposent de combiner leur algorithme de fragmentation avec un algorithme d'allocation des fragments sur les noeuds distants.

5.5.2 La fragmentation horizontale

Elle consiste à diviser une relation R en sous ensembles de n-uplets appelés fragments horizontaux, chacun étant défini par une opération de restriction appliquée à la relation (Figure 5.7). Les n-uplets de chaque fragment horizontal satisfait une clause de prédicats.

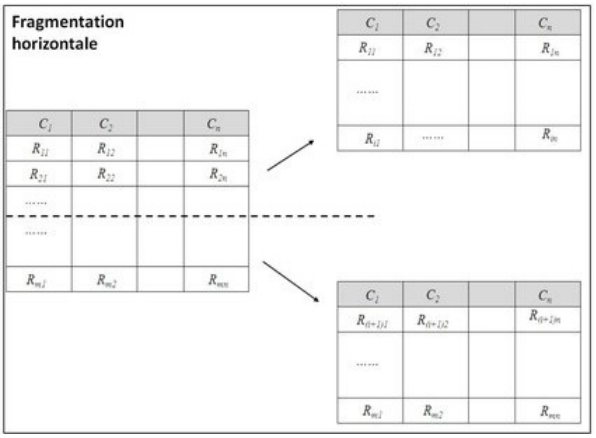


FIGURE 5.7 – Fragmentation Horizontale

Tout algorithme de fragmentation horizontale nécessite la donnée d'un ensemble de requêtes les plus fréquentes. A partir de ces requêtes, on extrait deux types d'informations :

— Les informations qualitatives concernent les tables de dimension et sont représentées par les prédicats de sélection simples utilisés dans les requêtes.

— Les informations quantitatives concernent la sélectivité de ces prédicats et les fréquences d'accès des requêtes.

On rappelle qu'un prédicat simple p est défini par :
p : Ai Θ Valeur
Où Ai est un attribut d'une relation à fragmenter,
$\Theta \in \{=, <, \leq, \geq, >, \neq\}$,
Valeur ∈ Domaine (Ai).

Pour ce qui est de la fragmentation horizontale, certains auteurs ont proposé une technique de construction d'un entrepôt réparti en utilisant la stratégie descendante. Cette stratégie est utilisée pour la conception de bases de données réparties. Elle part du schéma conceptuel global d'un entrepôt, qu'elle répartit pour construire les schémas conceptuels locaux. Cette répartition se fait en deux étapes essentielles, à savoir, la fragmentation et l'allocation, suivies éventuellement d'une optimisation locale. (Saichi , 2009)

Bellatreche (Bellatreche, 2000) applique la fragmentation horizontale dérivée sur un schéma en étoile et propose plusieurs approches basées sur un ensemble de requêtes. L'auteur adapte les algorithmes proposés dans le contexte des bases de données réparties. Ces algorithmes se basent sur la complétude et la minimalité des prédicats ou sur les affinités des requêtes.

Compromis entre le coût de maintenance des fragments et le coût d'exécution des requêtes. Ils sont basés sur des modèles de coût et procèdent en trois étapes :

— génération de plusieurs schémas de fragmentation,

— évaluation de ces schémas,

— sélection d'un schéma optimal.

Le premier algorithme est exhaustif et consiste à construire tous les schémas de fragmentation possibles par fragmentation horizontale. Il énumère ensuite ces schémas et calcule pour chacun d'eux le coût d'exécution des requêtes de la charge.

Il sélectionne finalement le schéma qui correspond au coût minimum Le deuxième algorithme est approximatif. Il construit un schéma initial par l'algorithme de fragmentation dirigée par les affinités, puis l'améliore par des opérations de fusion ou de décomposition des fragments. Finalement, le troisième algorithme exploite un algorithme génétique pour sélectionner un schéma de fragmentation que l'on va comparer avec notre contribution dans le chapitre conception.

5.5.3 La fragmentation mixte

La fragmentation mixte partitionne verticalement des fragments horizontaux ou horizontalement des fragments verticaux (Figure 5.8). Les algorithmes de fragmentation mixte ont été étudiés dans le contexte relationnel et sont subdivisés en deux types : la fragmentation par création de grille et la fragmentation par définition de vues.

Selon (Wu et Buchmaan,1997), les auteurs recommandent de combiner la fragmentation horizontale et la fragmentation verticale. Selon eux, la table de faits peut être partitionnée horizontalement à partir de fragments définis sur les dimensions. Elle peut aussi

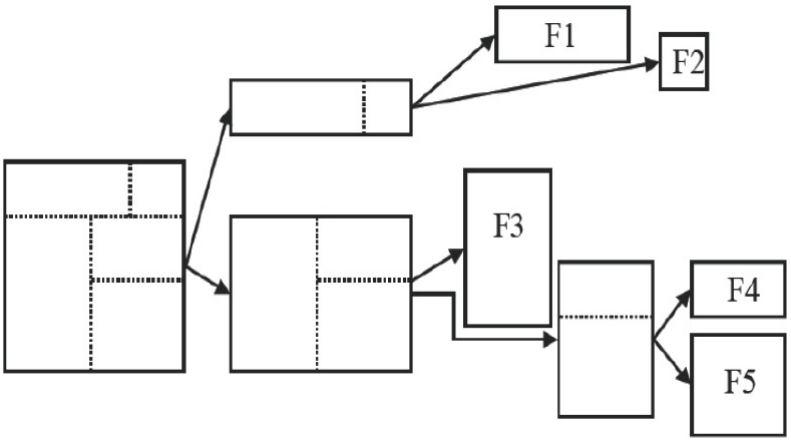


FIGURE 5.8 – Fragmentation mixte

être partitionnée verticalement selon les clés étrangères faisant référence aux dimensions. Cependant, aucun algorithme de fragmentation n'est proposé.

5.6 Conclusion

Les entrepôts de données sont devenus maintenant non pas un phénomène de mode mais un instrument indispensable à la bonne marche de l'organisation. Ils sont en effet à la base de toute stratégie et prise de décision de l'entreprise. Il faut en extraire les informations nécessaires à la prise de décision et également leur structure. De cet entrepôt sont extraites des bases de données multidimensionnelles, car elles permettent de regarder l'organisation sous différents angles ou dimensions, ces dernières sont constituées que de données propres à la décision.

Dans ce chapitre, nous avons présenté les techniques principales utilisées pour optimiser le temps d'exécution des requêtes dans les entrepôts de données. Il s'agit de la fragmentation, des vues matérialisées, et de la création d'index.

Il est à noter aussi, que données traditionnelles (ou spatiales) sont devant un processus incrémental produisant ainsi des ED(S) volumineux. Nous serons dans certains cas, dans l'obligation de l'utilisation des systèmes NoSQL permettant la gestion des Big Data. Ceci fera l'objet du chapitre suivant.

Chapitre 6

Entrepôts de données NoSQL

Ce chapitre est consacré pour premièrement présenter un Rappel des bases de données NoSql. Ensuite, nous allons passer à la présentation des entrepôts de données NoSql.

6.1 Introduction

Sous l'impulsion du volume croissant de données (Big Data) issues de différentes applications (réseaux sociaux), les approches d'entrepôt de données (DW) doivent être adaptées. Généralement, les modèles en étoile, flocon de neige ou constellation sont utilisés comme modèles logiques. Tous ces modèles sont inadéquats lorsqu'il s'agit de données sociales qui nécessitent des systèmes évolutifs et flexibles. Au lieu de cela, les systèmes NoSQL commencent à se développer.

Les systèmes NoSQL ont montré leurs avantages par rapport aux systèmes relationnels en termes de flexibilité et de traitement de données volumineuses. Dans la littérature, un certain nombre de chercheurs ont reconnu les faiblesses du modèle de données OLAP traditionnel et ont proposé des approches pour la migration de bases de données relationnelles vers des bases de données NoSQL (approches indirectes).

6.2 Modèles NoSQL

Le terme NoSQL désigne une nouvelle approche des bases de données basée sur une performance capable de gérer une grande quantité d'informations, d'assurer une haute disponibilité de service et d'avoir une bonne scalabilité pour gérer la montée en charge. Le NoSQL adopte une approche non relationnelle.

Il existe différentes manières de structurer les données mais l'ensemble des solutions développées peut être divisé en quatre modèles : le modèle orienté clé-valeur, modèle orienté colonnes, le modèle orienté documents et le modèle orienté graphes. Pour distinguer ces modèles, E. Evans dans (Evans, 2004) définit la notion d'agrégat comme une unité d'information identifiée par une racine. Cet agrégat est l'objet de la modélisation. Les modèles orienté clé-valeur, orienté colonnes et orienté documents peuvent modéliser directement un agrégat d'informations, tandis que modèle orienté graphe ne le permet pas (El Malki, 2016).

6.2.1 Modèle orienté clé-valeur

Le modèle clé-valeur (Cattell, 2011 ; Dey et al., 2013) considère chaque enregistrement comme une clé associée à une valeur. Cette approche s'assimile à un tableau associatif de deux colonnes où la clé identifie de manière unique la valeur associée. Contrairement au modèle relationnel, la taille et le type de la valeur ne sont pas déterminés au préalable. Le cas du modèle clé-valeur est plus flexible, la valeur de la clé peut changer d'un enregistrement à l'autre, elle peut à chaque ligne avoir un type et une taille différente.

Une seconde caractéristique du modèle clé-valeur réside dans la modularité des clés, qui peuvent être générées automatiquement ou construites selon une certaine logique. La manière dont la clé peut être construite est laissé libre ; le développeur peut imaginer diverses méthodes de génération de clés pour faciliter sa recherche.

Exemple 6.1 :
Si l'on prend l'exemple de la Figure 6.1, de la base de données relatives aux Tweets (tweeter), nous pouvons construire des clés basées sur une concaténation Id-U et Country. Cette clé permet de rechercher plus rapidement tous les utilisateurs d'un pays donné ayant émis un Tweet. La clé peut être étendue. Si l'on poursuit l'exemple précédent, pour rechercher efficacement tous les utilisateurs d'un pays donné en fonction du temps, nous pouvons utiliser Id-U, Country et Day.

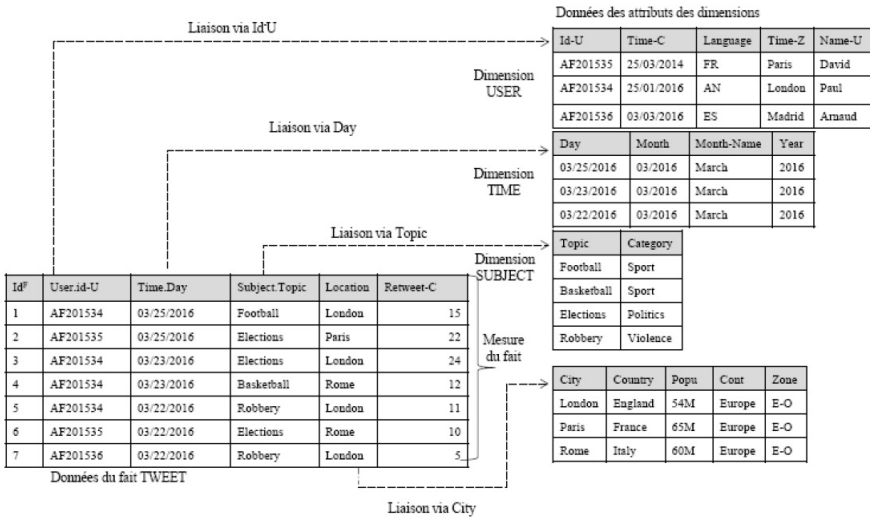


FIGURE 6.1 – Exemple d'entrepôts de données multidimentionnelles R-OLAP concernant des tweets

Le modèle orienté clé-valeur sera comme illustré dans la Figure 6.2.

L'organisation de données en paires clé-valeur permet de bonnes performances. Les clés sont nativement indexées. La communication avec la base de données est limitée aux opérations de bases, désignées par l'acronyme CRUD (create, read, update, delete). En revanche, pour des recherches à partir d'autres critères, notamment des parties de la valeur, les performances sont impactées. L'absence de structuration de la valeur oblige l'utilisateur à programmer l'accès aux valeurs dans un langage externe au système. On

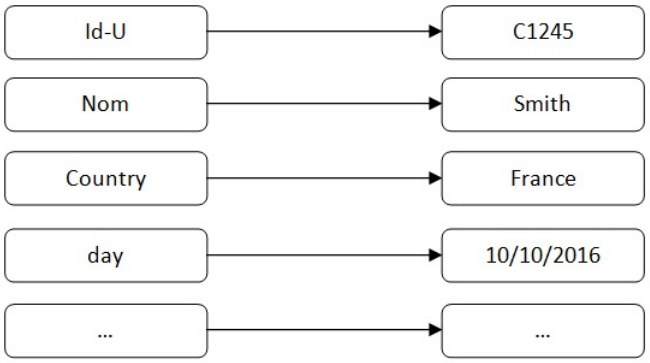


FIGURE 6.2 – Principe du modèle orienté clé-valeur

parle de *« user-defined function (UDF) »*.

6.2.2 Modèle orienté documents

Selon le même principe que le modèle orienté clé-valeur, le modèle orienté documents est constitué de paires clé-valeur. La clé identifie une valeur structurée, appelée document. Un document est considéré comme une hiérarchie d'éléments pouvant être soit des valeurs atomiques, soit des valeurs composées (valeurs atomiques multiples ou documents imbriqués). Le niveau d'imbrication n'est pas limité. Une valeur peut être également une référence vers un autre document. L'ensemble des documents est contenu dans une collection qui correspond à la notion de table en relationnel, et suit un stockage physique horizontal.

Dans le modèle orienté documents, un document est une structure lisible par le moteur NoSQL. Il est défini dans un format textuel balisé, généralement le format JSON (Java Script Objet Notation) (Crockford, 2006). Il facilite la représentation de données structurées notamment d'une manière hiérarchique, voir Figure 6.3. D'autres formats de représentation sont possibles tels que le XML.

Dans l'approche NoSQL orientée documents, le schéma n'est pas établi à l'avance, chaque document peut avoir sa propre structure, on parle de dynamic schema (Scherzinger et al., 2013). Nativement, l'enregistrement des valeurs dans un document n'est soumis à aucun contrôle du système. Cependant, il est préférable de garder une structure minimale commune afin de faciliter la manipulation des données.

Contrairement au modèle clé-valeur, il est possible de manipuler directement les éléments constituant une valeur donnée. Par exemple, il devient possible de manipuler une sous-partie d'un document représentant un utilisateur comme son nom ou son adresse sans devoir extraire toute la valeur. De plus, le modèle orienté documents permet l'indexation élargie à d'autres attributs (autres que la clé) ce qui peut améliorer les performances de l'interrogation.

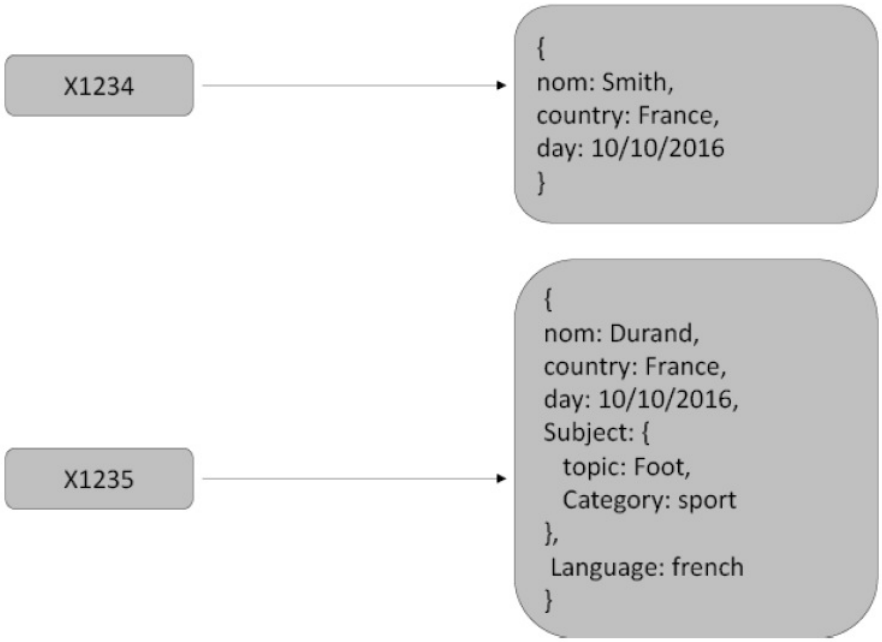


FIGURE 6.3 – Principe du modèle orienté documents

6.2.3 Modèle orienté colonnes

Le modèle orienté colonnes (Chang et al., 2008 ; Aniceto et al., 2015) est la seconde forme évoluée du modèle clé-valeur. Dans les bases de données relationnelles, la structure des données est déterminée en avance par le schéma de la relation, avec un nombre limité de colonnes, chacune similaire pour tous les enregistrements (« n-uplets »). Le modèle orienté colonnes fournit un schéma flexible (colonnes non typées) pouvant varier entre chaque enregistrement (chaque ligne). La flexibilité d'une base NoSQL orientée colonnes permet de gérer l'absence de certaines colonnes entre les différentes lignes de la table.

Une base de données orientée colonnes est un ensemble de tables qui sont définies ligne par ligne, mais dont le stockage physique est organisé verticalement par groupe de colonnes, appelés « familles de colonnes ». Une famille de colonnes peut contenir un très grand nombre de colonnes (George, 2011). Le nombre de colonnes peut varier d'une ligne à l'autre ; chaque famille de colonnes s'apparente à un ensemble clé-valeur où la clé est vue comme une colonne associée à une valeur. Voir Figure 6.4

Un autre avantage des modèles orientés colonnes est la possibilité de stocker plusieurs valeurs « historisées » (versionning). Pour illustrer cet avantage, nous pouvons nous appuyer sur un exemple d'actualité en France ; la répartition géographique des régions de France a été modifiée et cela nécessite une évolution des données. Dans le modèle relationnel, la solution passe par la création d'une nouvelle table avec la nouvelle répartition et maintenir en parallèle les deux tables. Dans le modèle orienté colonnes il est possible d'insérer les nouvelles données et maintenir un numéro de version par valeur (Ravat et al., 2005).

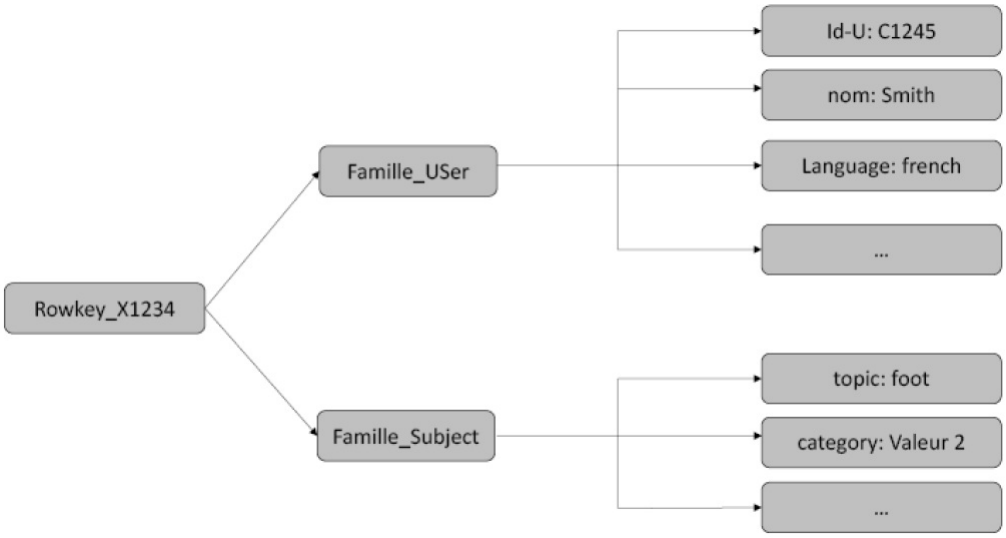


FIGURE 6.4 – Principe du modèle orienté colonnes

6.2.4 Modèles orienté graphes

Le modèle orienté graphes se distingue des modèles précédents car il ne peut prendre en compte nativement la notion d'agrégats d'information proposée par (Evans, 2004). Le modèle orientés graphes est fondé sur la théorie des graphes. Le modèle orienté graphes repose sur trois notions ; noeud, relation et propriété. Chaque noeud possède des propriétés. Les relations relient les noeuds et possèdent éventuellement des propriétés. Ce type d'approche facilite les requêtes navigationnelles entre les noeuds en suivant les relations qui les relient : chaque noeud a un pointeur physique vers les noeuds voisins permettant une recherche locale rapide.Voir Figure 6.5

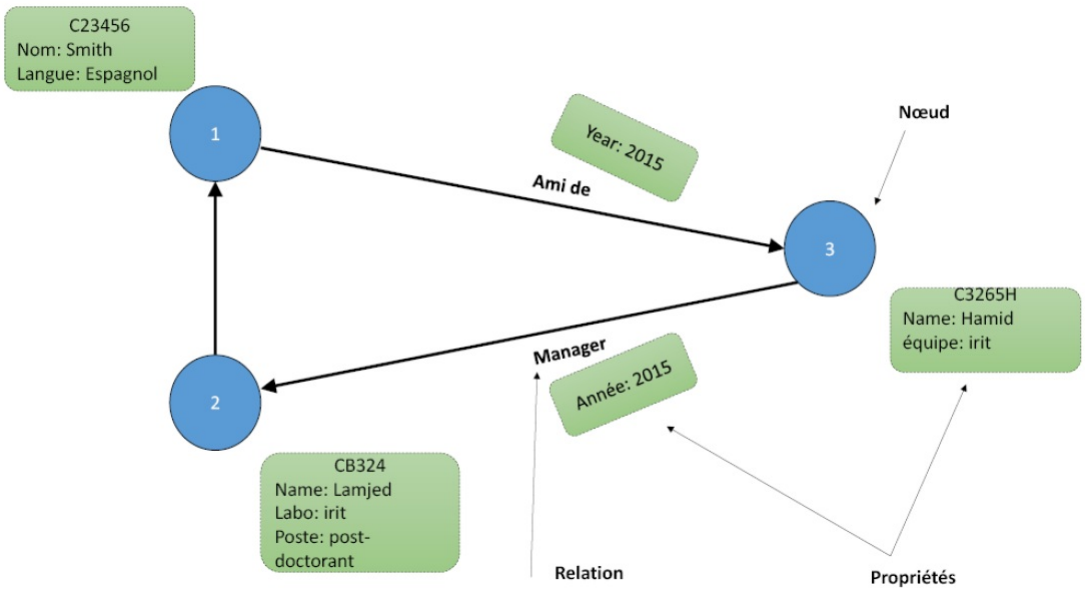


FIGURE 6.5 – Principe du modèle orienté graphes

La structure du modèle orienté graphes est très adaptée pour répondre à des problématiques telles que la gestion d'un réseau social d'une entreprise ou tout autre stockage nécessitant le parcours de graphes.

Le modèle orienté graphe se caractérise lui aussi par une flexibilité de schéma ; il n'est pas nécessaire de créer un schéma au préalable pour les noeuds et les relations.

6.3 Entrepôts de données sous NoSQL

Dans cette section, nous présentons les travaux de recherche portant sur le développement des entrepôts de données avec ces nouveaux systèmes.Voir Figure 6.6

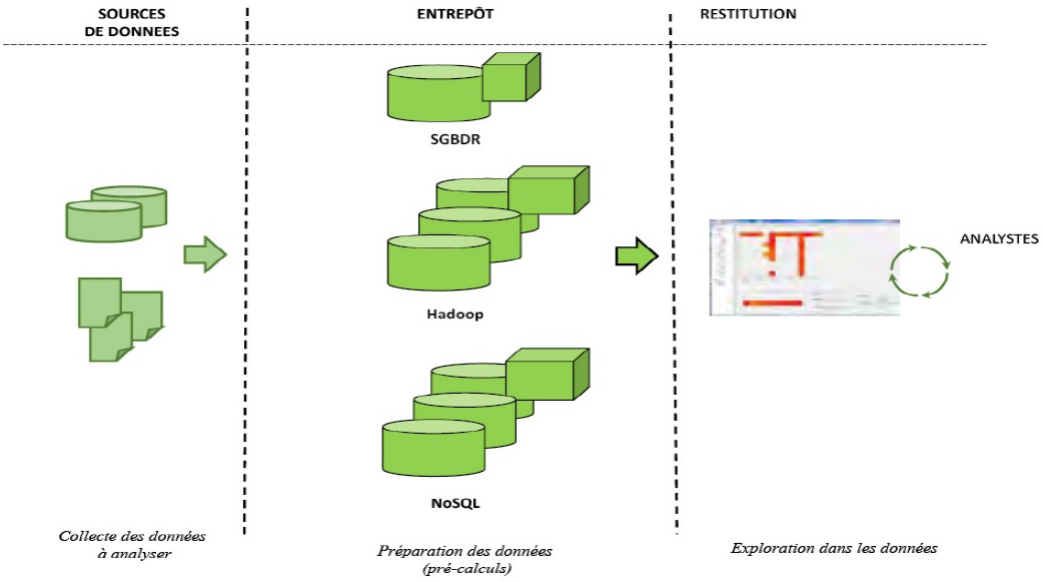


FIGURE 6.6 – Nouvelle architecture des systèmes d'aide à la décision intégrant le NoSQL

Nous étudions tout particulièrement les travaux visant à implanter les modèles multidimensionnels du niveau conceptuel à l'aide des modèles NoSQL (Dehdouh et al., 2014 ; Chevalier et al., 2015). Ils se résument en deux catégories :

— La première catégorie consiste à traduire de manière indirecte les schémas multidimensionnels. Le processus réutilise les règles de traduction du conceptuel vers le R-OLAP, puis est étendu par de nouvelles règles permettant un passage du modèle intermédiaire R-OLAP vers le modèle NoSQL cible.

— La seconde catégorie consiste à traduire directement les schémas multidimensionnels du niveau conceptuel vers le modèle NoSQL cible.Voir Figure 6.7

6.3.1 Processus de traduction indirecte

De très nombreux travaux ont étudié la transformation des schémas conceptuels des entrepôts de données multidimensionnelles vers le modèle relationnel. Nous détaillons ici simplement les travaux qui se focalisent sur la transformation de schémas relationnels

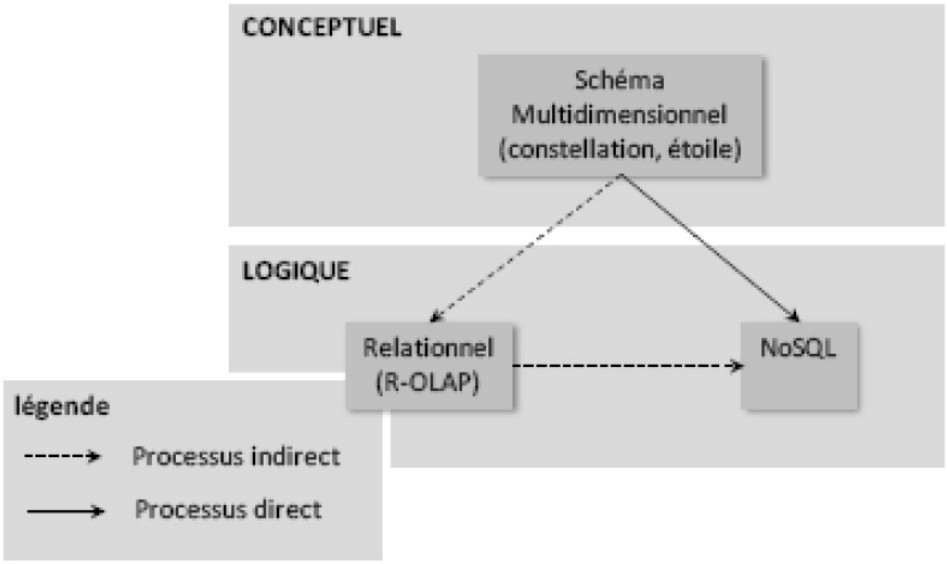


FIGURE 6.7 – Processus de transformation des entrepôts de données multidimensionnelles du niveau conceptuel vers le niveau logique

en NoSQL. Ceux-ci, ciblent principalement les modèles orienté colonnes ou orienté documents.

(Steve Ataky et al., 2015) développent un framework utilisant le système HBase basé sur un modèle orienté colonnes. Le schéma relationnel est restructuré dans une unique table. Cette dernière comporte une famille de colonnes pour chaque table relationnelle. Une famille de colonnes additionnelle est utilisée pour conserver les clés des tuples des relations initiales.

(Chongxin Li, 2010) propose également une méthode en trois étapes pour structurer automatiquement un schéma relationnel dans HBase. A chaque étape, un ensemble de règles est appliqué pour la correspondance entre le schéma relationnel et le modèle logique HBase. La gestion des dépendances se traduit par l'ajout d'une famille de colonnes comme dans l'approche précédente.

(Scavuzzo et al., 2014) utilisent également le système HBase. Pour chaque élément du schéma relationnel, une règle désigne l'élément correspondant dans le modèle orienté colonnes. Chaque table est automatiquement transformée en une famille de colonnes placée dans une même table HBase. Les clés primaires sont concaténées et utilisées pour identifier la ligne. Chaque attribut du schéma relationnel est une colonne.

Des travaux similaires, consistent à traduire le schéma relationnel dans un modèle orienté documents.

Tous ces travaux proposent de convertir le schéma relationnel dans MongoDB. Toutes les relations du schéma sont regroupées dans une même collection. Chaque tuple de la relation universelle est traduit en un document imbriquant les valeurs issues de chaque relation. Les données sont ainsi transformées dans un schéma dénormalisé dans l'objectif d'éviter l'utilisation de jointure dans ces environnements distribués.

Outre les approches académiques, il existe des solutions industrielles de transformations

de données relationnelles vers des bases de données NoSQL. Dans la solution Apache Sqoop, les données sont transformées depuis une base de données relationnelle vers l'espace de stockage HDFS puis en second lieu les données sont transformées de l'espace HDFS vers une base de données NoSQL. Bien que Sqoop assure tout le job d'importation dans les deux sens, il ne permet pas d'importer les dépendances entre les tables.

Une deuxième solution est Datax. Elle permet, et avec des performances relativement intéressantes, de faire des migrations entre deux bases de données à très grande vitesse. Datax fournit des fonctionnalités pour la planification des processus d'importation. Tous les processus d'importation et d'exportation sont assurés par des plugins Datax. En revanche, et comme la solution Sqoop, les relations entre les tables ne sont pas assurées dans ces mécanismes d'échanges.

Ces travaux peuvent être combinés aux approches d'implantation R-OLAP des entrepôts de données, mais la prise en compte du schéma multidimensionnel et des treillis d'agrégats associés ne sont pas abordés.

6.3.2 Processus de traduction direct

Nous relevons également un certain nombre de travaux, qui traitent de l'implantation directement de schémas multidimensionnels conceptuels dans les modèles NoSQL.

1. **Hive**
   Le système Hive est un système spécialisé pour la construction d'entrepôts de données massives. Ce dernier repose sur un espace de stockage, appelé memstore s'interfaçant avec d'autres bases de données NoSQL (HBase dans la majorité des cas) et un langage d'interrogation proche du SQL appelé HQL.

   Des scripts HQL permettent d'insérer les données dans des tables externes (external table), par exemple des tables HBase tout en gardant les métadonnées dans des tables Hive. Les données sont manipulées grâce à la correspondance entre les attributs de Hive et les attributs de la table HBase. Il est aussi possible d'interroger les données directement depuis un terminal HBase. L'utilisation de Hive est motivée par la richesse de son langage d'interrogation qui permet d'utiliser des fonctionnalités relationnelles et l'utilisation de requêtes assez complexes.Voir Figure 6.8

2. **Travaux sur les entrepôts multidimensionnels**
   Dans l'approche (Abelló et al., 2011) les auteurs proposent trois méthodes pour construire un cube de données dans le modèle orienté colonnes.

   — Dans la première méthode, les données sont dénormalisées dans une seule table.

   — La seconde ajoute des index au niveau des valeurs afin de réduire la taille des données manipulées lors de l'interrogation.

   — Dans la dernière, les auteurs utilisent un index bitmap pour déterminer les positions des identifiants pour réduire encore le nombre de valeurs parcourues.

   D'autres travaux visent à développer un entrepôt de données dans un système NoSQL orienté colonnes, ou orienté clé-valeur. L'objectif de ces travaux est de proposer des benchmarks et n'est pas centré sur le processus de transformation de modèles.

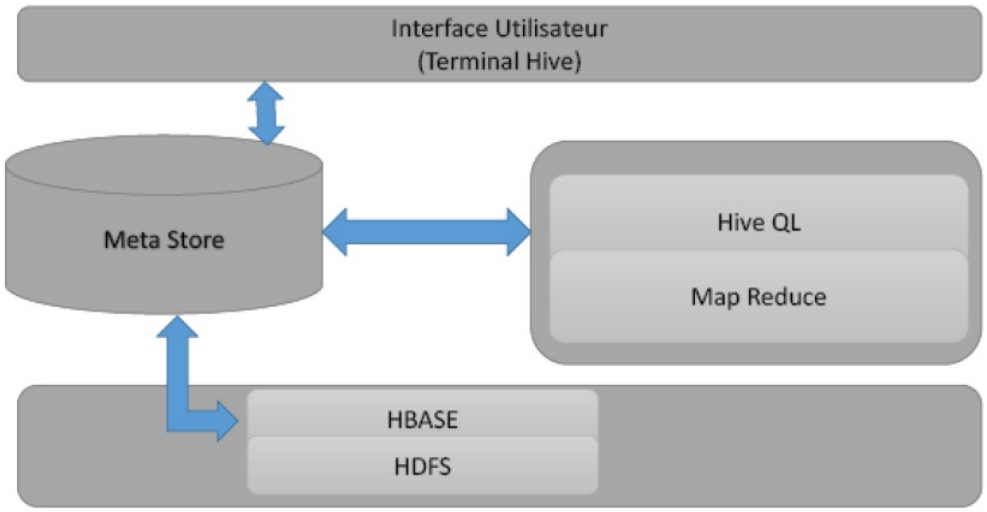


FIGURE 6.8 – Architecture Hive

L'approche de (Scabora et al., 2016) consiste à dénormaliser les données du schéma conceptuel en étoile dans une table HBase composée de deux familles de colonnes. La première famille de colonnes reçoit le fait et les dimensions fréquemment interrogées tandis qu'une deuxième famille de colonnes regroupe les autres dimensions (EL MALKI, 2016).

Voir Figure 6.9

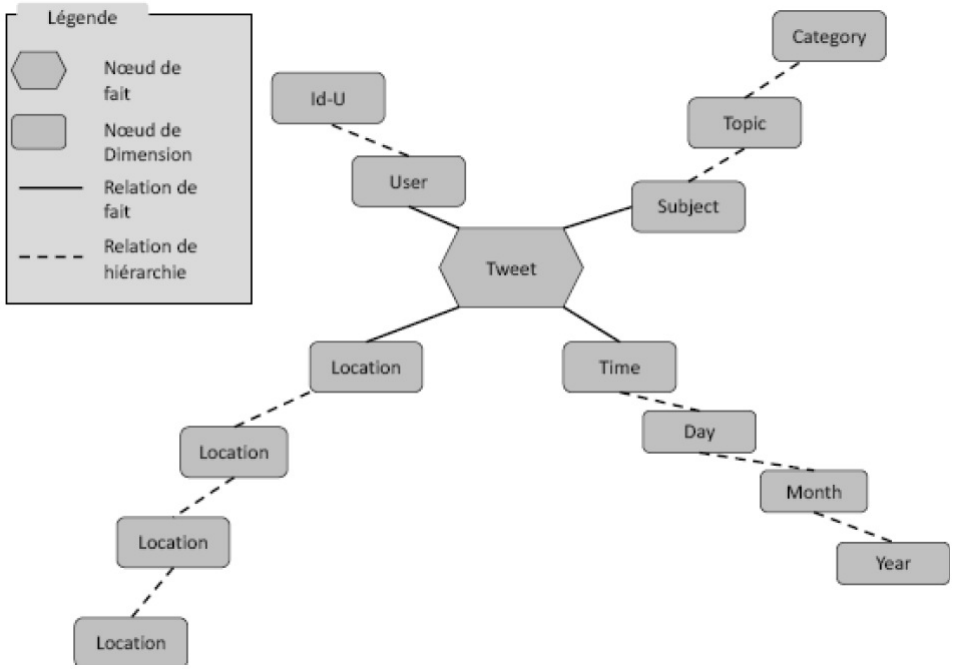


FIGURE 6.9 – Représentation logique orientée graphes (Castelltort et Laurent 2014)

6.4 Modélisation Logique Not-Only-Sql

Dans ce qui suit nous allons nous focaliser sur la modélisation logique des deux modèles NoSQL : orienté documents et orienté colonnes (El Malki, 2016).

6.4.1 Modélisation multidimensionnelle orientée documents

Modèle NoSQL orienté documents

Dans le modèle orienté documents, les données sont organisées en collections de documents. Nous utilisons deux constructeurs :

— [ ] pour représenter une structure formée d'un ou plusieurs attributs, et

— {} pour représenter un ensemble.

Définition 6.1 : Une collection C est définie par (N^C, E^C) où

— N^C est le nom de la collection,

— $E^C = d_{1...}, d_t$ est un ensemble de documents.

La structure des documents est définie par l'intermédiaire d'attributs (pouvant s'apparenter à des couples clé/valeur où le nom de l'attribut est la clé). Nous distinguons les attributs simples dont les valeurs sont atomiques, des attributs composés dont les valeurs sont elles-mêmes des documents appelés documents imbriqués.

Définition 6.2 : Un document di est défini par (S_i, V_i) où

— $S_i = [a_{1,...}, a_n ai]$ est le schéma du document formé par un ensemble d'attributs,

— $V_i = [a_1 : v_1, ..., a_{nai} : v_{nai}]$ est la valeur du document.

Il est important de noter que le schéma est inhérent à chaque document. Dans les systèmes NoSQL orientés documents, chaque document à son propre schéma, pouvant varier d'un document à l'autre, même au sein d'une même collection.

Un attribut simple a_k peut correspondre à une seule valeur atomique vk ou un ensemble de valeurs atmoiques $[a_{k1}, a_{k2}, ...]$.

Un attribut composé a_k peut être composé, c'est-à-dire constitué par une structure imbriquée S_k. Celle-ci peut être mono-valuée (structure plate) ak[Sk], soit multi-valuée $a_k\{[S_k]\}$.

Exemple 6.2 : Ci-dessous nous présentons un document di décrivant une personne.

$di = ($
$[id, nom, prenom, daten[jour, mois, annee], telephone\{[numero]\}],$
$[id : 12345,$
$nom : "Ali",$
$prenom : "mohammed",$
$daten : [jour : "24", mois : "08", annee : "1984"],$
$telephone : \{[numero : "06"], [numero : "04"]\}]$
$)$

L'attribut composé telephone pourrait être un attribut simple dont la valeur est constituée d'un ensemble de valeurs, c'est-à-dire telephone : {ȷ06ȷ, ȷ04ȷ}.

Les sous-sections suivantes présentent 4 processus de traduction, en NoSQL orienté documents, d'un entrepôt de données multidimensionnelles en constellation.

Processus de traduction plate en orienté documents

Ce premier processus repose sur une dénormalisation des données du schéma en constellation. A chaque fait F$\in F^S$ et ses dimensions associées $Star^S(F)$ correspond une collection de même nom (N^F, E^F). Chaque instance du fait est traduite par un document di. Les attributs représentant les mesures du fait et les attributs de dimensions sont contenus dans un même document plat sans imbrication.

Un document di est défini par un schéma Si constitué de la manière suivante :

— id est l'identifiant du document,

— chaque mesure de M^F forme un attribut m_k^F simple,

— chaque attribut de $D_j \in Star^S(F) A_j^D$ forme un attribut ak simple

— A l'instar des implantations R-OLAP, nous ne matérialisons pas au niveau logique les attributs extrémités all^D des dimensions.

Exemple 6.3 : Considérons le schéma conceptuel de l'ED des Tweet. Chaque instance du fait F_Tweet est traduite par un document. L'expression ci-dessous présente la collection obtenue et détaille un document :

{...,
$di = ([id,$
$idUser, name, language, time\_c, time\_z,$
$city, country, population, zone,$
$day, month, month\_name, year,$
$topic, category, Retweet\_NB],\ [id : 12345,$
$idUser$: "C02265", $name$: "$Smith$", $language$: "$french$", $time\_c$: "$Paris$",
$time\_z$:]$France$[,
$city$: "$Paris$", $country$: "$France$", $population$: 66000000,
$zone$: "$Europe - Ouest$",
day : "$03 - 31 - 2015$", $month$: "$03 - 2015$", $month\_name$: "$march$", $year$: 2015,
$topic$: "$foot$", $category$: "$sport$", $Retweet\_NB$: 200],), ...
}

La Figure 6.10 présente sous une forme graphique le document di constitué par un ensemble d'attributs simples issus des mesures et des attributs des dimensions liées.

Cette implantation évite l'utilisation des jointures entre le fait et les dimensions, mais génère un important niveau de redondance par duplication des données des dimensions.

Processus de traduction par imbrication en orienté documents

Ce second modèle vise a tirer profit des possibilités d'imbrication offertes par le modèle oriente documents. Il combine l'utilisation d'attributs simples et d'attributs composés.

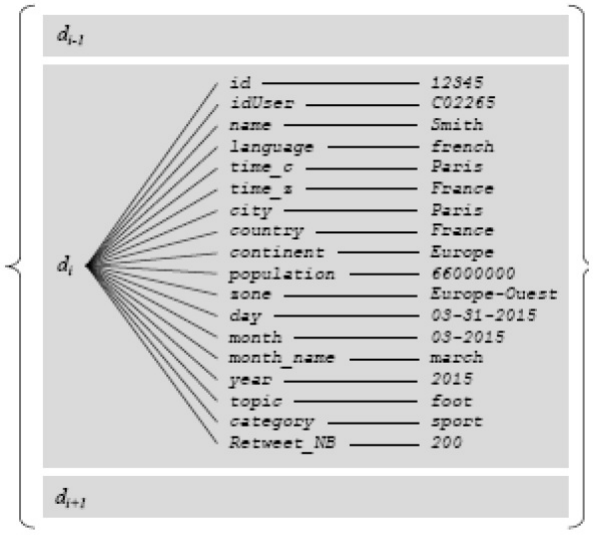


FIGURE 6.10 – Exemple de document par traduction plate

Comme précédemment, a chaque fait $F \in F^S$ et ses dimensions associées $Star^S(F)$ correspond une collection de même nom (N^F, E^F). Chaque instance du fait est traduite par un document d_i.

L'imbrication est utilisee pour rassembler les attributs representant les mesures du fait. De meme, les attributs de chaque dimension associee $Star^S(F)$ sont rassembles dans un meme attribut compose.

Un document di est defini par un schema Si constitue de la maniere suivante :

— id est l'identifiant du document,

— un attribut composé est défini pour le fait, et chaque mesure de M^F forme un attribut m_k^F simple imbriqué,

— un attribut composé est défini pour chaque dimension, chaque attribut $a_k^{D_j}$ de la dimension D_j forme un attribut simple imbriqué.

Exemple 6.4 : Considérons l'instance utilisée dans l'exemple précédent, que nous restructurons ci-dessous en fonction de l'approche par imbrication :
{...,
$di = ([id,$
$User[idUser, name, language, time\_c, time\_z],$
$Location[city, country, population, zone],$
$Time[day, month, month\_name, year],$
$Subject[topic, category], Tweet[Retweet\_NB]],$
$[id : 12345,$
$User : [idUser : "C02265", name : "Smith", language : "french", time\_c : "Paris", time\_z :$
$"France"],$
$Location : [city : "Paris", country : "France", population : 66000000, zone : "Europe -$
$Ouest"],$
$Time : [day : "03-31-2015", month : "03-2015", month\_name : "march", year : 2015],$

$Subject : [topic : "foot", category : "sport"]$,
$Tweet : [Retweet\_NB : 200]],),$
$...\}$

La Figure 6.11 présente sous une forme graphique le document di constitué par un ensemble d'attributs composés issus du fait (imbrication des mesures) et de chaque dimension associée (imbrication des attributs de la dimension).

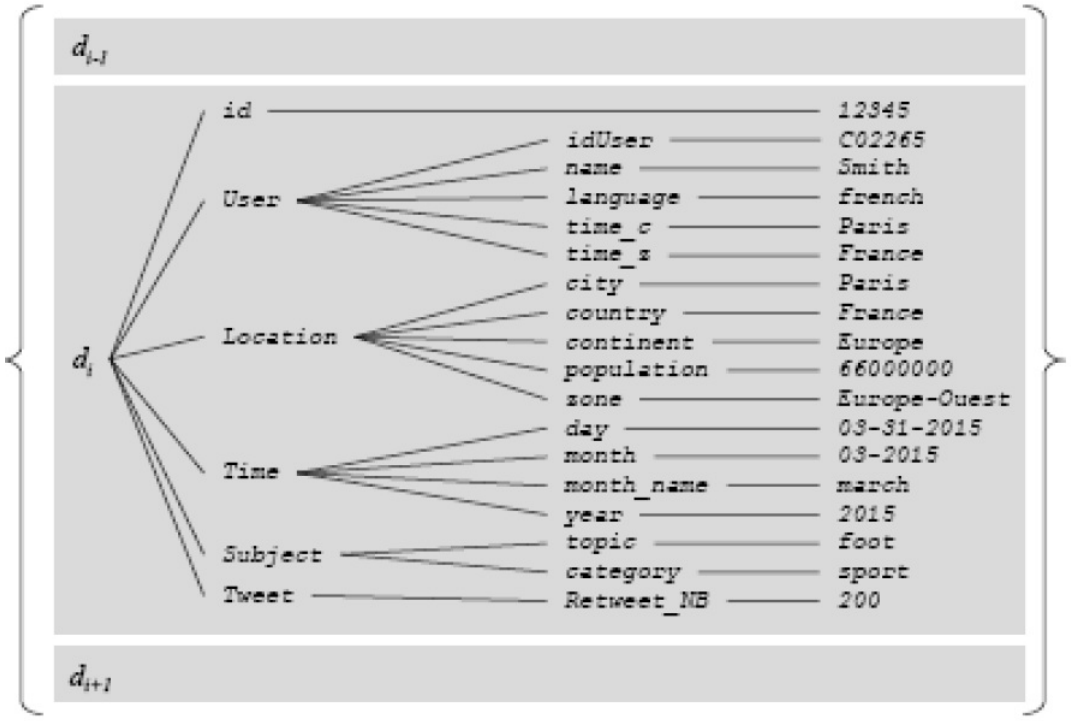


FIGURE 6.11 – Exemple de document par traduction imbriquée

Processus de traduction hybride en orienté documents

Comme les approches précédentes, une traduction hybride rassemble au sein d'une même collection les données issues d'un fait $F \in F^S$ et des dimensions associées $Star^S(F)$.

Les attributs des dimensions sont convertis en attributs simples, à plat sans imbrication, et stockés dans un document de la collection, à raison d'un document par dimension. Les mesures du fait sont converties en attributs simples, à plat sans imbrication, et stockés dans un document de la même collection. Les documents des faits contiennent également les références aux documents représentant les dimensions associées. Pour chaque instance de dimension, un document d_i est défini par un schéma S_i constitué de la manière suivante :

— id est l'identifiant du document, correspondant à la racine id^{D_j} de la dimension,

— chaque attribut $a_K^{D_j}$ de la dimension Dj forme un attribut simple.

Pour chaque instance du fait, un document d_i est défini par un schéma S_i constitué de la manière suivante :

— id est l'identifiant du document,

— chaque mesure de M^F forme un attribut m_k^F simple,

— pour chaque dimension $D_j \in Star^S(F)$ associée, un attribut simple id^{D_j} est ajouté référençant un document lié.

Exemple 6.5 : Considérons l'exemple de document précédent. Dans l'approche hybride, un document est constitué pour chaque instance des dimensions ainsi qu'un autre document pour l'instance du fait. Nous obtenons les documents définis ci-dessous :

{...,
$diTweet = ([id, idUser, city, day, topic,\ Retweet\_NB]$,
$[id : 12345, idUser :$ "$C02265$", $city :$ "$Paris$", $day :$ "$03-31-2015$", $topic :$ "$foot$", $Retweet\_NB : 200],)$,
$diUser = ($
$[idUser, name, language, time\_c, time\_z]$,
$[idUser :$ "$C02265$", $name :$ "$Smith$", $language :$ "$french$", $time\_c :$ "$Paris$", $time\_z :$ "$France$"$],)$,
$diLocation = ([city, country, population, zone], [city :$ "$Paris$", $country :$ "$France$", $population : 66000000$,
$zone :$ "$Europe - Ouest$"$],)$,
$diTime = ([day, month, month\_name, year], [day :$ "$03 - 31 - 2015$", $month :$ "$03 - 2015$", $month\_name :$ "$march$", $year : 2015],)$,
$diSubject = ([topic, category], [topic :$ "$foot$", $category :$ "$sport$"$]$,
$), ...$
}
La Figure 6.12 présente sous une forme graphique les documents $d_i^{Tweet}, d_i^{User}, d_i^{Location}, d_i^{Subject}$, constitués par un ensemble d'attributs simples issus respectivement du fait et de chaque dimension.

6.4.2 Modélisation multidimensionnelle orientée colonnes

Modèle NoSQL orienté colonnes

Dans le modèle orienté colonnes, les données sont organisées en tables contenant un ensemble de lignes, chacune organisée en colonnes. Le formalisme est basé sur deux constructeurs :

— [ ] pour représenter une structure formée d'un ou plusieurs colonnes, et

— {} pour représenter un ensemble.

Définition 6.3 : Une table T est définie par (N^T, E^T) où

— N^T est le nom de la table,

— $E^T = \{r_1, ..., r_t\}$ est un ensemble de lignes.

La structure des lignes est définie par l'intermédiaire de colonnes (pouvant s'apparenter à des paires clé-valeur). Nous distinguons les colonnes simples dont les valeurs sont

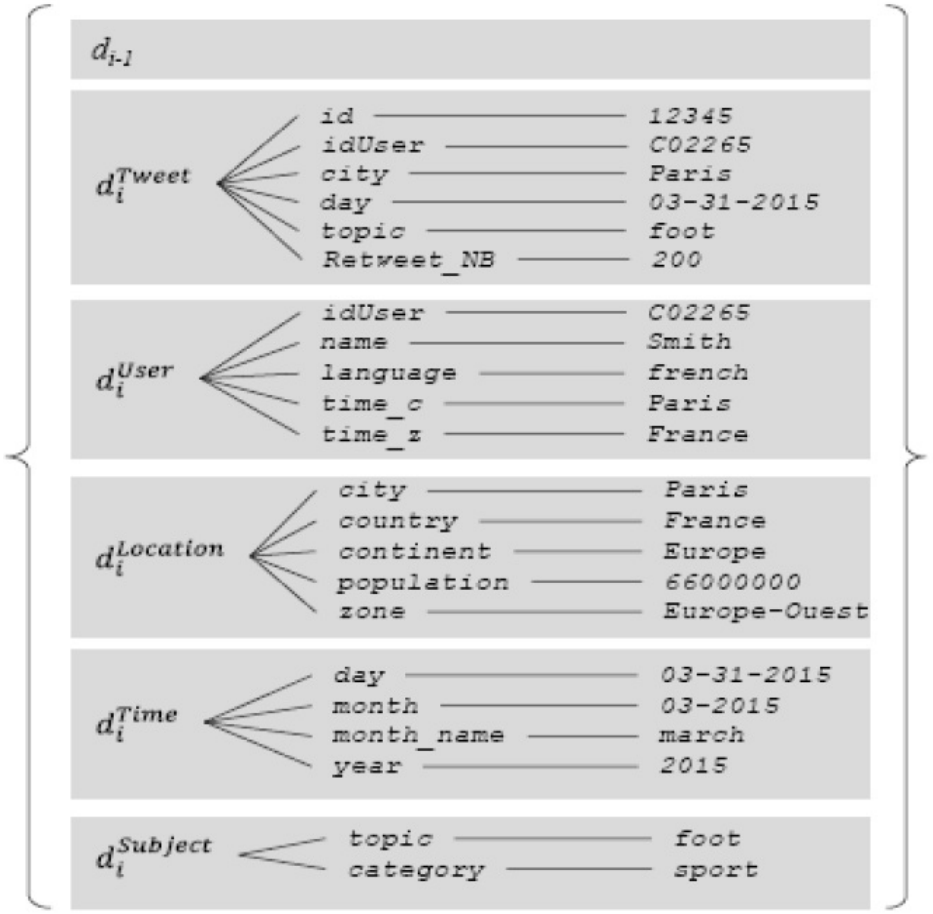


FIGURE 6.12 – Exemple de document par traduction hybride

atomiques, des colonnes composées (appelées familles de colonnes) dont les valeurs sont elles-mêmes des regroupements de colonnes.

Définition 6.4 : Une ligne r_i est définie par (S_i, V_i) où

— $S_i = [c_1, ..., c_{nci}]$ est le schéma de la ligne, formé par un ensemble de colonnes,

— $V_i = [c_1 : v_1, ..., c_{nci} : v_{nci}]$ est la valeur de la ligne.

Le schéma est inhérent à chaque ligne. Dans les systèmes NoSQL orientés colonnes, chaque ligne à son propre schéma, pouvant varier d'une ligne à l'autre, même au sein d'une même table.

Il est important de noter qu'un seul niveau d'imbrication n'est possible : une famille de colonnes est composée uniquement de colonnes simples. Une colonne c_k appelée famille de colonnes, peut être composée de colonnes, c'est-à-dire constituée par une structure imbriquée S_k. Celle-ci suit une structure mono-valuée c'est-à-dire une structure plate à un seul niveau d'imbrication.

Exemple 6.6 :

Ci-dessous nous présentons une ligne r_i décrivant une personne.

$ri = ([id,$

$personne[nom, prenom],$

$daten[jour, mois, annee]$,
$telephone[fixe, portable]]$,
$[id : 12345, personne[nom : "Mohamed", prenom : "Ali"]$,
$daten[jour : "24", mois : "08", annee : "1984"]$,
$telephone[fixe : "04", portable : "06"]$
)
Remarquons que dans l'approche orientée colonnes, il n'est pas possible de construire des colonnes multi-valuées. Dans l'exemple, nous ne pouvons modéliser la colonne telephone par un ensemble de colonnes $\{[numero]\}$.

Les sous-sections suivantes présentent 4 processus de traduction, en NoSQL orienté colonnes, d'un entrepôt de données multidimensionnelles en constellation.(El Malki, 2016)

Processus de traduction plate en orienté colonnes

A chaque fait $F \in F^S$ et ses dimensions associées $Star^S(F)$ correspond une table de même nom (N^F, E^T). Ce premier processus repose sur une dénormalisation des données du schéma en constellation. Chaque instance du fait est traduite par une ligne r_i. Les mesures du fait et les attributs de dimensions sont traduits en colonnes simples au sein d'une unique famille de colonnes.

Une ligne r_i est définie par un schéma S_i constitué de la manière suivante :

— id est l'identifiant de la ligne,

— chaque mesure de M^F forme un attribut m_k^F simple de la famille de colonnes,

— chaque attribut de $U_{D_J \in star^s(F)} A^{D_j}$ forme un attribut a_k simple.

A l'instar des implantations ROLAP, nous ne matérialisons pas au niveau logique les attributs extrémités all^D des dimensions.

Exemple 6.7 :
Considérons le schéma conceptuel des Tweets. Chaque instance du fait F_{Tweet} est traduite par une ligne. L'expression ci-dessous présente la table obtenue et détaille une ligne :
{...,
$ri = ([id,$
$Tweet[idUser, name, language, time\_c, time\_z,$
$city, country, population, zone,$
$day, month, month\_name, year, topic, category,$
$Retweet\_NB]]$,
$[id : 12345, Tweet : [$
$idUser : "C02265", name : "Smith", language : "french",$
$time\_c : "Paris", time\_z : "France",$
$city : "Paris",$
$country : "France", population : 66000000,$
$zone : "Europe - Ouest",$
$day : "03 - 31 - 2015", month : "03 - 2015", month\_name : "march",$
$year : 2015,$
$topic : "foot",$

category : "*sport*",
Retweet_NB : 200]],),
...}

La Figure 6.13 présente sous une forme graphique la ligne r_i. Elle est constituée d'une identifiant (row key) et d'une famille de colonnes.

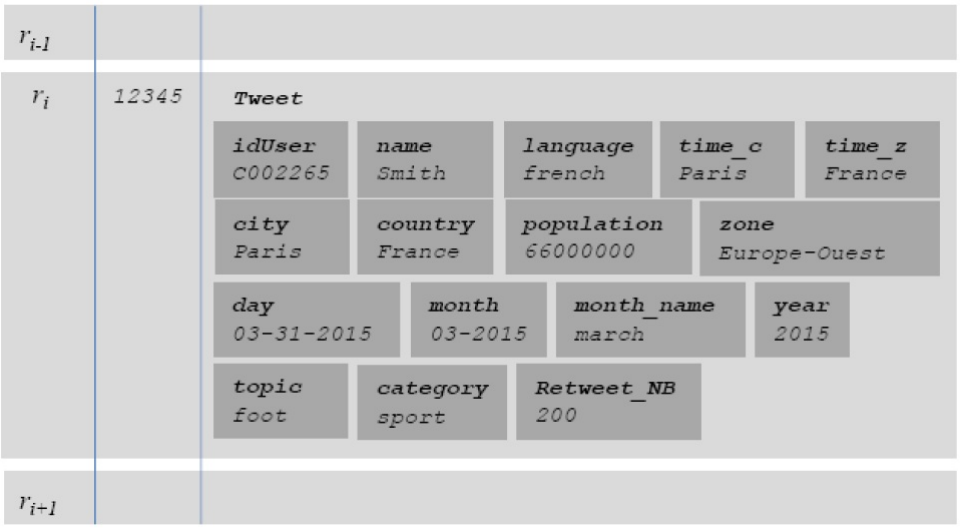


FIGURE 6.13 – exemple de ligne par traduction plate

Cette implantation évite l'utilisation des jointures entre le fait et les dimensions, mais génère un important niveau de redondance par duplication des données des dimensions entre les différentes lignes.

Processus de traduction par imbrication en orienté colonnes

Comme précédemment, à chaque fait $F \in F^S$ et ses dimensions associées $Star^S(F)$ correspond une table de même nom (N^F, E^F). Chaque instance du fait est traduite par une ligne r_i. Ce processus distribue les colonnes issues des mesures, et les colonnes issues de chaque dimension dans différentes familles de colonnes.

Une ligne r_i est définie par un schéma S_i constitué de la manière suivante :

— *id* est l'identifiant de la ligne,

— une famille de colonnes est définie pour le fait, et chaque mesure de M^F forme une colonne m_k^F imbriquée dans la famille,

— une famille de colonnes est définie pour chaque dimension, chaque attribut $a_k^{D_j}$ de la dimension Dj forme une colonne imbriquée dans la famille.

Exemple 6.8 :
Considérons la ligne décrite dans l'exemple précédent, que nous restructurons ci-dessous en fonction de l'approche par imbrication :

{...,
$ri = ([id, User[idUser, name, language, time\_c, time\_z],$
$Location[city, country, population, zone],$
$Time[day, month, month\_name, year], Subject[topic, category], Tweet[Retweet\_NB]], [id :$
12345,
$User : [idUser : "C02265", name : "Smith", language : "french", time\_c : "Paris", time\_z :$
$"France"],$
$Location : [city : "Paris", country : "France", population : 66000000, zone : "Europe -$
$Ouest"],$
$Time : [day : "03-31-2015", month : "03-2015", month\_name : "march", year : 2015],$
$Subject : [topic : "foot", category : "sport"],$
$Tweet : [Retweet\_NB : 200]],),$
...}

La Figure 6.14 présente sous une forme graphique la ligne ri constituée par un ensemble de familles de colonnes issues du fait et de chaque dimension associée.

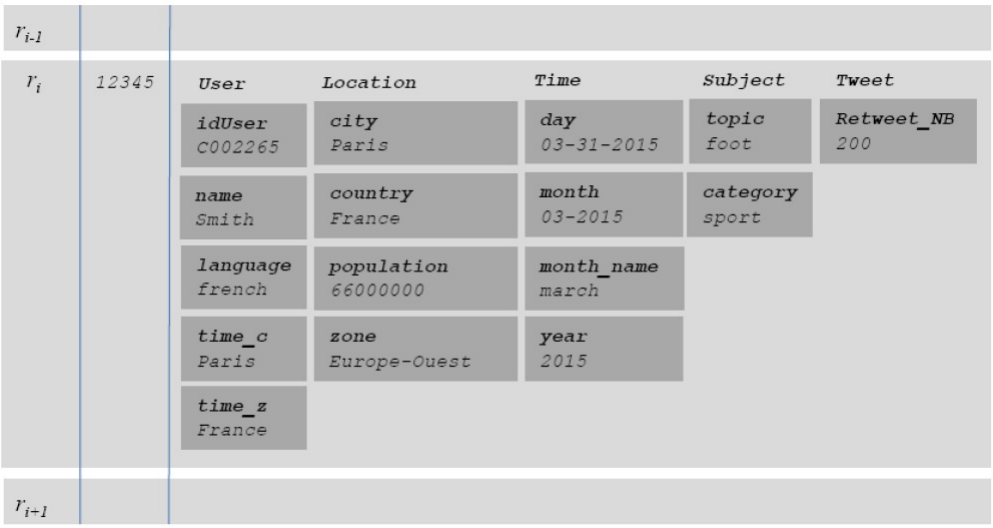


FIGURE 6.14 – Exemple de ligne par traduction imbriquée

Processus de traduction hybride en orienté colonnes

L'approche hybride consiste à décomposer au sein de la table les données issues d'un fait $F \in F^S$ et de chaque dimension associée $Star^S(F)$.

Les attributs des dimensions sont convertis en colonnes, sous une même famille de colonnes, et stockés comme une ligne de la table, à raison d'une ligne par dimension. Les mesures du fait sont converties en colonnes, sous une même famille de colonnes, et stockés sous forme de lignes de la même table. Les lignes des faits contiennent également les références aux lignes représentant les dimensions associées.

Pour chaque instance de dimension, une ligne r_i est définie par un schéma S_i constitué de la manière suivante :

— id est l'identifiant de ligne, correspondant à la racine id^{D_j} de la dimension,

— chaque attribut $a_k^{D_j}$ de la dimension D_j forme une colonne imbriquée dans une unique famille de colonnes de même nom que la dimension.

Pour chaque instance du fait, une ligne r_i est définie par un schéma S_i constitué de la manière suivante :

— id est l'identifiant de ligne,

— chaque mesure de M^F forme une colonne m_k^F imbriquée dans une unique famillede colonnes de même nom que le fait,

— pour chaque dimension $D_j \in Star^S(F)$ associée, une colonne id^{D_j} est ajoutée référençant une ligne liée issue d'une dimension.

Exemple 6.9 :
Considérons l'exemple de document précédent. Dans l'approche hybride, une ligne est constituée pour chaque instance des dimensions ainsi qu'une autre pour l'instance du fait. Nous obtenons les lignes définies ci-dessous :
{...,
$r_i^{Tweet} = ([id, idUser, city, day, topic, Retweet\_NB], [id : 12345, idUser : "C02265", city :$
$"Paris", day : "03-31-2015", topic : "foot", Retweet\_NB : 200],),$
$r_i^{User} = ([idUser, name, language, time\_c, time\_z], [idUser : "C02265", name : "Smith", language :$
$"french", time\_c : "Paris", time\_z : "France"],),$
$r_i^{Location} = ([city, country, population, zone], [city : "Paris", country : "France", population :$
$66000000, zone : "Europe - Ouest"],),$
$r_i^{Time} = ([day, month, month\_name, year], [day : "03-31-2015", month : "03-2015", month\_name :$
$"march", year : 2015],),$
$r_i^{Subject} = ([topic, category], [topic : "foot", category : "sport"],),$
...}

La Figure 6.15 présente sous une forme graphique les lignes $r_i^{Tweet}, r_i^{User}, r_i^{Location}, r_i^{Time}, r_i^{Subject}$ constituées par une famille de colonnes issues d'attributs respectivement du fait et de chaque dimension.

Cette implantation limite la redondance des données des dimensions des deux modèles précédents, mais introduit une hétérogénéité des structures des lignes placées dans la même table. L'approche hybride nécessite l'utilisation d'auto-jointures pour retrouver les documents de lignes liées aux lignes du fait.

Processus de traduction éclatée en orienté colonnes

La traduction éclatée distribue dans plusieurs tables les données : les données issues d'un fait $F \in F^S$ sont placées dans une première table, et les données de chaque dimension associée $Star^S(F)$ sont placées dans des tables distinctes (une table par dimension).

Les attributs des dimensions sont convertis en colonnes, sous une même famille de colonnes, et stockés dans une ligne de la table dédiée, T^{D_j}. Les mesures du fait sont converties en colonnes, sous une même famille de colonnes, et stockés dans une ligne

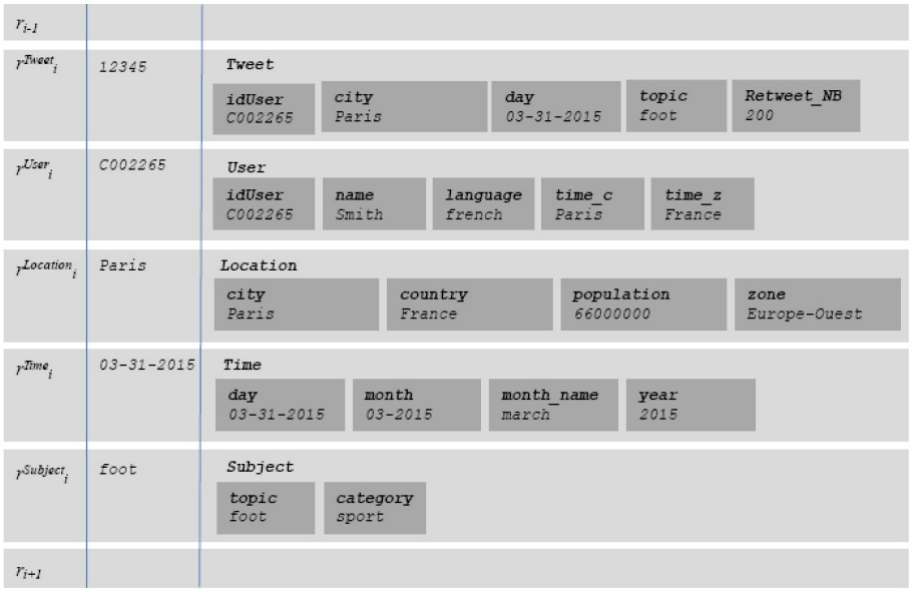


FIGURE 6.15 – Exemple de ligne par traduction hybride

d'une autre table T^F. Les lignes des faits contiennent également les références aux lignes représentant les dimensions associées, stockées dans les tables des dimensions.

Pour chaque instance de dimension, une ligne $r_i \in TD_j$ est définie par un schéma S_i constitué de la manière suivante :

— id est l'identifiant de la ligne, correspondant à la racine idD_j de la dimension,

— chaque attribut $a_k^{D_j}$ de la dimension D_j forme une colonne imbriquée dans une unique famille de colonnes de même nom que la dimension.

Pour chaque instance du fait, une ligne $r_i \in T^F$ est définie par un schéma Si constitué de la manière suivante :

— id est l'identifiant de la ligne,

— chaque mesure de M^F forme une colonne m_k^F imbriquée

— pour chaque dimension $D_j \in Star^S(F)$ associée, une colonne id^{D_j} est ajoutée référençant une ligne liée issue de la table de la dimension.

Exemple 6.10 :

Considérons l'exemple de ligne précédent. Dans l'approche éclatée, une ligne est constituée pour chaque instance des dimensions ainsi qu'une autre ligne pour l'instance du fait. Ces lignes sont placées dans des tables distinctes. Nous obtenons les lignes définies ci-dessous :_

{...,

$r^{Tweet} = ([id, idUser, city, day, topic, Retweet\_NB], [id : 12345, idUser : "C02265",$

$city : "Paris",$

$day : "03 - 31 - 2015",$

$topic :]foot[, Retweet\_NB : 200],$

$), ...$

}

{...,
$r_i^{User} = ([idUser, name, language, time\_c, time\_z],$
$[idUser : "C02265", name : "Smith", language : "french",$
$time\_c : "Paris", time\_z : "France"],$
$), ...$
}

{...,
$r_i^{Location} = ([city, country, population, zone],$
$[city : "Paris", country : "France",$
$population : 66000000, zone : "Europe - Ouest"],),$
...
}

{...,
$r_i^{Time} = ([day, month, month\_name, year],$
$[day : "03 - 31 - 2015", month : ȷ03 - 2015ȷ,$
$month\_name : "march", year : 2015],),$
...}

{
$r_i^{Subject} = ([topic, category],$
$[topic : "foot", category : "sport"],), ...$
}

La Figure 6.16 présente sous une forme graphique les lignes $r_i^{Subject}, r_i^{Time}, r_i^{Location}, r_i^{User}, r_i^{Tweet}$ appartenant respectivement aux tables $T^{Subject}, T^{Time}, T^{Location}, T^{User}, T^{Tweet}$ dont la construction est homogène pour toutes ses lignes. Les lignes sont constituées par une famille colonnes issues respectivement du fait et de chaque dimension.

Comme l'approche précédente, cette implantation réduit la redondance des données des dimensions et place les lignes dans différentes tables structurellement homogènes. L'approche éclatée nécessite l'utilisation de jointures pour retrouver les lignes de dimensions des différentes tables liées aux lignes du fait (El Malki, 2016).

6.5 Conclusion

La logique de la croissance de l'informatisation et de la globalisation d'Internet aboutit à une augmentation des volumes de données à stocker et à manipuler.

Pour cette raison et autres, les moteurs NoSQL sont maintenant un choix intéressant de gestion des données. En particulier grâce à leur excellente intégration dans les environnements de développement libre, dans des langages comme PHP ou Java, et dans tous types d'environnements, également sur les clouds.

Dans ce chapitre, nous avons exploré les différents concepts relatifs au monde NoSQL,

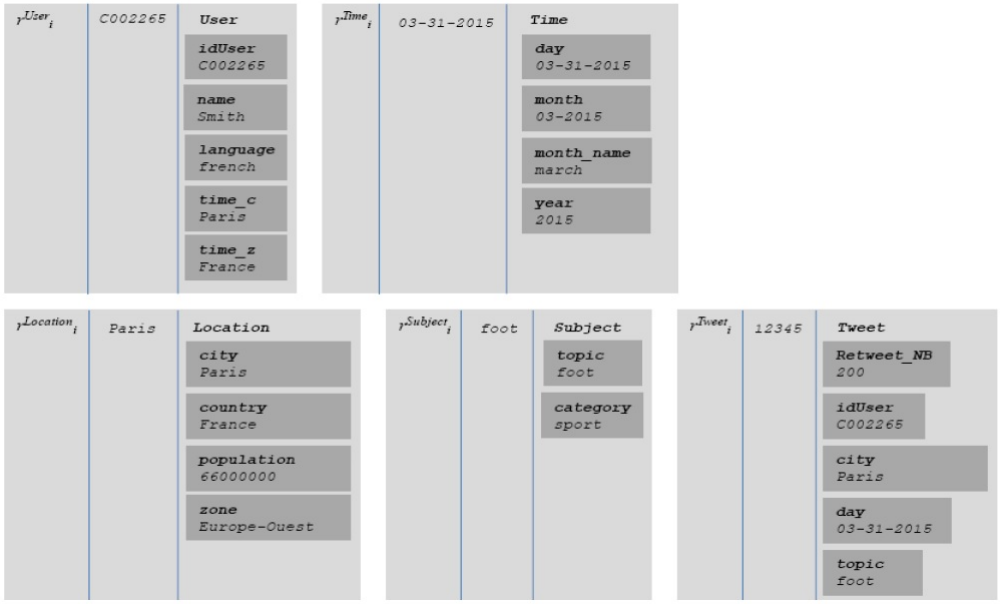


FIGURE 6.16 – Exemple de documents par traduction éclatée

avant de passer à la présentation des différents modèles d'entrepôts de données basés sur des systèmes NoSQL.

La partie suivante sera consacrée aux annexes relatives à un Panorama des Solutions Industrielles suivi par une série d'exercices.

Conclusion Générale

« Qui voit de haut voit bien, qui voit de loin voit juste.»
Victor Hugo

Ces dernières années, les systèmes décisionnels ont connu un important développement. Ils ont atteint aujourd'hui un certain niveau de maturité, permettant de comprendre les voies de développement.

Une "Bonne prise de décision" nécessite d'avoir les informations pertinentes et appropriées sur lesquelles on se base pour le choix parmi les alternatives. Dans certains cas, il est nécessaire de prendre en compte des données historiques existantes collectées auprès de diverses sources, réunies et organisées dans des entrepôts de données. Le processus d'organisation et des informations sur les différentes options est appelé processus de modélisation. Les modèles sont créés pour aider les décideurs à comprendre les différentes situations. Ils peuvent aller de représentations assez informelles à des relations mathématiques complexes.

Dans ce cours, nous proposons plusieurs solutions d'implantation des entrepôts de données dans les systèmes SQL et NoSQL. Ces alternatives offrent chacune des avantages qui tiennent compte de spécificités de ces nouveaux paradigmes caractérisés par des structures de données complexes. Ceci en prenant en compte l'ensemble des concepts de la modélisation multidimensionnelle. Les solutions que nous définissons permettent l'implantation les différents schémas de données, et leur optimisation.

La mise en place de moyens décisionnels au niveau de toute entreprise peut se faire progressivement pour des raisons techniques, d'utilisation et d'apprentissage organisationnel. Nous devons souligner l'importance de la bonne gouvernance. Il s'agit donc de piloter l'étude, la mise en place et l'utilisation de moyens décisionnels. Cela doit être collaboratif, correctif et transparent ; et qui nécessite à :

— définir des processus clés, en particulier : budget, gestion de projet, applications décisionnelles, gestion de l'information.

— fixer des objectifs : création de valeur, conformité, risques, efficacité des processus ... etc.

En fin, pour mettre en oeuvre avec succès un système décisionnel il faut procéder avec professionnalisme et rigueur au niveau des structures, des méthodes et des outils.

Annexe A. Panorama des Solutions Industrielles

Nous dénombrons un grand nombre de solutions qui sont très bien adoptées par la communauté industrielle. Dans ce qui suit nous présentons les solutions principales pour chaque modèle.

I Outils d'implémentation d'ED basés sur SQL

1. **SQL Server** Principalement conçu pour rivaliser contre des concurrents Oracle Database (DB) et MySQL. Comme tous les grands SGBDR, SQL Server prend en charge la norme ANSI SQL, le langage SQL standard. SQL Server contient également des Transats-SQL, son propre implémention SQL. SQL Server Management Studio (SSMS) (auparavant connue sous le nom de l'entreprise) est l'outil de l'interface principale de SQL Server, et il prend en charge les environnements 32 bits et 64 bits.
2. **Oracle** Oracle est un système de gestion de base de données et aussi un outil d'entreposage de données, il présente une nouvelle architecture multipropriétaires facilitant le déploiement et la gestion de Clouds 3 de base de données. Oracle est écrit en langage C et est disponible sur de nombreuses plates-formes matérielles dont : AIX (IBM), Solaris (Sun), HP/UX (Hewlett Packard), Windows NT (Microsoft), Linux
3. **Pentaho** Pentaho est un outil d'entrepôts de données utilisé pour la migration de données d'une base à une autre, l'alimentation d'un datawarehouse et de datamarts. C'est une solution d'informatique décisionnelle open source entièrement développée en Java. Elle porte sur toute la chaîne décisionnelle et utilise différents outils et composants :
   — Pour la collecte et l'intégration : les outils d'ETL (Extract-Transform-Load) Kettle ou Mondarian.
   — Pour la diffusion : un serveur d'application JBoss ou TOMCAT.
   — Pour la présentation : JFreeReport, BIRT ou encore JasperReport.
4. **Talend open studio** Talend open studio est un outil d'entrepôts de données utilisé pour l'échange Inter-Application (EAI), la migration de vieux applicatifs vers de nouveaux, le contrôle de l'information et l'alimentation du décisionnel. C'est un ensemble puissant et flexible de produits Open source pour développer,

tester, déployer et administrer des projets de gestion de données et d'intégration d'applications.

II Outils d'implémentation d'ED NoSQL

A Les solutions clé-valeur

1. Voldemort Il est sorti en tant qu'open source en 2009. Développé en java, il est actuellement en usage interne chez LinkedIn. Voldemort assure la « cohérence éventuelle » : les lectures et les écritures peuvent être effectuées directement sur les noeuds par les clients. Voldemort permet au développeur de spécifier le facteur de réplication pour chaque partie de la table de hachage. Cela implique que le développeur partitionne manuellement l'espace des clés.
2. Riak Développé par la société Basho sur le modèle de dynamo d'Amazon, Riak est un modèle NoSQL orienté clé-valeur qui offre de bonnes performances et une simplicité d'administration. Le partitionnement de la donnée se fait dans une architecture décentralisée (maitre-maitre) par hachage. Aucun serveur maitre n'est désigné et chaque noeud est indépendant. Le client peut adresser sa requête à n'importe quel noeud. Si ce dernier n'a pas la donnée, la requête est redirigée vers le noeud contenant la donnée.
3. Redis Développé en langage C, et soutenu par la suite par VMware, Redis (Remonte Dictionnary Service) est un système NoSQL orienté clé-valeur. Il est en open source. Redis est totalement distribué et gère intégralement ses données en mémoire. La distribution des données se fonde sur une architecture maître-esclave avec la particularité qu'un esclave peut être à son tour maître pour un autre esclave. Redis est un moteur très rapide grâce à son stockage en mémoire et qui via son architecture favorise l'intégrité des données.
4. Memcached Memcached est écrit en C et utilise Berkeley DB pour la persistance des données et la réplication. Memcached est un moteur orienté colonnes qui emploie une approche maître-esclave pour l'accès aux données. Memcached fonctionne avec un noeud maître unique et de multiples noeuds de calcul. Les clients peuvent accéder en lecture à un noeud quelconque dans le système mais ne peuvent écrire sur le noeud maître.

B Les Solutions orientées colonnes

1. Cassandra Cassandra est une solution libre publiée en 2009. Il s'agit d'un moteur orienté colonnes totalement distribué. Cassandra, diffère légèrement de la définition du modèle orienté colonnes puisqu'il ajoute la notion de super colonnes, c'est-à-dire un regroupement de famille de colonnes. Pour l'interrogation des données, Cassandra utilise le langage d'analyse CQL, proche du langage SQL. Toutefois CQL ne permet pas de faire des agrégations. Pour ce faire il faut définir ses propres fonctions en java par exemple..

2. HBase HBase est initiée en 2006 par l'entreprise Powerset. HBase est un moteur orienté colonnes. Il est considéré comme une contribution de Hadoop puisqu'il est interfacé avec Hadoop et ne peut fonctionner sans. Il utilise HDFS comme espace de stockage et MapReduce pour les traitements. Son développement est prévu pour la gestion de gros volumes de données dans une architecture totalement distribuée. Pour l'interrogation des données, HBase offre peu de fonctions d'analyse (scan, get, put...). Il est nécessaire d'écrire ses propres fonctions MapReduce avec des langages d'interrogations externes tels que Hive, Phoenix, Java, Python...
3. Hypertable Hypertable a été développé par la société Zvents en 2007. Cette solution open source est totalement distribuée et repose sur une architecture maître-esclave. Hypertable fonctionne sur HDFS pour tirer bénéfice de la réplication automatique des données et la tolérance aux pannes. Hypertable est écrit en C++ pour faciliter un plus grand contrôle de la gestion de la mémoire (mise en cache, la réutilisation, la récupération, etc.). Il offre la possibilité de créer et de modifier des tables d'une manière analogue à celle du langage SQL.

C Les Solutions orientées documents Le modèle orienté documents est le concurrent direct du modèle orienté colonnes, il connait un succès considérable. Plusieurs solutions ont été développées.

1. **MongoDB** La solution MogoDB a été créée en 2009 par la société 10gen. Comme les autres solutions, MongoDB est développée pour répondre aux besoins internes de la société en termes de traitement et de stockage de gros volume de données. Elle intègre son propre protocole de distribution et un langage natif d'interrogation. Dans son mécanisme de stockage, MongoDB stocke l'ensemble des collections dans des bases de données regroupées dans des espaces de noms. Chaque collection est stockée séparément et indépendamment sur disque. Pour l'interrogation des données, MongoDB offre un langage de manipulation de données très riche. De plus, il est possible de coder ses propres fonctions MapReduce. MongoDB se caractérise aussi par son propre moteur d'agrégation. Un autre avantage non négligeable est l'intégration d'un moteur de manipulation de données géospatiales.
2. **CouchDB** Développée en 2005, CouchDB est un moteur de stockage orienté documents, qui a été développé en langage Erlang. CouchDB est caractérisé par une bonne cohérence des données grâce à sa stratégie de type MVCC (Mutiversion concurency controle). Il dispose également d'un système de version pour gérer les éventuels conflits. Chaque document contient un numéro de version qui correspond au nombre de fois où le document a été modifié. Comme MongoDB les données sont stockées dans des documents de format JSON et son manipulables par des requêtes écrites en JavaScript.
3. **SimpleDB** SimpleDB est un système de stockage orienté documents dé-

veloppé depuis 2007 par Amazon et utilisé dans les offres de cloud, EC2 (Elastic Computing Cloud) et S3 (Simple Storage Service). Comme son nom l'indique, son modèle est simple. Il se contente des opérations de base (telles que Select, Delete, GetAttributes, et PutAttributes). SimpleDB est considéré comme le plus simple des systèmes de stockage orientés documents, car il n'utilise pas l'imbrication des données. Les documents sont regroupés dans des domaines (équivalent de la notion de collection dans MongoDB). Les index du domaine sont automatiquement mis à jour lorsque les attributs d'un document donné sont modifiés.

D Les Solutions orientées graphes La solution Neo4j conserve certaines caractéristiques des systèmes relationnels (transactions) mais avec une nouvelle philosophie de structuration des donnée, basée sur la théorie des graphes. Il bénéficie également d'un espace de stockage extensible et tolérant aux pannes. Neo4j est très populaire grâce à sa structure reposant sur deux objets fondamentaux : les noeuds et les relations. Le noeud définit un concept réel ou abstrait. Il peut posséder des propriétés et peut avoir des relations.

Une limite notable est que Neo4j ne repose pas sur des mécanismes de sharding. Il ne permet pas de découper les données et en faire des partitions mais en revanche il respecte les propriétés ACID.

E Systèmes relationnels extensibles Contrairement aux autres bases de données NoSQL, ce type de bases de données dispose d'un schéma relationnel prédéfini avec une interface SQL et supporte les transactions ACID. Traditionnellement conçus pour des environnements centralisés, ces systèmes peuvent etre étendus aux environnements massivement distribués.

1. **MySQL Cluster** MySQL Cluster fait partie de la version de MySQL depuis 2004, et le code est développé à partir d'un projet antérieur d'Ericsson. La solution MySQL Cluster est devenue possible en remplaçant le moteur InnoDB avec une couche distribuée appelé NDB.

   Les données sont distribuées et répliquées sur l'ensemble des noeuds pour palier à un éventuel problème d'indisponibilité d'un ou plusieurs noeuds. Depuis la version 4.1, MySQL est capable de gérer une grappe de serveurs MySQL, chaque noeud d'une grappe stocke une copie de la base de données. Le cluster MySQL prend en charge le stockage en mémoire (In-memory) ainsi que le stockage des données sur disque. Le stockage en mémoire (In-memory) permet des réponses en temps réel.

2. **VoltDB** C'est un nouveau SGBDR open-source conçu pour la haute performance (par noeud), ainsi que l'évolutivité ; il est conçu spécialement pour répondre à des besoins OLTP.

   Les fonctions d'évolutivité et de disponibilité sont compétitifs avec le cluster MySQL et les systèmes NoSQL.

3. **NuoDB** Crée en 2008 sous le nom de NimbusDB, NuoDB est une base de données relationnelle. Son architecture distribuée repose sur le modèle P2P

et une communication asynchrone.

Parmi les avantages principaux :

— Partitionnement horizontal ou vertical à la volée.

— Haute disponibilité du SGBD qui grâce à son système de réplication qui continue à fonctionner même s'il y a plusieurs pannes.

— Plusieurs API supportées : node.js, Java EE, Python...

En fin, pour mettre en oeuvre avec succès un système décisionnel il faut procéder avec professionnalisme et rigueur au niveau des structures, des méthodes et des outils.

Annexe B. Série d'exercices

Exercice 1 :

a) Donnez 3 différences entre une BD transactionnelle (OLTP) et un entrepôt de données (OLAP).

b) Décrivez le schéma en étoile que l'on retrouve dans les entrepôts de données. Précisez le rôle des différents types de tables dans un tel schéma.

c) Expliquez la différence entre les clauses CUBE et ROLLUP.

d) Identifiez un groupe d'attributs formant une hiérarchie dimensionnelle dans la table de dimension suivante : produit : idproduit(pk), description, SKU, Marque, sousCategorie, Categorie, Département, Poids, Taille, Couleur, ... Comment serait modélisée cette hiérarchie dans un schéma normalisé ?

e) à quoi sert la pré-agrégation des faits et comment peut-on implémenter cette stratégie dans le contexte des BD relationnelles ?

f) Décrivez brièvement l'architecture bus de magasins de données pour les entrepôts de données. Quel type de modélisation est normalement employé pour les magasins de données ?

g) à quoi sert la pré-agrégation des faits et comment peut-on implémenter cette stratégie dans le contexte des BD relationnelles ?

h) Dans quelle(s) situation(s) l'architecture d'entrepôts de données fédérés est-elle recommandée ?

i) Nommez deux avantages de la modélisation dimensionnelle par rapport au modèle entités-relations, dans le contexte analytique ?

j) Illustrez à l'aide d'un exemple chacune des opérations OLAP suivantes : slice, rotate, roll-up, et drill-down.

k) Quelle est la différence entre les approches ROLAP et MOLAP pour la gestion de données dimensionnelles ?

Exercice 2 : Considérez le schéma suivant :

— Film(idFilm, titre, annee, description) ;

— Personne(idPersonne, nom, dateNaissance, bio) ;

— RoleFilm(idFilm, idPersonne, personnage) ;

— GenreFilm(idFilm, genre) ;

a) Écrivez une requête SQL utilisant les fonctions analytiques d'Oracle permettant d'afficher, pour chaque genre et chaque année, le nombre de films produits depuis cette année ayant ce genre.

b) Quelle est la différence entre les résultats obtenus par les deux requêtes suivantes ?

Requête 1 :

```
SELECT F.idFilm, COUNT(*) as col
FROM Film F, GenreFilm GF
WHERE F.idFilm = GF.idFilm
GROUP BY F.idFilm
```

Requête 2 :

```
SELECT F.idFilm, COUNT(*) OVER ( PARTITION BY F.idFilm) as col
FROM Film F, Genre G
WHERE F.idFilm = G.idFilm
```

Exercice 3 :

Une agence de voyage aimerait pouvoir analyser ses données afin de planifier de meilleures campagnes de promotion auprès de ses clients. Plus particulièrement, elle aimerait analyser le nombre et le montant des ventes en fonction :

— De la destination : hôtel, ville, pays, région, catégorie de région (ex : bord de mer, alpine, etc.), catégorie de destination (ex : familial ou non), catégorie hôtel (ex : 1-4 étoiles) ;

— De la date d'achat : jour de l'année, jour de la semaine, mois, année, saison touristique (ex : basse ou haute saison) ;

— De la date de départ : jour de l'année, jour de la semaine, mois, année, saison touristique (ex : basse ou haute saison) ;

— Du forfait : nombre de personnes, nombre de nuits, type de forfait (ex : tout inclus, repas inclus, etc.), type de chambre (ex : standard, suite, penthouse, etc.) ;

— Du client : groupe d'âge, sexe, adresse, type d'acheteur (ex : nouveau, récurrent, etc.) ;

— Du canal de vente : catégorie (ex : magasin, internet, etc.) ;

— De la promotion : catégorie (ex : 2 pour 1, rabais 10%, rabais 25%, etc.), début et fin de validité ;

— Du mode de paiement : catégorie (ex : crédit, comptant, etc.) ;

a) Proposez un schéma en étoile permettant de faire ces analyses. Identifiez clairement les clés primaires et étrangères des tables de faits et de dimension ;

b) Identifiez, pour chaque table de dimension, une hiérarchie de niveaux de granularité (e.g., attribut1 $\leftarrow$ attribut2 $\leftarrow \dots$) ;

c) Proposez une stratégie d'agrégation ajoutant une nouvelle table de faits agrégés. Donnez le code SQL permettant de créer cette nouvelle table.

Exercice 4 : TU Hôtels est une petite chaîne d'hôtels ayant des propriétés dans plusieurs états américains. L'entreprise possède une base de données centralisée pour stocker et faire

le suivi des réservations de ses clients. En 2008, ils ont installé des restaurants appelés Café in the Hotel dans plusieurs de leurs hôtels. Un système est employé pour faire le suivi des commandes et les relayer aux employés dans les cuisines. TU Hôtels aimerait utiliser les données qu'ils ont emmagasinées pour mieux comprendre la performance de leurs hôtels et restaurants. Ils ont également accès à une base de données de critiques de clients provenant du site web HotelComplainer.com. La tâche est de faire la conception de deux magasins de données (data marts) utilisant les données provenant des trois sources mentionnées ci-haut. Vous devrez faire un schéma en étoile pour chaque magasin de données en choisissant les dimensions, les faits, et les attributs à partir des sources, dont le schéma est fourni ci-dessous. La table suivante présente les questions analytiques auxquelles devra répondre vos magasins de données :

Data mart 1 : Performance des hôtels

— Durant quel mois y a-t-il le plus grand nombre de réservations de chambre ?

— Quelle est la saison morte pour les hôtels situés dans une région particulière ?

— Quels hôtels génèrent le plus de revenus (non attribuables aux restaurants) ?

— Quel est la durée moyenne des séjours dans les hôtels de 4.5 étoiles ou plus ?

— Les fumeurs restent-ils plus longtemps que les non-fumeurs ?

— Pour un hôtel donné, combien y a-t-il de clients provenant d'un autre état ?

Data mart 1 : Performance des hôtels

— Quels restaurants génèrent le plus de revenus ?

— Les restaurants les mieux cotés génèrent-ils plus de revenus ? Quel est l'item commandé le plus souvent dans une région particulière ?

Pour compléter l'exercice, vous devrez suivre les étapes suivantes :

1. Identifiez le principal évènement d'affaires pour chaque magasin de données ; Posez-vous la question suivante : "Quel est l'évènement d'affaires qui génère la (les) métrique(s) de performance ?
2. Identifiez les attributs associés aux faits. Posez-vous la question suivante : "Comment l'évènement d'affaires est-il mesuré ?"
3. Identifiez les dimensions et leurs attributs. Posez-vous la question suivante : "Quelles données caractérisent les différents aspects de l'évènement d'affaires ?"
4. Élaborez le schéma en étoile selon les principes vus en classe.

Exercices supplémentaires

Exercice 5 :

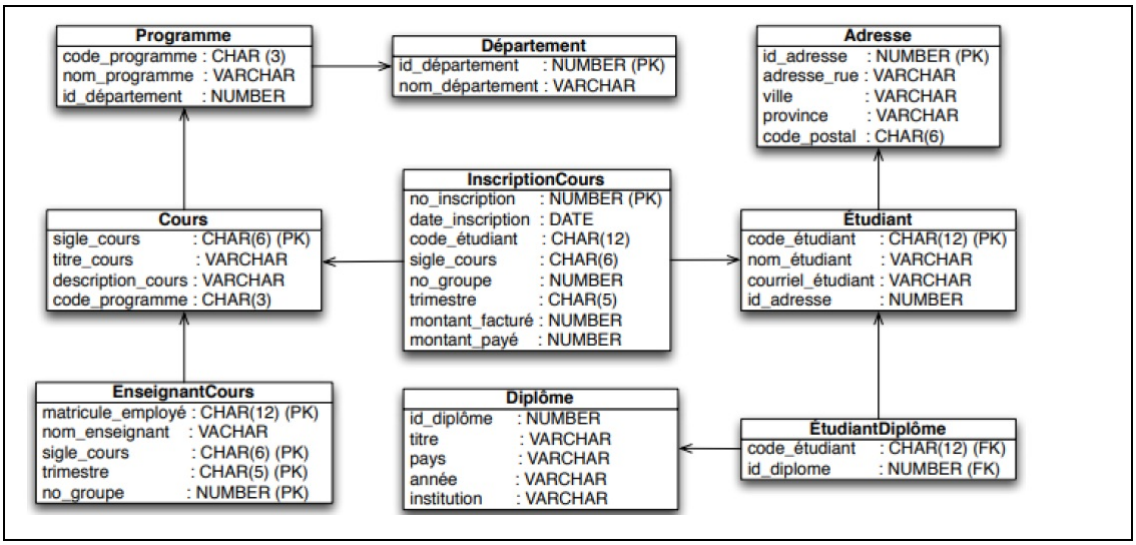
L'École de Technologie Infinie (ETI) désire développer un magasin de données afin d'améliorer la gestion de ses programmes d'enseignement. La principale source d'informations serait la base de données du registraire, dont le schéma relationnel est fourni ci-dessous. Le magasin de données devrait permettre de répondre, entre autres, aux questions analytiques suivantes :

— Quels sont les cours d'un certain programme ayant le plus (le moins) d'inscriptions ?

— Quel est le nombre moyen d'étudiants par groupe ?

— Quels enseignants ont eu les plus gros (plus petits) groupes d'étudiants ?

— Quelle est la proportion d'étudiants, ayant suivi les cours d'un certain programme, qui habitent à l'extérieur de Montréal ?



Exercice 6 :

Il s'agit de modéliser le Datawarehouse des ventes d'une entreprise commerciale. Cette entreprise vend des produits regroupés par familles de produits. Une vente correspond à un produit et un seul ; la vente est effectuée par l'un des vendeurs du service de vente spécialisé dans le produit. Le Datawarehouse doit pouvoir fournir le chiffre d'affaires des ventes d'un produit, par date, client, et vendeur, ainsi que toutes les sommations possibles de chiffre d'affaires.

Les objets Du Datawarehouse sont les suivants :

— produit, caractérisé par : code_produit, code_famille, etc...

— client, caractérisé par : code_client, nom, CSP (catégorie socio-professionnelle), etc ...

— vente, caractérisée par : code_date, code_produit, code_client, code_vendeur, Chiffre d'affaires

— vendeur, caractérisé par : code_vendeur, nom, code_service, etc...

— date, caractérisée par : code_dat, semaine, mois, année, etc...

1. Donner les définitions des termes suivants : table de faits, table de dimension, indicateur, hiérarchie.
2. Tracer le schéma en étoile dimensionnel du Datawarehouse, en précisant pour chaque table sa nature dimensionnelle (table de faits ou table de dimension), ses clés, ainsi que la nature des champs.

Exercice 7 : Une université cherche à étudier les facteurs influant sur la réussite de ses étudiants aux examens. Pour cela elle décide de construire un Datawarehouse. Elle souhaite pouvoir répondre aux questions suivantes :

— Quel est le nombre de réussites aux examens par cours, pour l'année 2007 ?

— Quel est le nombre de réussites aux examens d'un cours obligatoire, pour l'année 2007 ?

— Quel est le nombre de réussites aux examens par sexe (féminin, masculin), pour l'année 2007 ?

— Combien d'étudiants ayant un âge de 22 ans ont réussi leurs examens de base de données relationnelle ?

— Quel est le nombre de réussites aux examens pendant le semestre d'hiver 2006 ?

Pour cela elle dispose des données suivantes : Pour chaque examen passé, on connaît l'âge et le sexe de l'étudiant, le nom du cours (les cours peuvent être regroupés en cours obligatoire et cours à option), la date de l'examen, la note obtenue et si l'examen est réussi ou non.

Proposez un modèle en étoile pour cette application. Recherchez tout d'abord les différentes dimensions et proposez une hiérarchie pour ces dimensions.

Exercice 8 :

Un distributeur (grossiste) approvisionne plusieurs magasins en produits, en effectuant au plus une livraison par jour et par magasin. Les informations qui figurent sur chaque bon de livraison sont les suivantes : le numéro du bon de livraison, la date de livraison, la référence du magasin, et pour chaque type de produit livré sa référence et la quantité livrée (le nombre d'articles). Ces informations sont stockées chez le distributeur, et accumulées pendant des longues périodes afin de les analyser pour améliorer le service de distribution.

Les analyses se font suivant plusieurs axes, et à plusieurs niveaux, en analysant les mouvements des produits par jour et par mois, par ville et par région, par fournisseur et par catégorie de produit.

On supposera qu'un fournisseur peut fournir au distributeur des produits dans plusieurs catégories et qu'une catégorie de produit peut être fournie par plusieurs fournisseurs.

Définir le schéma dimensionnel du Datawarehouse permettant d'analyser la quantité livrée par ville et catégorie de produits, en faisant apparaître clairement les dimensions et les indicateurs.

Proposition de solutions Sol-Exercice 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.

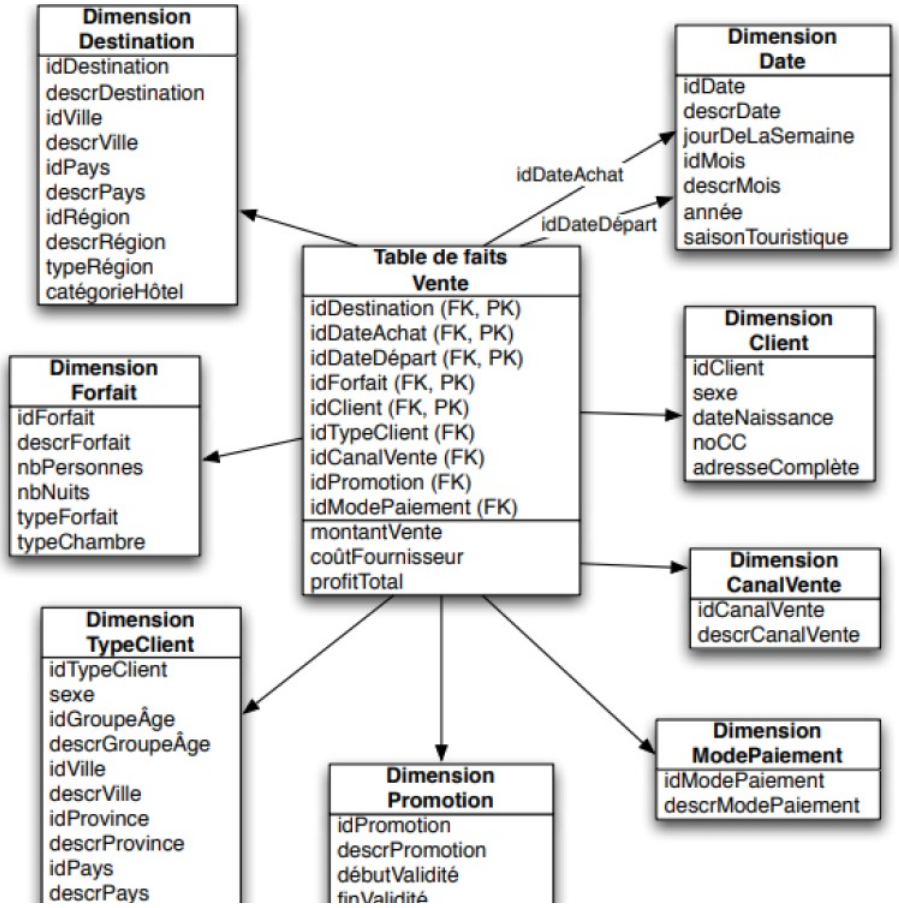
c) CUBE : Regroupe successivement les lignes d'une table selon chaque sous-ensemble de Colonnes.

Sol-Exercice 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.

Sol-Exercice 3 :



a

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 précalculant 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
```

Sol-Exercice 4 :

Solution pour le datamart No1

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.
   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.
   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.

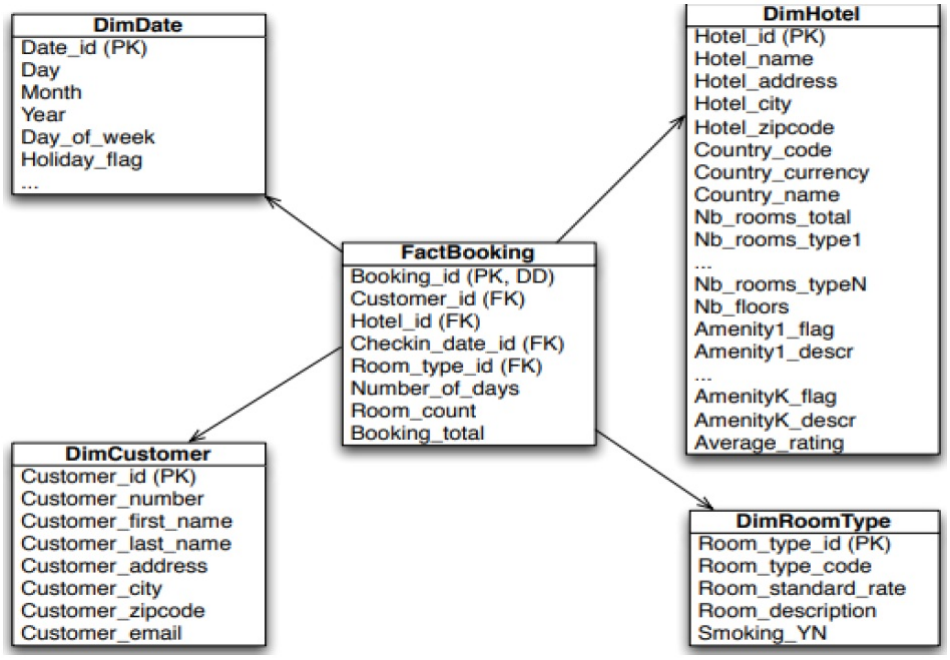
Dimension DateArrivée (DimCheckinDate) :

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**



Bibliographie

D. Akon, A. Fekete, and U. Rohm. Scalable transactions across heterogeneous nosql key-value data stores. In *Proceedings of VLDB Endow*, 2013.

R. Aniceto, R. Xavier, V. Guimaraes, F. Hondo, M. Holanda, M.E. Walter, and S. Lifschitz. Evaluating the cassandra nosql database approach for genomic data persistency. *International Journal of Genomics*, 2015.

Y. Bédard and J. Han. Fundamentals of spatial data warehousing for geographic knowledge discovery. dans geographic data mining and knowledge discovery, 2nd edition, h. j. miller and j. han, chap. 3, research monographs in geographic information systems. taylor and francis. 2008.

Y. Bédard and J. Han. Geographic data mining and knowledge discovery, chapter fundamentals of spatial data warehousing for geographic knowledge discovery. taylor and francis. 2009.

Y. Bédard and M. Rivest, S.and Proulx. Spatial online analytical processing (solap) : Concepts, architectures, and solutions from "data warehouses and olap : concepts, architectures, and solutions from a geomatics engineering perspective. dans r. wrembel, and c. koncilia (eds.), data warehouses and olap : Concepts, architectures and solutions (pp. 298-319). hershey, pa : Igi global. 2007.

L. Bellatreche. Utilisation des vues matérialisées, des index et de lafragmentation dans la conception logique et physique d'un entrepôt de données, université clermont-ferrand, france, 2000.

S. Bimonte, A. Tchounikine, and M. Miquel. Towards a spatial multidimensional model. In *8th ACM international workshop on Data warehousing and OLAP. 39-46. Bremen.* Germany, a.

S. Bimonte, A. Tchounikine, and M. Miquel. Spatial olap : Open issues and a web based prototype. In *10th AGILE International Conference on Geographic Information Science Aalborg University, Denmark pp 1-11.* Denmark, b.

K. Boulil. Une approche automatisée basée sur des contraintes d'intégrité définies en uml et ocl pour la vérification de la cohérence logique dans les systèmes solap : Applications dans le domaine agri-environnemental. thèse de doctorat en informatique. université blaise pascal - clermont-ferrand ii, france, 2012.

K. Boulil, S. Bimonte, and F. Pinet. A uml and spatial ocl based approach for handling quality issues in solap systems. In *Proceedings of the14th International Conference on Enterprise Information Systems. Wroclaw, Poland : 6*. ICEIS 2012.

K. Boulil, S. Bimonte, and F. Pinet. Un modèle uml et des contraintes ocl pour les entrepôts de données spatiales. de la représentation conceptuelle à l'implémentation. *Ingénierie des Systèmes d'Information*, 16(6) :11–39, 2011.

K. Boulil, S. Bimonte, and F. Pinet. Conceptual model for spatial data cubes : A uml profile and its automatic implementation. *Computer Standards and Interfaces*, 38 : 113–132, 2015.

K. Boulil, S. Bimonte, and F. Pinet. Un cadre conceptuel basé sur uml et spatial ocl pour la définition des contraintes d'intégrité dans les systèmes solap. *Revue Internationale de Géomatique*, 26(1) :97–131, 2016.

Scabora Lucas C., Jaqueline J. Brito, Ricardo Rodrigues Ciferri, and Cristina Dutra de Aguiar Ciferri. Physical data warehouse design on nosql databases - olap query processing over hbase. In *In Proceedings of the 18th International Conference on Enterprise Information Systems Pages 111-118, Rome, Italy — April 25 - 28*, 2016.

A. Castelltort and A. Laurent. Nosql graph-based olap analysis. *SCITEPRESS - Science and and Technology Publications*, page 217–224, 2014.

R. Cattell. Scalable sql and nosql data stores. *SIGMOD Rec*, 39(4) :12–27, 2011.

F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber. A distributed storage system for structured data. *ACM Trans Comput Sy*, 4(1) :4–26, 2008.

S. Chaudhuri and V. Narasayya. Index merging. In *Proceedings of the International Conference on Data Engineering (ICDE)*. ICDE 1999.

M. Chevalier, M. El Malki, A. Kopliku, T. Olivier, and R. Tournier. Benchmark for olap on nosql technologies comparing nosql multidimensional data warehousing solutions. In *In 9th IEEE International Conference on Research Challenges in Information Science, RCIS, IEEE, 480–485*, 2015.

Li. Chongxin. Transforming relational database into hbase : A case study. In *In : Software Engineering and Service Sciences (ICSESS), IEEE. pp 683-687*, 2010.

E. F. Codd. *Providing OLAP (On-line Analytical Processing) to User-analysts : An IT Mandate*. IBM, in technical reports edition, 1993.

M. L. Damiani and S. Spaccapietra. Spatial data warehouse modelling. processing and managing complex data for decision support. *Hershey, IGI Global*, pages 1–27, 2006.

A. Datta, K. Ramamritham, and H. Thomas. Curio : A novel solution for efficient storage and indexing in data warehouses. In *Proceedings of the International Conference on Very Large Databases*, 1999.

K. Dehdouh, O. Boussaid, and F. Bentayeb. Columnar nosql star schema benchmark. in yamine ait ameur, ladjel bellatreche, and george a. papadopoulos, eds. model and data engineering. *Lecture Notes in Computer Science. Springer International Publishing*, 2014.

M. EL MALKI. Modélisation nosql des entrepôts de données multidimensionnelles massives. thèse de doctorat en informatique, université toulouse jean jaures, france, 2016.

E. Eric. *Domain-driven design : tackling complexity in the heart of software.* Addison-Wesley Professional, 2004.

L. George. *HBase : The Definitive Guide.* O'Reilly Media, Inc, 2011.

O. Glorio and J. Trujillo. An mda approach for the development of spatial data warehouses. data warehousing and knowledge discovery. *I.-Y. Song, J. Eder and T. Nguyen, Springer Berlin / Heidelberg*, 5182 :23–32, 2008.

J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube : A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Min. Knowl. Discov*, 1(1) :29–53, 1997.

W.H. Inmon. *Building the Data Warehouse.* 4th Edition, Wiley press, 2005.

Goglin J.-F. *La construction du datawarehouse : du datamart au dataweb. Hermes.* 2ème édition, Hermes, 2001.

R. Kimball. *The data warehouse toolkit.* John Wiley and Sons, 1996.

R. Kimball and M. Ross. *Entrepôts de données, Guide pratique de modélisation dimensionnelle.* Vuibert Informatique Press, Paris, France, 2003.

S. Lujan-Mora, J. Trujillo, and I.Y. Song. A uml profile for multidimensional modeling in data warehouses. *Data and Knowledge Engineering*, 59(3) :725–769, 2006.

E. Malinowski and E. Zimanyi. Hierarchies in a rnultidimensional model : From conceptual modeling to logical representation. *Data and Knowledge Engineering*, 59(2) :348–377, 2006.

E. Malinowski and E. Zimanyi. *Advanced Data Warehouse Design : From Conventional to Spatial and Temporal Applications.* Springer, 2008. ISBN 978-3-540-74404-7.

J.N. Mazon, J. Lechtenbörger, and J. Trujillo. A survey on summarizability issues in multidimensional modeling. *Data and Knowledge Engineering*, 68(12) :1452–1469, 2009.

L. Naoum. Un modèle multidimensionnel pour un processus d'analyse en ligne de résumés flous. thèse de doctorat, université de nantes, france., 2006.

T.B. Pedersen and N. Tryfona. Pre-aggregation in spatial data warehouses. In *7th International Symposium on Advances in Spatial and Temporal Databases, Springer-Verlag : 460-480*, 2001.

T.B. Pedersen, C.S. Jensen, and C.E. Dyreson. A foundation for capturing and querying complex multidimensional data. *Information Systems*, 26(5) :383–423, 2001.

F. Pinet. Modélisation des contraintes d'intégrité dans les systems d'information environnementaux. habilitation à diriger les recherches. université blaise pascal, clermont-ferrand : 112, france, 2010.

F. Pinet and M. Schneider. A unified object constraint model for designing and implementing multidimensional systems. *Journal on Data Semantics*, 13 :37–71, 2009.

F. Ravat, O. Teste, and Zurfluh G. Manipulation et fusion de données multidimensionnelles. *In RNTI-E-3 Revue des Nouvelles Technologies de l'Information, réd., Extraction et Gestion des Connaissances*, 1, 2005.

S. Rivest, Y. Bedard, M. Proulx, M. Nadeau, F. Hubert, and J. Pastor. Solap technology : Merging business intelligence with geospatial technology for interactive spatio-temporal exploration and analysis of data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(1), 2005.

S. Saichi. *Optimisation de requêtes dans les entrepôts de données.* Mémoire de Magister en informatique, Université Oran1, Algérie, 2009.

M. Salehi. Developing a model and a language to identify and specify the integrity constraints in spatial datacubes. thèse de doctorat, faculté des études supérieures de l'université laval, canada, 2009.

M. Salehi, Y. Bédard, , and S. Rivest. Formal conceptual model and definition framework for spatial data cubes. *Geomatica*, 64 :313–326, 2010.

M. Scavuzzo, E. Di Nitto, and S. Ceri. Interoperable data migration between nosql columnar databases. In *In 18th EDOCW, pages 154–162. IEEE*, 2014.

S. Scherzinger, Meike K., and Uta S. Managing schema evolution in nosql data stores. In *ArXiv13080514 Cs (August*, 2013.

N. Stefanovic, J. Han, and K. Koperski. Object-based selective materialization for efficient implementation of spatial data cubes. *IEEE Transactions on Knowledge and Data Engineering*, 12(6) :938–958, 2000.

A. Steve, B. Patrick, and M. Luis. From relational database to column-oriented nosql database : Migration process. *International Journal of Engineering Research and Technology (IJERT)*, 4(5), 2015.

Red Brick Systems. *Star schema processing for complex queries.* Papier Blanc, 1997.

O. Teste. Modélisation et manipulation d'entrepôts de données complexes et historisées. thèse de doctorat en informatique, université paul sabatier - toulouse iii, france, 2000.

Sellis T. Theodoratos, D. Datawarehouse schema and instance design. In *17th International Conference on Conceptual Modeling*, pages 363–376, 1998.

P. Valduriez. *Join indices.* ACM Transactions on Database Systems, 1987.

M-C. Wu and A. Buchmann. Research issues in data warehousing. In *In Datenbanksysteme in Bro Technik und Wissenschaft (BTW'97*, 1997).

C. Xi-Qian, C. Zhong-Xian, and C. Xiu-Kun. Applying dp to etl of spatial data warehouse. machine learning and cybernetics. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, 2004).