Les entrepôts de données pour le décisionnel: Concepts et notions de base

Sarah NAIT BAHLOUL

Avant-propos

Ce polycopié est le support écrit du cours "*Bases de Données Multidimensionnelles et Entrepôts de données*" dispensé aux étudiants de première année Master de la spécialité "*Systèmes d'Information et Données*". Il est destiné aux étudiants suivant ce cours ou à toute personne souhaitant découvrir le monde décisionnel.

Il porte sur l'introduction des concepts clés du monde décisionnel : définition, architecture, modélisation, étapes de mise en place, stockage et interrogation de données. Le but de ce polycopié est de sensibiliser l'étudiant à l'importance des différentes étapes lors de la mise en oeuvre de projets décisionnels. Plus particulièrement, à l'importance de fixer clairement au préalable les objectifs à atteindre et d'avoir une vue globale sur les ressources (données, logiciels,...) disponibles. Ces deux éléments sont à la base du choix de la solution à mettre en place.

Bien évidemment, chacune des notions présentées dans ce polycopié ne sera vue que de manière introductive, mais l'ensemble de ces chapitres se veut représenter un bagage informatique minimal pour tout futur consultant décisionnel. Ce cours a comme pré-requis des notions élémentaires des bases de données et du langage de requêtes SQL.

Le polycopié se veut axé sur la présentation théorique des concepts de l'Informatique Décisionnel. Il n'y aura donc pas (ou très peu) de références technologiques. Un autre polycopié sera dédié à la présentation de quelques outils phares utilisés dans la mise en place d'un systèmes d'information décisionnel et de son exploitation.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Depuis la démocratisation des ordinateurs, toute information utile dans une organisation est stockée et répertoriée dans un système d'information. Ces systèmes d'informations dits opérationnels (SIO) ont pour seul objectif d'aider à la réalisation des activités au sein de l'organisation en garantissant la cohérence des données et en réduisant l'espace de stockage.

Depuis, d'autres besoins sont apparus parmi lesquels le besoin de pouvoir expliquer des résultats, le besoin de mieux répondre aux exigences des clients, le besoin d'optimiser le rendement d'une entreprise, etc. L'analyse de données est apparue pour répondre à ces besoins. Cette nouvelle discipline est vue comme un processus qui à partir d'hypothèses et de données permet à une personne de générer de la connaissance. À partir de ces connaissances, il est possible de prendre des décisions sur les nouvelles stratégies à adopter pour impacter positivement l'entreprise. Cette prise de décision implique de pouvoir au préalable se poser les bonnes questions et de savoir ou et comment chercher. C'est dans ce cadre qu'est apparu le domaine décisionnel.

Prenons un exemple pour illustrer ce besoin décisionnel. Les systèmes mis en place dans les universités permettent principalement d'enregistrer les activités relatives à ses acteurs (personnels, enseignants, étudiants, etc.). Il existe plusieurs systèmes d'informations opérationnels pour une seule université. Si cette dernière souhaite mettre en place de nouvelles stratégies pour améliorer son rendement et son image, elle doit se lancer dans un processus d'analyse de données. Quelle est le taux de réussite des étudiants ? Quelles sont les raisons qui font qu'un département X a de meilleurs résultat qu'un département Y ? Quel est le rendement en recherche ? Que faut-il changer pour améliorer les résultats ?, etc.

Les systèmes d'informations opérationnels ne peuvent pas et n'ont pas comme objectif de répondre à ce type de questions. Pour ce faire, il est important de mettre en place un nouveau système regroupant toutes les données utiles afin de pouvoir les analyser et en extraire les réponses. Ce type de système est appelé système d'information décisionnel.

Avant de présenter les différentes étapes pour mettre en place un système décisionnel, nous revenons en détail sur les systèmes d'informations opérationnels et leurs limites.

1.1 Système d'information opérationnel

Les systèmes d'information opérationnels, ou encore **transactionnels**, constituent ce qu'on appelle les systèmes **OLTP** (On-Line Transactional Processing). Ils sont destinés à assurer la prise en charge d'un grand nombre de requêtes peu complexes sur une quantité faible de données. Ces systèmes sont robustes performants et sécurisés et pour cause, ils se basent sur :

— Les propriétés ACID (Atomicité, Consistance, Isolation et Durabilité) d'une transaction pour assurer la cohérence des données et la continuité du système.

— Un modèle entité-association et sa normalisation pour représenter les données de manière cohérente et sans redondances, permettant ainsi l'exécution efficace d'un grand ensemble de requêtes peu complexes (Ajout, suppression, mise à jour, etc).

1.1.1 Modèle entité-association et normalisation

Le modèle entité-association constitue l'un des premiers et des plus courants modèles conceptuels de données. Il permet une description naturelle du monde à partir de concepts d'entités et d'associations. La figure 1.1 illustre de manière simplifiée un exemple réduit de modèle entité-association relatif à un système d'information universitaire.

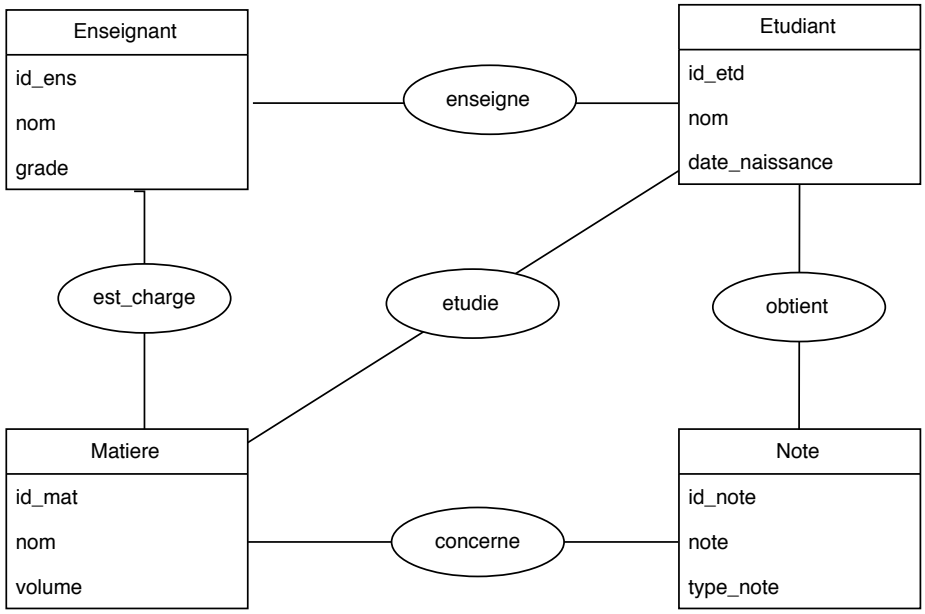


FIGURE 1.1: Exemple d'un modèle entité-association

Les acteurs du système sont représentés en entités : Enseignant, Étudiant, etc. Et la relation entre ces acteurs est représentée par des associations : Enseigne, Etudie, etc.

Quant à la normalisation, son objectif est de construire un schéma de données en supprimant les redondances. Cela garantit la cohérence en éliminant les risques d'erreurs lors des opérations de mise à jour. En effet, ces dernières sont souvent effectuées dans un SIO. Le niveau de normalisation (il en existe 8) est choisi selon que les tables de la base sont plus sollicitées en lecture ou plus en écriture. Une bonne conception d'une base de données dans un SIO doit être au minimum en 3eme forme normale car plusieurs écritures sont effectuées.

1.1.2 Limites des SIO dans un contexte décisionnel

Aujourd'hui, avec l'afflux continu de données, aussi bien internes qu'externes, un certain nombre de nouveaux besoins sont apparus :

— Dans un marché de plus en plus concurrentiel et globalisé, le besoin de prendre la bonne décision au bon moment est devenu crucial.

— Les besoins analytiques sont devenus complexes et nécessitent de manipuler un grand ensemble de données à la fois.

— Le besoin de garder l'historique de toutes les données transitant dans le système.

Les SIO tels qu'ils sont conçus peinent à répondre à ces nouveaux défis et pour cause :

— Le modèle entité-association en 3eme forme normale du SIO a été pensé pour assurer la cohérence de données et la réalisation de transactions sans erreurs. Il est très performant pour répondre a un nombre élevé de requêtes simples mais n'est pas adapté pour des requêtes analytiques trop complexes nécessitant un grand nombre de jointures.

— Un SIO doit être cohérent à un instant T et pour cela, les données enregistrées sont souvent modifiées et parfois supprimées sans que les anciennes valeurs de ces données ne soient conservées, alors que ces dernières constituent un ensemble d'informations important pour prendre des décisions.

— En règle générale, dans une organisation, le développement se fait de manière verticale. Autrement dit, quand un nouveau besoin se présente, un nouveau SIO est développé pour y répondre. Par conséquent, une organisation regroupe généralement plusieurs SIO répondant chacun à un besoin différent. Ces SIO travaillent et évoluent de manière indépendante dans des environnements bien hétérogènes. Dans un contexte décisionnel, il est important de regrouper les informations dans un seul endroit sous une même logique pour pouvoir les exploiter de manière optimale.

Afin de pallier les limites des SIO, un nouveau système d'information a été pensé pour répondre aux besoins décisionnels. Les systèmes d'informations décisionnels, par opposition aux systèmes d'information opérationnels, ont pour vocation de répondre aux besoins des décideurs d'une manière fiable et rapide.

1.2 Système d'information décisionnel

Les systèmes d'information décisionnels (SID) sont des systèmes englobant un ensemble d'outils permettant l'organisation d'ensemble de données de façon spécifique, facilement accessibles et ont pour vocation la prise de décision. Ils constituent ce qu'on appelle les systèmes **OLAP** (On-Line Analytical Processing). Dans ce type de système, les analystes, les responsables et le personnel exécutif peuvent mieux comprendre les données via un accès rapide, consistant et interactif à une large variété de vues possibles de l'information. OLAP transforme les données du plus bas niveau afin de montrer leur véritable dimension dans l'entreprise, selon la compréhension et le point de vue de chaque utilisateur. Les systèmes OLAP utilisent une vue multidimensionnelle de données agrégées pour fournir un accès rapide à l'information stratégique pour des analyses plus poussées.

1.2.1 Modèle multidimensionnel

Face aux limites du modèle entité-association, une nouvelle modélisation a vu le jour : la modélisation multidimensionnelle. Son inventeur, Ralph Kimball, propose une nouvelle manière de représenter les données en fonction des besoins analytiques. Pour cela, il introduit de nouveaux concepts :

— Les activités et les évènements dans une organisation sont décrits par des **faits**.

— La mesure d'un fait est décrite par un **indicateur** appelé aussi **mesure**.

— Le contexte de la réalisation d'un fait est décrite par un ensemble de **dimensions**.

Afin de représenter les informations dans un modèle multidimensionnel il est important pour les concepteurs du SID de collaborer avec les décideurs (utilisateurs du SID) eux mêmes en prenant en compte leurs visions et leurs besoins.

En reprenant l'exemple du système d'information universitaire, la figure 1.2 représente le modèle multidimensionnel dédié à l'analyse des résultats de passage des étudiants au sein de l'université.

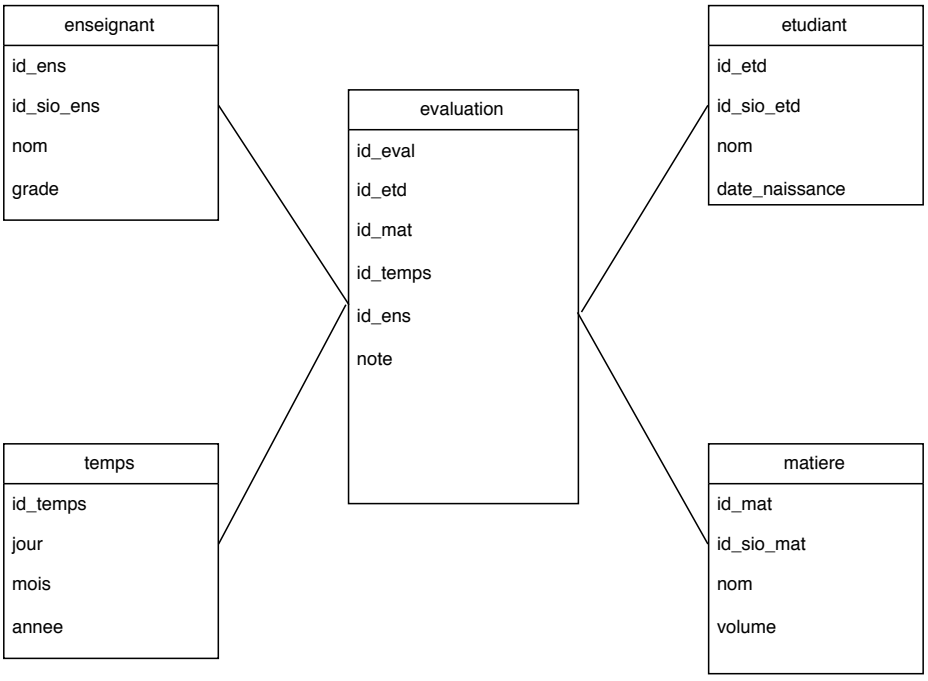


FIGURE 1.2: Exemple d'un modèle multidimensionnel

1.2.2 Vue globale d'un SID

Nous illustrons dans la figure 1.3 les différents composants d'un système d'information décisionnel.

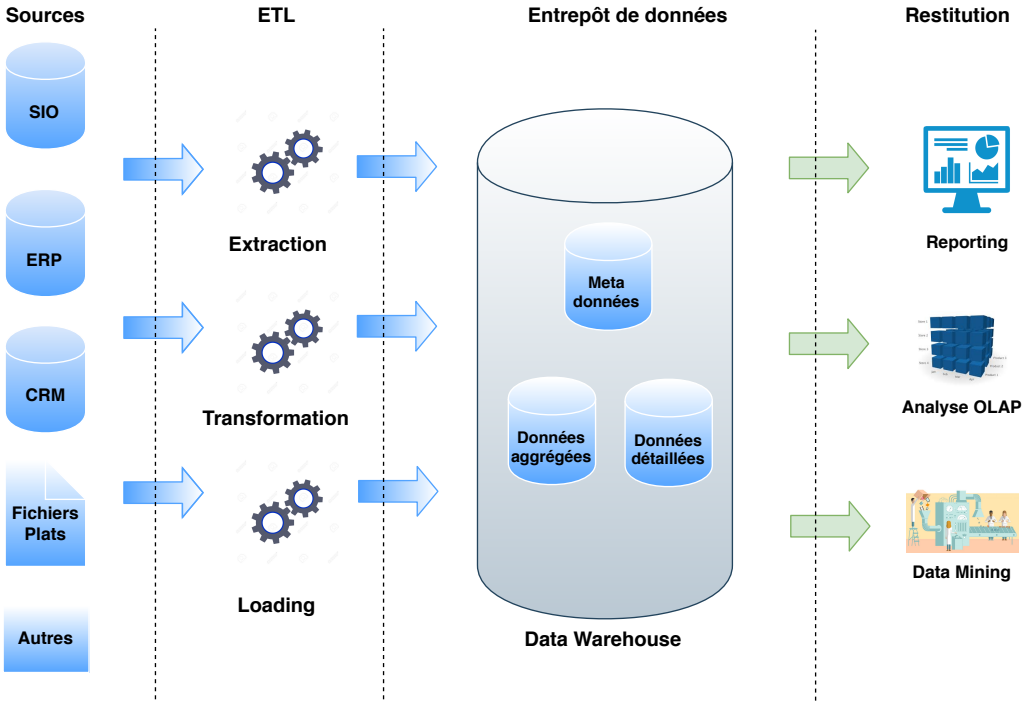


FIGURE 1.3: Architecture globale d'un SID

— **Sources** : Composée de données hétérogènes provenant des différents SIO de l'organisation.
— **Alimentation** : Composée d'outils ETL (Extract, Transforme and Load) pour extraire et transformer les données à partir des sources et les charger dans le modèle décisionnel.
— **Stockage** : Collection de données intégrées, organisées et historisées pour la prise de décision sous forme d'un entrepôt de données (DataWarehouse) et/ou de magasins de données (DataMarts).
— **Restitution** : composée d'applications pour l'exploitation de données (Reporting, indicateurs,...).

1.2.3 Mise en place d'un SID

Le processus de mise en place un système d'information décisionnel est composé de quatre différentes étapes.

1. **Étude des besoins et de l'existant** Lors de cette phase, il est important de dégager les buts recherchés, les questions auxquelles le système à mettre en place doit répondre. À partir de là, il faut déterminer et recenser les données qui jouent un rôle important.
2. **Modélisation et conception** La deuxième étape consiste à choisir la modélisation adéquate de l'entrepôt de données et l'architecture qui répond aux exigences techniques et organisationnelles de l'organisation.
3. **Mise en place de l'entrepôt de données** Cette étape consiste à mettre en place techniquement l'entrepôt, à savoir, choisir la stratégie de stockage à adopter et les outils d'intégration à mettre en place pour, l'extraction, la transformation et le chargement des données.

4. **Mise en place d'outils d'exploitation**
   - **Requêtes analytiques** : permettent aux utilisateurs et analystes d'interagir directement avec l'entrepôt de données pour répondre aux besoins de leurs métiers.
   - **Reporting** : Destiné essentiellement à la production et à la présentation périodique de rapports et de tableaux de bords sur les activités et les résultats d'une organisation. Ces outils permettent de donner une précieuse vue d'ensemble.
   - **Analyse dimensionnelle** : Permet aux utilisateurs d'analyser les données selon différents critères afin de suivre les performances d'une entreprise.
   - **Fouille de données** : Appelé aussi forage de données, ces techniques permettent d'extraire de nouvelles connaissances en appliquant divers algorithmes. À partir de ces connaissances, de nouvelles corrélations entre les données et des prédictions sur les résultats futurs peuvent se faire.

1.3 SIO VS SID

Le tableau 1.1 résume les principales différences entre les deux types de système.

Critère	OLTP (transactions)	OLAP (analyse)
Objectif	Production	Décisionnelle
Utilisateurs	Agents opérationnels (nombreux)	Analystes/décideurs (peu nombreux)
Données	Courantes, détaillées, évolutives	Historisées, agrégées, statiques
Taille de données	Peu volumineuse	Très volumineuse
Fréquence de mise à jour	En temps réel	En batch
Requêtes	Simples et répétitives	Complexes et imprévisibles
Fréquence des requêtes	Très fréquente	Peu fréquente
Temps d'exécution des requêtes	Court	Long
Mode d'accès	Lecture/écriture	Lecture
Degré de normalisation	Normalisé	Dénormalisé
Mode d'alimentation	Utilisateurs de la base	ETL uniquement

TABLE 1.1: Comparaison entre OLTP et OLAP

1.4 Organisation du manuscrit

Ce polycopié est consacré à la présentation des aspects théoriques et aux différentes briques nécessaires pour la mise en place d'un SID.

Dans le chapitre 2, nous présenterons le cœur du SID, à savoir, les entrepôts de données. Nous y détaillerons ses caractéristiques. Le chapitre 3 présentera les différentes architectures d'un entrepôt de données. Nous nous focaliserons dans le chapitre 4 sur les concepts clés d'un entrepôt de données et son modèle conceptuel. Un guide de bonne pratique pour la conception y sera présenté. Dans le chapitre 5, nous présenterons les différentes approches d'intégration. Nous nous attarderons sur l'approche ETL qui est le coeur du processus d'intégration dans un entrepôt de données. Le chapitre 6 présentera les différentes stratégies de stockage physique et les différentes opérations réalisables sur l'entrepôt de données à des fins d'analyse

et d'exploitation. Nous présenterons les deux langages de requêtes les plus utilisés : Le SQL étendu pour l'OLAP et le langage MultiDimensional eXpressions (MDX).

Chapitre 2

Introduction aux entrepôts de données

2.1 Définition

Les entrepôts de données centralisent les informations en provenance de différentes sources hétérogènes et ont comme objectif de fournir une vue globale aux décideurs. Il existe plusieurs définitions de ce qu'est un entrepôt de données (en anglais, DataWarehouse), mais la communauté s'entend pour citer celle d'un des pionniers du domaine décisionnel : Bill Inmon.

Définition de Bill Inmon : L'entrepôt de données est une collection de données orientées sujet, intégrées, non volatiles et historisées, organisées pour le support d'un processus d'aide à la décision.

— **Orientées Sujet :** Les données des SIO sont organisées par processus fonctionnel. Les SID quant à eux regroupent et organisent les données autour des sujets majeurs de l'organisation. Ces sujets sont souvent transverses par rapport aux structures fonctionnelles du SIO. La figure 2.1 illustre la mise en place d'une table Enseignants dans le SID. Cette table sera alimentée à partir de données se trouvant dans différents SIO.

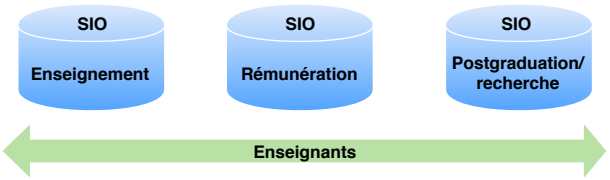


Figure 2.1: Données orientées sujet

— **Données intégrées :** Les données proviennent de sources hétérogènes et doivent être intégrées dans un environnement homogène. Cette partie est la plus complexe dans la mise en place d'un entrepôt. En effet, elle demande aux concepteurs de mettre en place un référentiel unique et cohérent et des règles d'intégration. La figure 2.2 montre un exemple d'intégration de données.

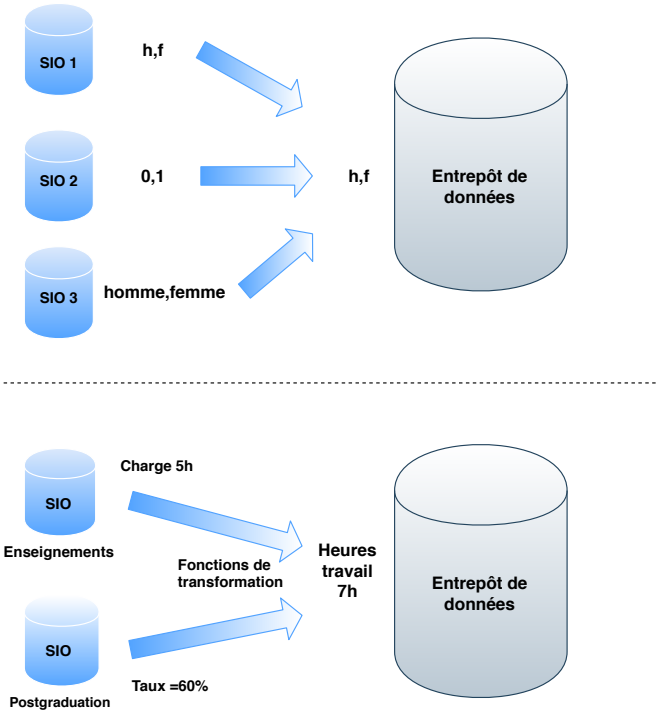


FIGURE 2.2: Données intégrées

— **Données non volatiles :** Dans un entrepôt de données, les données ne doivent être ni modifiées ni supprimées. Il est très important de garder trace de toutes les informations utiles à la prise de décision. La figure 2.3 illustre la spécificité d'un entrepôt de données où seules les opérations de sélection y sont autorisées. Contrairement aux SIO, où il est possible de supprimer et de mettre à jour les données.

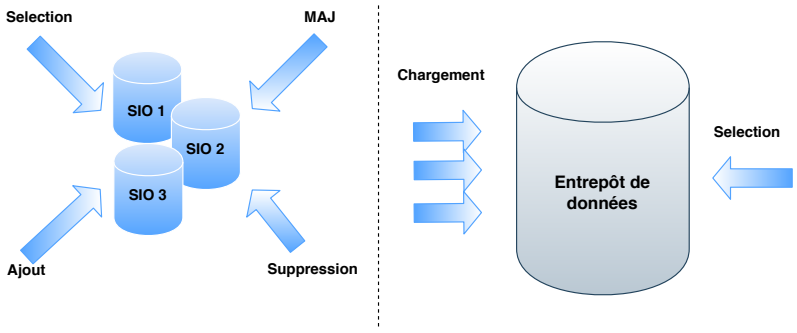


FIGURE 2.3: Données non volatiles

— **Données historisées :** Pour garantir la cohérence dans la prise de décision, un référentiel de temps est mis en place afin d'identifier la donnée dans le temps. En effet, une même requête sur un même ensemble de données doit retourner exactement le même résultat, peu importe le moment de son exécution. L'exemple dans la table 2.1 montre une manière de garder trace de l'évolution de grade d'un enseignant. Pour cela, une date de mise à jour est insérée à chaque changement, tout en gardant l'historique.

sk_id	id	nom	grade
A	1	Benali	MCA

sk_id	id	nom	grade	date_maj
A	1	Benali	MCA	25/09/2017
B	1	Benali	Prof	30/06/2018

TABLE 2.1: Table enseignant avant et après mise à jour

Nous nous attarderons dans la section suivante sur les différentes solutions proposées pour garantir une bonne historisation de données.

2.2 Les données dans l'entrepôt

Plusieurs classes de données existent dans un entrepôt de données. Il est possible de les catégoriser comme suit selon les axes historique et synthétique :

- **Les données détaillées courantes :** Ces données reflètent les faits les plus récents. Elles sont généralement une copie des données des SIO. Elles permettent différents axes d'analyse. Elles sont volumineuses et nécessitent un matériel de stockage adéquat.
- **Les données détaillées historisées :** Elles représentent les faits historiques. Elles sont moins souvent interrogées que les données courantes mais offrent la possibilité de déterminer des analyses sur le passé d'une organisation. C'est l'ensemble de données le plus volumineux.
- **Les données agrégées :** Elles correspondent à des éléments d'analyse répondant aux besoins des décideurs. Ces données sont déjà pré-traitées et représentent un premier résultat d'analyse. Il existe deux catégories : les faiblement et les fortement agrégées. Ces données représentent un avantage dans la facilité d'analyse et la rapidité d'accès. Leur inconvénient réside principalement dans l'impossibilité de retrouver le détail des données si ces dernières n'ont pas été conservées. Pour déterminer les données agrégées il faudrait étudier quels sont les ensembles d'attributs les plus pertinents à résumer ainsi que la fréquence de leur mise à jour.
- **Les méta-données :** Elles représentent les données sur les données. Elles permettent de répondre aux questions qui se posent en ce qui concerne les données :
  - Les informations sur les données stockées : leur format, signification, leur origine et comment sont-elles calculées (Le processus d'extraction à partir des bases sources)....
  - La date du dernier chargement de l'entrepôt de données.
  - Etc.

2.3 Magasins de données

Le magasin de données, connu aussi sous le nom de *DataMart* est considéré comme une vue partielle orientée métier de l'entrepôt de données. Deux définitions majeures sont à retenir :

Définition de Bill Inmon Le DataMart est issu d'un flux de données provenant du DataWarehouse. Contrairement à ce dernier qui présente le détail des données pour toute l'entreprise, le DataMart a pour vocation de présenter la donnée de manière spécialisée, agrégée et regroupée fonctionnellement.

Définition de Ralph Kimball Le DataMart est un sous-ensemble du DataWarehouse, constitué de tables au niveau détail et à des niveaux plus agrégés, permettant de restituer tout le spectre d'une activité métier. L'ensemble des DataMarts de l'entreprise constitue le DataWarehouse.

2.3.1 Caractéristiques

— Contient une partie du contenu de l'entrepôt de données.

— Se focalise sur un seul métier ou fonction.

— Sert à faire des analyses spécifiques et plus rapides.

— Même processus de conception qu'un entrepôt de données mais gère moins de ressources.

— Simple à concevoir.

— Meilleure performance grâce à des requêtes moins complexes.

2.4 Dualité Entrepôt et Magasin

Le tableau 2.2 résume les principales différences entre un entrepôt et un magasin de données.

Critère	Entrepôt de données (ED)	Magasin de données
Objectif	Plusieurs domaines d'analyse	Un domaine d'analyse
Temps de développement	Années	Mois
Complexité de développement	grande	Faible à moyenne
Taille de données	Très volumineuse	Moyennement volumineuse
Type de données	Courantes et historiques	Courantes et historiques
Transformation de données	Grande	Moyenne
Fréquence de mise à jour	Moins fréquente (mensuelle)	Fréquente (par heure, jour ou semaine)

TABLE 2.2: Comparaison entre entrepôt et Magasin de données

Chapitre 3

Architecture d'un entrepôt de données

3.1 Introduction

La mise en place d'un entrepôt de données est un défi de taille. Il existe plusieurs façons de penser un entrepôt. Le choix de la bonne architecture se base sur la combinaison de plusieurs critères : les besoins exprimés en termes d'objectifs, des utilisateurs cibles, du temps alloué pour la mise en place...etc. Nous retrouvons dans la littérature 5 différentes architectures. Deux d'entre elles sont les plus utilisées dans la pratique. Elles correspondent aux visions des deux fondateurs du domaine décisionnel : Bill Inmon et Ralph Kimball.

Nous présentons dans ce chapitre les différentes architectures, leurs caractéristiques, avantages et limites.

3.2 Magasins de données indépendants

Comme définit précédemment, un magasin de données peut être vu comme une sous-partie d'un entrepôt de données répondant à un besoin fonctionnel précis. Dans une logique de regrouper tous les besoins d'une organisation, ce type d'architecture propose de regrouper les différents magasins de données. Ces magasins de données sont construits de manière indépendante les uns des autres, pouvant utiliser des sources de données indépendantes. La figure 3.1 illustre l'indépendance de mise en place de chaque magasin et la possibilité aux différents utilisateurs d'utiliser les différents magasins ensemble.

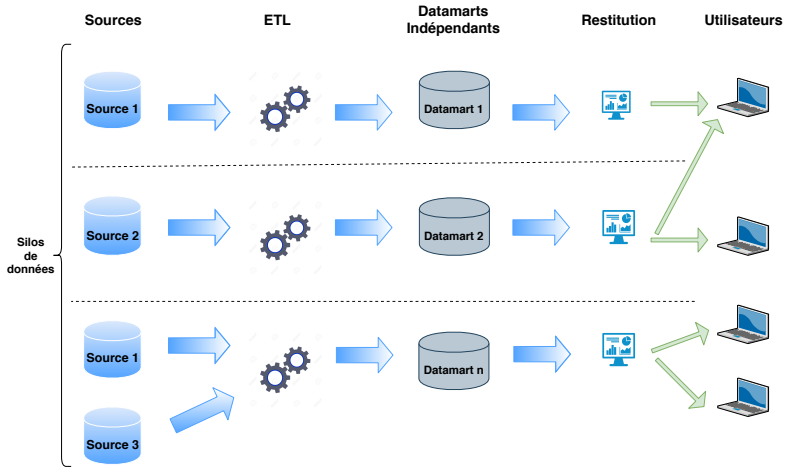


Figure 3.1: Magasins de données indépendants

Avantages et inconvénients

+ Architecture simple et peu coûteuse dans sa mise en place.
+ Fortement orientée sujet.
+ Personnalisation des analyses pour les différents utilisateurs.
– Certaines données peuvent se retrouver dans différents magasins, ce qui implique une répétition de traitements.
– Incohérence et redondances entre les magasins de données.
– Analyse inter-fonctionnelle difficile, voire impossible.

3.3 Architecture en bus de magasins de données

Cette architecture est proposée par R. Kimball. Connue aussi sous le nom d'approche **Bottom-Up**, elle propose de construire des magasins de données mais en utilisant des dimensions conformes. Autrement dit, il s'agit toujours d'une conception fortement orientée sujet mais pour certaines données, les magasins utiliseront des dimensions communes. Par exemple, les dimensions temporelle et géographique. Cela permet de pallier les limites de l'architecture précédente en éliminant certaines redondances et incohérences au niveau des données.

La figure 3.2 illustre l'architecture. L'ensemble des magasins de données représentent un entrepôt de données.

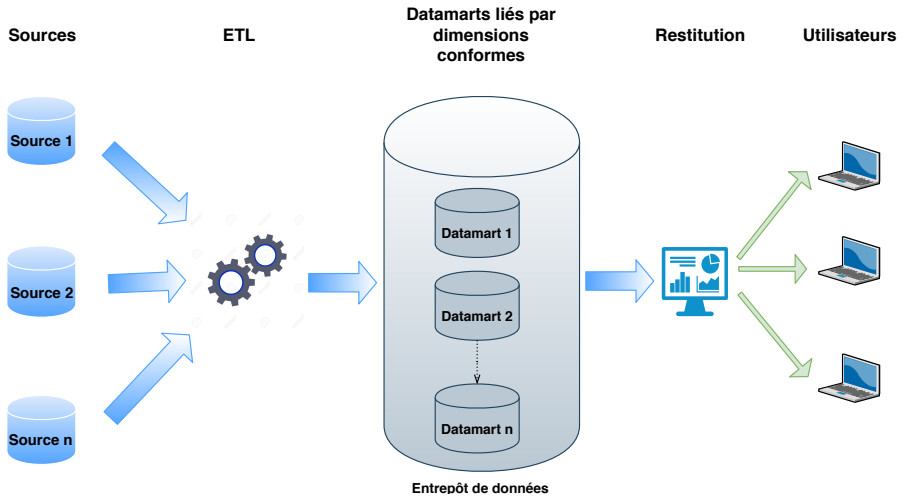


FIGURE 3.2: Architecture en bus de magasins de données

Avantages et inconvénients

+ Architecture incrémentale : Permet de rajouter des magasins au besoin et de traiter les processus les plus importants en premier.
+ Fortement orientée sujet.
+ Une intégration de données cohérentes grâce aux dimensions conformes.
– Analyse inter-fonctionnelle peu performante impliquant plusieurs magasins.
– Planification de nouveaux magasins complexe car il faudra les intégrer à l'existant.

3.4 Architecture Hub-and-Spoke

Proposée par B. Inmon, cette architecture est l'opposée de celle en bus d'un point de vue approche. Appelée aussi approche **Top-Down**, elle propose de centraliser toutes les données en construisant en premier lieu tout l'entrepôt de données (*hub*) et d'alimenter à partir de ce dernier les différents magasins (*spokes*).

La figure 3.3 présente l'architecture. L'entrepôt contiendra les données atomiques, quant aux magasins, ils contiendront des données agrégées. Les analyses se font directement sur les magasins.

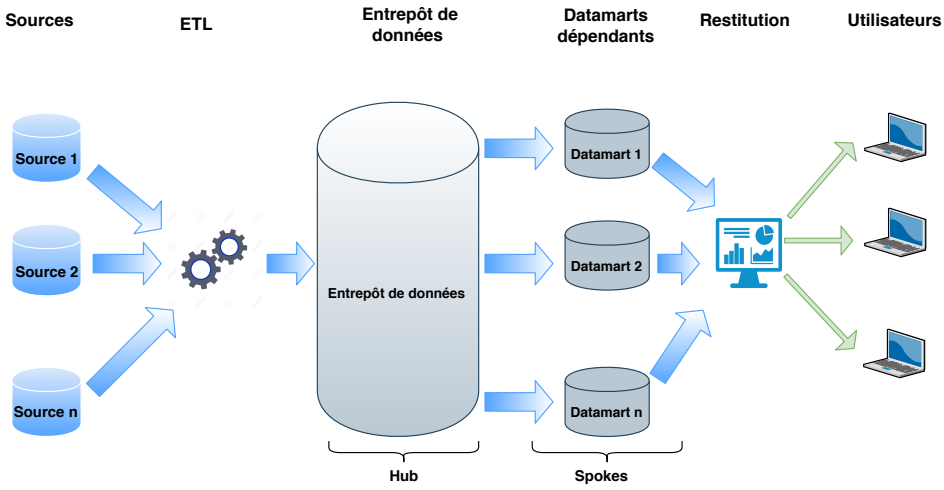


FIGURE 3.3: Architecture Hub and Spoke

Avantages et inconvénients

+ Intégration et consolidation complète de toutes les données dans un seul entrepôt.
+ Approche extensible : Il est plus facile de définir de nouveaux magasins.
– Analyse inter-fonctionnelle peu performante impliquant plusieurs magasins.
– Temps de mise en place : Construction de tout l'entrepôt avant de créer un magasin de données.

3.5 Architecture centralisée

Similaire à l'architecture Hub and Spoke, son but est de centraliser toutes les données dans un seul entrepôt. La différence réside dans l'inexistence des magasins de données dans l'architecture centralisée. La figure 3.4 illustre le principe. Les requêtes analytiques se font directement sur l'entrepôt. Ce dernier contient les données aussi bien détaillées que résumées.

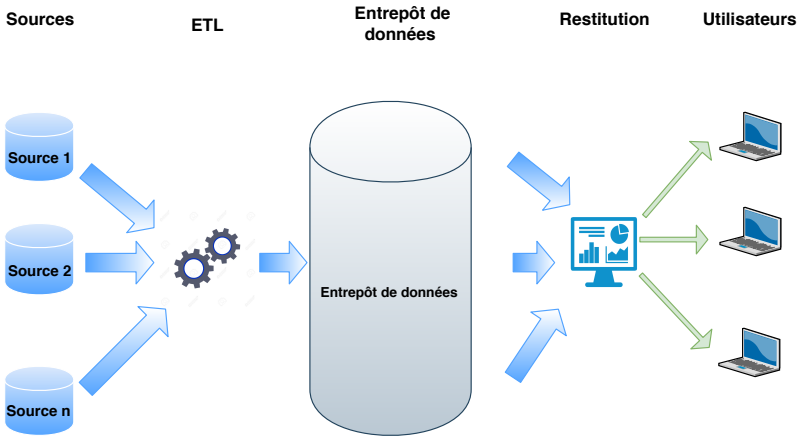


FIGURE 3.4: Architecture centralisée

Avantages et inconvénients

+ Les utilisateurs peuvent requêter toutes les données de l'entrepôt.
+ Performance optimale.
– Approche non incrémentale.
– Extensibilité limitée et très coûteuse : Il faudrait repenser toute la conception de l'entrepôt.

3.6 Architecture fédérée

Cette architecture est utilisée dans le cas où un ou plusieurs entrepôts sont déjà mis en place, comme dans le cas de fusions de compagnies. Au lieu de reconcevoir un nouvel entrepôt, l'architecture propose de mettre en place un entrepôt de données virtuel. Ce dernier représente une vue globale sur les différents entrepôts existants et un point d'entrée unique pour les utilisateurs. La figure 3.5 présente l'architecture. L'intégration de données dans l'entrepôt peut être logique ou physique à l'aide de métadonnées.

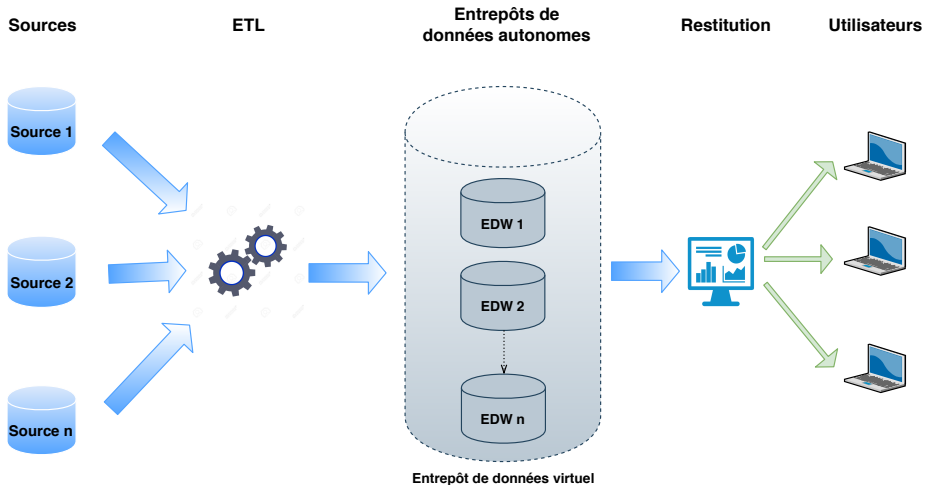


FIGURE 3.5: Architecture fédérée

Avantages et inconvénients

+ Pratique s'il existe au-préalable des entrepôts déjà mis en place.
+ L'intégration virtuelle ne demande que peu de ressources matérielles additionnelles.
− La gestion de l'intégration est complexe : Il faut prendre en compte la synchronisation, le parallélisme...
− Performance analytique faible.

3.7 Facteurs à considérer pour le choix de l'architecture

Comme présenté auparavant, il faut prendre en considération plusieurs critères pour choisir l'architecture la plus adaptée pour la construction d'un entrepôt de données. Nous citons dans ce qui suit une liste non exhaustive des paramètres qui influencent ce choix :

— L'interdépendance informationnelle entre les métiers d'une organisation (Bonne intégration VS silos de données).
— L'urgence d'obtenir une solution fonctionnelle.
— Les contraintes sur les ressources (financières, mains d'œuvre...).
— Le nombre de sources de données.
— La quantité de données (Gigaoctets, Téraoctets, Zettaoctets...).
— La fréquence de mise à jour de données (Mise à jour hebdomadaire, temps réel...).
— La nature des tâches des utilisateurs finaux (rapports simples, fouille de données).
— Le nombre d'utilisateurs.
— Les objectifs du projet (stratégique, opérationnel...).
— ...

Chapitre 4

Conception d'un entrepôt de données

4.1 Introduction

La mauvaise performance des systèmes d'information opérationnels face aux requêtes d'analyse complexes est due à la modélisation relationnelle (schéma entité association). Cette limite a poussé les concepteurs à proposer une nouvelle modélisation pour l'entrepôt de données. En effet, comme vu plus haut, dans un système d'information opérationnel, les données sont représentées dans un modèle entité/association en 3-ème forme normale. Ce dernier privilégie la création de multitude de tables pour éviter toute redondance et donc risque d'incohérence. Il faudra réaliser beaucoup de jointures pour naviguer dans ces tables. Ce qui peut affecter les performances de requêtes complexes. Nous présentons dans ce chapitre la modélisation multidimensionnelle proposée pour optimiser les performances des requêtes analytiques.
Introduit par R. Kimball, un modèle dimensionnel vise à présenter les données sous une forme intuitive qui permet des accès très performants. Il permet de considérer un sujet analysé dans un espace à plusieurs axes d'analyse. Les données sont organisées de manière à mettre en évidence les sujets traités (tables de *faits*) et le contexte de la réalisation des faits (tables de *dimensions*). La figure 4.1 présente une vue globale sur ce qu'est un modèle multidimensionnel.

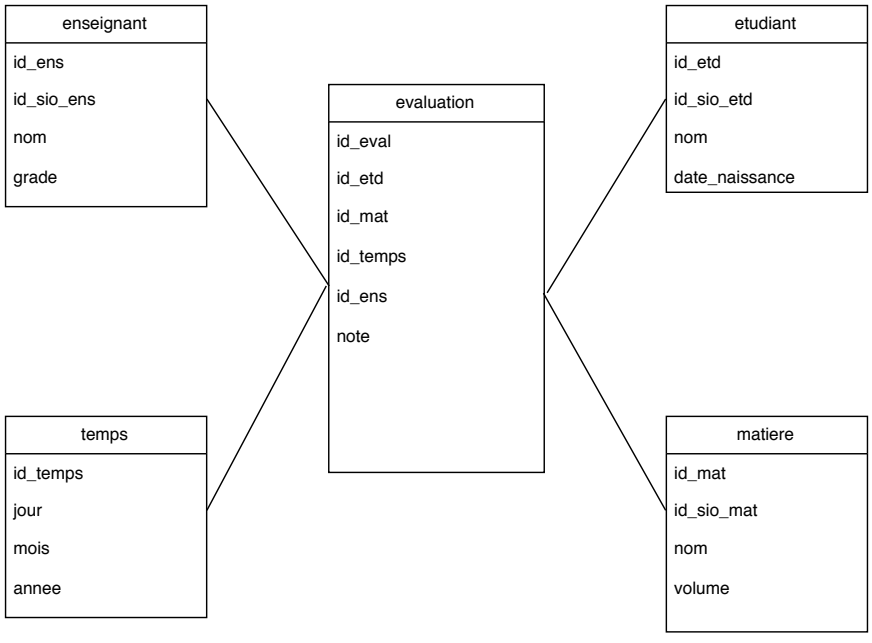


Figure 4.1: Modèle multidimensionnel

4.2 Fait

Un fait représente le sujet d'analyse, il permet de mesurer l'activité au sein d'une organisation. Par exemple, la note d'un étudiant, le nombre d'heures enseigné sont des faits qui peuvent être considérés pour mesurer l'activité au sein d'une université. Cette valeur peut être mesurée ou calculée selon les différents axes d'analyse. Par exemple, la moyenne est une mesure calculée à partir des notes d'un étudiant, pour un semestre donné, pour une année donnée.

La valeur de la mesure est en général une valeur numérique. Tous les faits sont stockés dans une table de faits. Cette dernière est composée d'un ensemble de clés étrangères (pointant vers les tables de dimensions) et des mesures du fait.

Une table de faits est caractérisée par un petit nombre de colonnes et un très grand nombre de lignes. Par exemple, la table de faits recensant les notes des étudiants contiendra toutes les notes de toutes les matières de tous les étudiants inscrits à l'université depuis sa création. La figure 4.2 résume ce qu'est une table de faits.

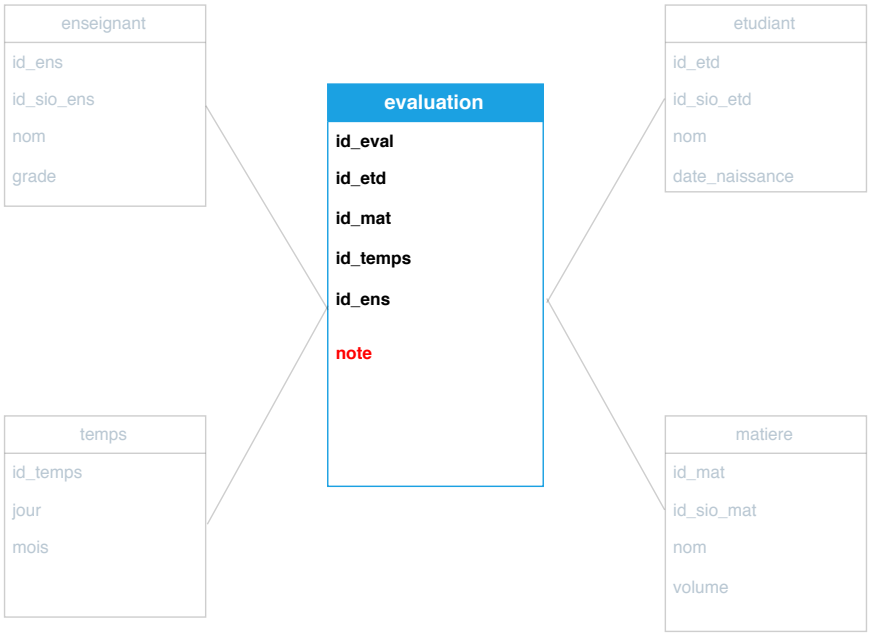


FIGURE 4.2: Table de faits

4.2.1 Additivité des mesures

Il existe trois types de mesures :

— Mesure additive : Elle est additionnable suivant toutes les dimensions. Par exemple, le nombre d'heures enseignées, la quantité de produits vendus,...etc.

— Mesure semi-additive : Elle est additionnable seulement suivant certaines dimensions. Par exemple, le stock des produits n'est pas additif par rapport à la dimension temps ni le solde d'un compte.

— Mesure non additive : Elle ne peut pas être additionnable quelle que soit la dimension. Par exemple, le prix unitaire d'un produit, la température...etc.

4.2.2 Tables de faits sans mesure

Dans certains cas, il peut exister des tables de faits sans mesure. Elle est composée uniquement de clés étrangères pointant vers les dimensions. Ce type de table est utilisée dans le suivi d'évènement et dans la couverture.

— Suivi d'événements pour l'analyse de la fréquentation journalière dans une université. Il n'y a aucune activité à mesurer.

— Couverture d'événements qui non pas eut lieu. Par exemple, les articles non vendus.

4.2.3 Granularité des faits

Le choix de la granularité des faits est très important lors de la conception d'un entrepôt de données. La granularité représente le niveau de détails pour représenter les faits. Par exemple, est ce qu'il est préférable de représenter le montant des ventes d'une journée entière ou bien le montant des ventes réalisées par heure du jour. Le choix dépend des besoins d'analyse de l'organisation. Il faut savoir que plus la granularité est fine, plus les analyses sont précises mais la taille de l'entrepôt se voit directement impactée.

4.3 Dimension

Une table de dimension contient les informations relatives au contexte de la réalisation des faits. Elle contient des critères (appelés aussi paramètres) suivant lesquels on souhaite évaluer et qualifier un fait. Elle représente un axe selon lequel les données sont analysées. Contrairement aux tables de faits, une table de dimension contient peu de lignes mais plusieurs colonnes. Ces colonnes sont généralement des attributs sous forme de descriptions textuelles permettant de qualifier ou d'expliquer le contexte, comme par exemple l'attribut nom d'une matière, le nom d'un étudiant...etc. Elles peuvent dans certains cas être du type discret (ensemble limité de valeurs) : le niveau d'étude (L1, L2...). Les tables dans la figure 4.3 sont des exemples de dimensions.

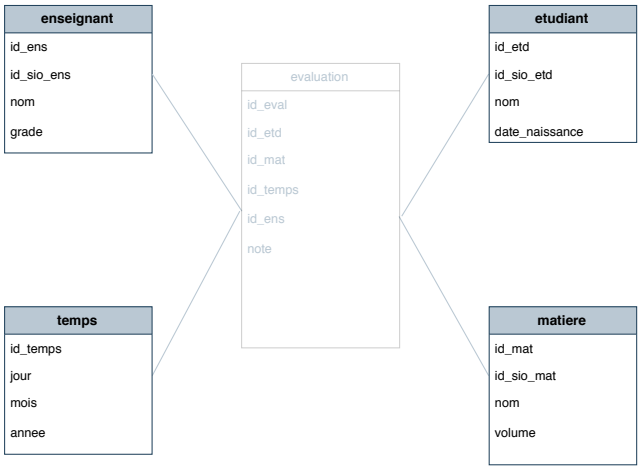


FIGURE 4.3: Tables de dimensions

Important

— La clé primaire d'une dimension doit toujours être une clé artificielle composée d'une seule colonne (exemple : auto-incrémental). Cela offre une performance d'accès en utilisant les index, la robustesse du système car les valeurs des clés ne changent pas contrairement à une clé naturelle.

— La dimension temps doit obligatoirement être présente dans tout entrepôt de données et être reliée à toute table de fait. Les données manipulées dans un entrepôt de données font souvent référence à la notion du temps : Date de l'événement, date de chargement, date de mise à jour...etc.

4.3.1 Hiérarchie dimensionnelle

Les analyses multidimensionnelles utilisent des hiérarchies d'attributs pour visualiser les données à différents niveaux de détail. Ces hiérarchies organisent les attributs entre eux selon une dépendance de hiérarchie. Par exemple, dans la dimension temps, il est possible de trouver différentes hiérarchisations :

— Année → Mois → Semaine → Jour

— Année → Trimestre → Mois

— ...

Le choix de la hiérarchie dépend des besoins d'analyse.

4.4 Mesures VS attributs d'une dimension

Pour certains attributs, il est parfois difficile de savoir s'ils représentent une mesure caractérisant un fait ou un attribut d'une dimension caractérisant ainsi le contexte. De manière générale :

— La valeur d'une mesure dépend d'un évènement alors que celle d'un attribut d'une dimension en est indépendante. Par exemple, le montant total d'une vente dépend d'une vente alors que le prix unitaire d'un produit est indépendant.

— Les mesures permettent de calculer des indicateurs de performance alors que les attributs permettent de filtrer et étiqueter les faits.

4.5 Types de dimensions

Lors de la conception d'un entrepôt de données, il est possible de rencontrer des cas particuliers. Pour exprimer ces différentes spécificités, plusieurs types de dimensions ont vu le jour. Nous présentons dans cette section les dimensions les plus utilisées. Nous détaillerons dans la section suivante tous les types de dimensions qui servent à répondre aux problèmes d'historisation de données.

Dimension indésirable (Junk)

Dans la conception d'entrepôts de données, nous rencontrons souvent une situation dans laquelle il existe des attributs indicateurs dans le système source qu'il

est nécessaire de conserver dans l'entrepôt de données pour des besoins analytiques. Ces champs ne correspondent généralement à aucune dimension particulière et ont la spécificité d'avoir une petite cardinalité (valeurs distinctes possibles).

Pour inclure ces informations dans l'entrepôt de données, il est nécessaire alors de construire de nombreuses dimensions de petite taille (autant de tables que d'attributs) et d'inclure la clé de chaque table dans la table de faits. Cette approche implique que la quantité d'attributs stockés dans la table de faits augmente de manière considérable et peut entraîner des problèmes de performance et de gestion.

Pour remédier à ce problème, il est possible d'utiliser une Junk dimension. Dans cette dernière, les champs indicateurs sont combinés en une seule dimension. De cette façon, une seule table de dimension est créée, et le nombre de champs dans la table de faits, ainsi que la taille de la table de faits, peut être réduit. .

Une Junk dimension crée des combinaisons de toutes les valeurs distinctes de ces attributs et les stocke dans une seule table de dimension. Les clés de substitution pour ces lignes sont insérées dans la table de faits. La table de faits, au lieu d'avoir plusieurs clés étrangères pour chacun de ces attributs, a une seule clé étrangère représentant une ligne de la table Junk dimension.

Dimension causale

D'un point de vue technique, une dimension causale est une dimension comme une autre (produit, employé, magasin, temps, etc.). Tout comme une dimension temporelle stocke des informations sur la date et l'heure du déroulement du fait, la dimension causale stocke des informations sur les causes du fait. Par exemple, la dimension promotion peut provoquer des ventes et cela permet de mesurer le taux de réussite du service marketing.

Dimension douteuse

C'est une dimension qui peut contenir de nombreux doublons ou des informations douteuses. Par exemple, une dimension Client dans laquelle la même personne peut apparaître de nombreuses fois (orthographes légèrement différentes, attributs différents, etc.). Elle est utilisée quand il y a un doute sur la qualité de données provenant des SIO.

Dimension dégénérée

Une dimension dégénérée est une dimension spéciale telle que le numéro de facture, l'identifiant d'une transaction...etc. Cependant, aucun attribut ne lui est associé car tous les attributs importants font déjà partie de leurs dimensions respectives. Ainsi, une facture peut avoir comme attribut le nom du client, mais cela fait déjà partie de la dimension client.

Une dimension dégénérée sera représentée par un attribut dans la table de faits. Cet attribut n'est pas considéré comme une mesure mais représente à lui seul la dimension.

Il faut toujours garder ces attributs dans la table de faits car ils permettent de :

— Répondre à certains types de questions, comme par exemple, quel est le nombre moyen d'articles correspondant à une même commande ?

— Retracer la provenance d'une ligne à une source de données.

Dimension conforme

Une dimension conforme permet une analyse cohérente de différentes mesures utilisées dans plusieurs modèles dimensionnels. Une dimension de produit peut être associée à différentes tables de faits, par exemple, table d'inventaire, de vente, d'approvisionnement et autres. Il est important de collaborer étroitement pour avoir une définition cohérente de cette dimension. La dimension temporelle est généralement une dimension conforme.

Dimension statique

Une dimension statique est une dimension qui n'est pas chargée à partir du système source. Les dimensions statiques sont générées à l'aide d'un script SQL ou d'une procédure stockée et sont chargées manuellement. La dimension temporelle est un exemple classique de dimension statique.

Dimension de jeu de rôles

Une dimension de jeu de rôles est une dimension qui peut avoir plusieurs alias pour réaliser une analyse dans un contexte différent. La dimension temporelle est une dimension de jeu de rôles. Supposons que la table de faits comporte une clé de date de commande, de date d'expédition et de date de livraison. Nous pouvons créer trois alias de la dimension Temps (trois rôles) et les nommer comme dimension date de commande, date d'expédition et dimension du délai de livraison. La clé est de garder à l'esprit qu'il n y'a pas création de plusieurs copies physiques de la table de dimension temporelle, mais uniquement des alias dans la couche logique.

4.6 Modèles de conception

Afin d'exploiter au maximum et de manière la plus performante possible ces tables. Trois types de modèles sont fréquemment utilisés : le modèle en étoile, en flocons ou en constellation.

4.6.1 Modèle en étoile

Dans ce modèle de conception, le schéma contient une table centrale (table des faits). Elle est composée de clé primaires et étrangères. Ces dernières pointent vers un ensemble de dimensions qui sont placées tout autour de la table de faits. La figure 4.4 présente un petit exemple.

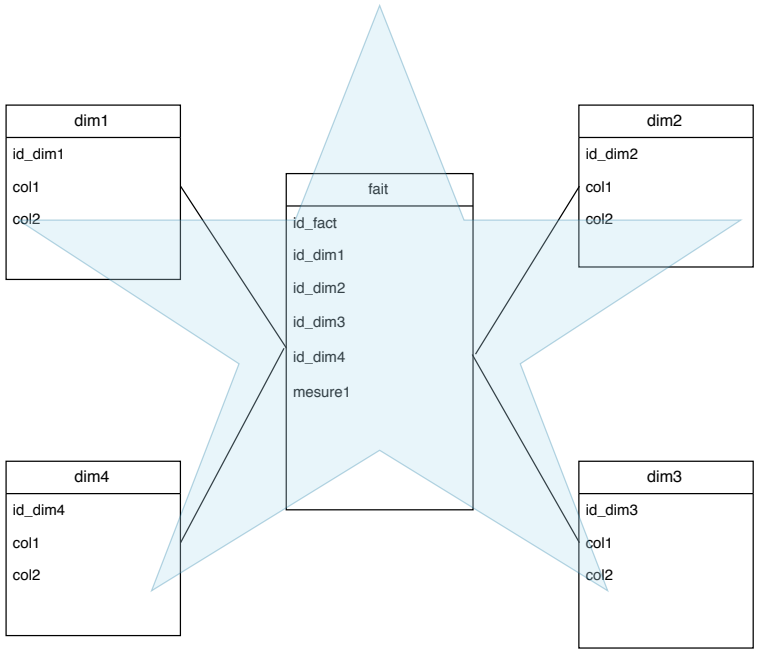


FIGURE 4.4: Modèle en étoile

Avantages et inconvénients

+ Facilité de navigation.
+ Très bonnes performances : nombre de jointures limité.
+ Gestion des agrégats.

− Redondances dans les dimensions.

4.6.2 Modèle en flocon

Le principe est le même que dans le modèle en étoile, il y a une table de faits et plusieurs dimensions. Mais en plus, les dimensions peuvent être décomposées en sous-dimensions. Le but est de représenter une hiérarchie au sein des dimensions. Ce modèle est plus complexe par rapport au modèle en étoile et est utilisé lorsque les tables sont très volumineuses. L'exemple illustré dans la figure 6.16 représente un modèle en flocon regroupant les hiérarchies de la dimension temps.

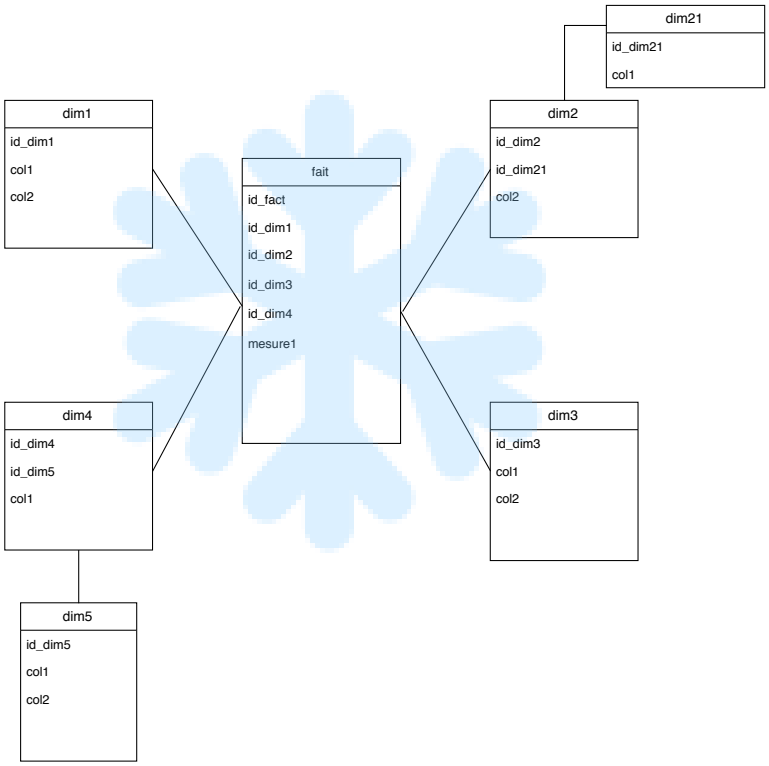


FIGURE 4.5: Modèle en flocon

Avantages et inconvénients

+ Réduction du volume de données dans les tables
+ Permet des analyses par pallier sur la dimension hiérarchisée.

– Nombres de jointures qui augmentent avec les dimensions hiérarchisées.
– Navigation moins simple.

4.6.3 Modèle en constellation

Le modèle en constellation se caractérise par la présence de plusieurs tables de faits. Autrement dit, il regroupe plusieurs modèles en étoile ou flocon via des dimensions conformes. La figure 4.6 illustre ce concept.

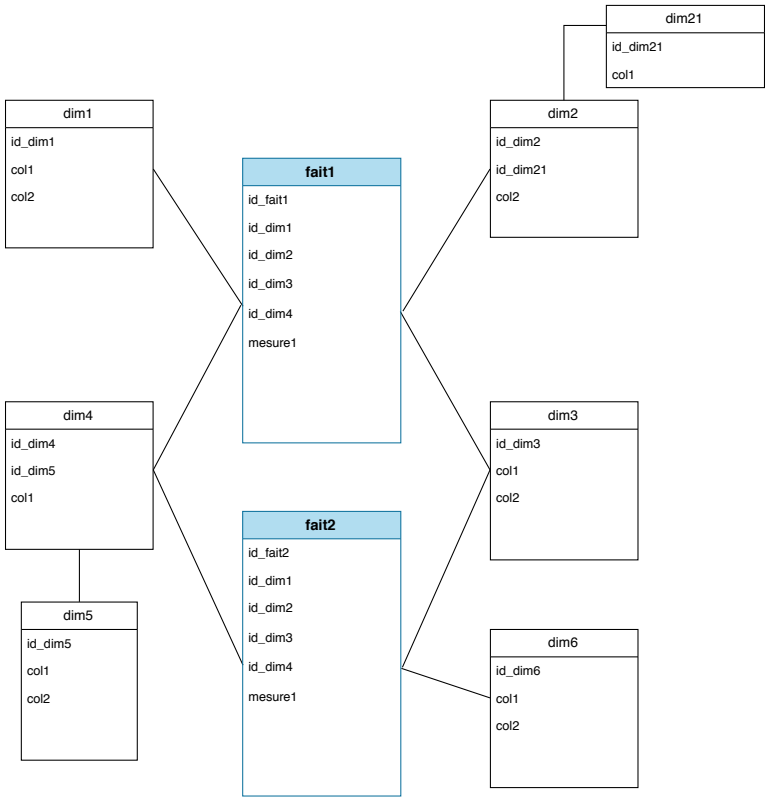


FIGURE 4.6: Modèle en constellation

4.7 Historisation de données

Un entrepôt de données tourne autour de deux principaux concepts : les faits et le contexte de leur déroulement. Dans les tables de faits, toutes les données mesurables (faits) y sont stockées, par exemple, la note d'un étudiant, la publication d'un enseignant...etc. Il est très rare qu'il y ait des changements dans les faits, une note par exemple ne change pas avec le temps.

Les dimensions quant à elles représentent le contexte de réalisation d'un fait. Elles représentent les axes d'analyse. Dans notre exemple, les dimensions étudiants, enseignants, matières...etc peuplent notre entrepôt. C'est ces dernières qui sont les plus sujettes aux changements. Le grade d'un enseignant peut évoluer dans le temps, le nom d'une matière, l'adresse d'un étudiant...etc. Pour certains de ces changements, il est primordial de pouvoir les capter et les conserver.

Pour répondre à ce défi, les dimensions ont été réparties en deux catégories selon leur évolution dans le temps : les dimensions à évolution lente (Slowly Changing Dimension (SCD)) et les dimensions à évolution rapide (Rapidly Changing Dimension (RCD)).

4.7.1 Dimensions à évolution lente (SCD)

Dans cette catégorie, les dimensions peuvent subir quelques changements dans le temps (assez peu fréquent). Pour prendre en charge ces derniers, Ralph Kimball propose trois types de gestion de changements. Prenons l'exemple de la table suivante

contenant les informations d'enseignants :

sk_id	id	nom	grade
A	1	Benali	MCA

Les enseignants changent de grade pendant leur carrière. Nous déroulerons dans ce qui suit les différentes solutions proposées pour capter et historiser ces changements.

Changements de type 0 : Valeurs originales

Le changement de type 0 s'applique sur les attributs qui ne doivent jamais changer de valeurs. Par exemple, la date de naissance, date de recrutement,...etc. Il est souvent appliqué aux attributs date.

Changements de type 1 : Écrasement de valeurs

Le changement de type 1 est appliqué sur les champs dits non pertinents pour la prise de décision. Pour cause, le changement de type 1 consiste à ne pas sauvegarder l'historique d'un champ. Autrement dit, ne garder que la dernière valeur. Dans notre exemple, le changement de nom d'un enseignant n'influe en rien la prise de décision. Par conséquent, un changement de nom impliquera une simple mise à jour de la table dimension concernée.

sk_id	id	nom	grade
A	1	Ben Ali	MCA

TABLE 4.1: Changement de type 1 : écrasement de valeurs

Changements de type 2 : Ajout de nouvelles lignes

Les changements sont gérés en ajoutant une nouvelle ligne pour chaque mise à jour. Cela permet une historisation illimitée pour chaque donnée. Il existe plusieurs méthodes pour marquer la version courante d'une donnée :

— Utiliser un champ "version" qui s'incrémente de manière séquentielle pour chaque nouvelle modification.

sk_id	id	nom	grade	version
A	1	Benali	MCA	1
B	1	Benali	Prof	2

TABLE 4.2: Changement de type 2 avec version

— Utiliser des champs "date_début" et "date_fin" pour préciser la période effective d'une donnée. Dans ce cas-là, la dernière version aura comme "date_fin" la valeur nulle ou une date très lointaine (exemple : 31/12/9999).

— Utiliser les champs "date_effective" et "Flag". La valeur booléenne True du champ "Flag" indique la version courante.

sk_id	id	nom	grade	date_debut	date_fin
A	1	Benali	MCA	25/09/2017	30/06/2018
B	1	Benali	Prof	30/06/2018	31/12/9999

TABLE 4.3: Changement de type 2 avec date début et fin

sk_id	id	nom	grade	date_effective	flag
A	1	Benali	MCA	25/09/2017	No
B	1	Benali	Prof	30/06/2018	Yes

TABLE 4.4: Changement de type 2 avec date effective et flag

Changements de type 3 : Ajout de nouvelles colonnes

Dans ce type de changement, de nouvelles colonnes sont ajoutées à la dimension pour garder l'historique. Cette méthode permet de sauvegarder un historique limité correspondant au nombre de colonnes ajoutées. Dans notre exemple, il est possible d'ajouter les colonnes "date_effective" et "grade_actuel" pour stocker le nouveau grade ainsi que la date de la promotion.

sk_id	id	nom	grade_original	grade_actuel	date_effective
A	1	Benali	MCA	Prof	30/06/2018

TABLE 4.5: Changement de type 3 : Ajout de nouvelles colonnes

La limite de ce modèle se trouve dans l'impossibilité de sauvegarder tout l'historique. Dans le cas où un enseignant se voit promu deux fois, il ne serait possible de garder que son dernier grade ainsi que son grade initial. Une autre alternative est d'au lieu de garder la valeur originale, on garderait l'avant-dernière valeur.

Changements de type 4 : Ajout d'une table historique

La méthode de type 4 propose d'utiliser une table additionnelle pour garder l'historique d'une dimension. La dimension elle-même ne contiendrait que les valeurs actuelles. Dans notre exemple, les changements de grades des enseignants se traduiraient comme suit :

sk_id	id	nom	grade
B	1	Benali	Prof

sk_id	id	nom	grade	date_effective
A	1	Benali	MCA	25/09/2017
B	1	Benali	Prof	30/06/2018

TABLE 4.6: Changement de type 4 : Table enseignant et sa table historique

Changements de type 6 : Combinaison de différentes méthodes

La méthode de type 6 est vue comme une combinaison des méthodes 1, 2 et 3 (1+2+3 =6). La dimension est initialement prévue pour stocker la valeur actuelle, la valeur historique, la période effective d'une valeur ainsi qu'un attribut "flag" pour

sk_id	id	nom	grade_actuel	grade_historique	date_debut	date_fin	Flag
A	1	Benali	MCA	MCA	25/09/2017	31/12/9999	Y

connaitre l'état courant. A l'initialisation, la valeur actuelle et la valeur historique sont les mêmes.

La mise à jour d'une donnée dans la dimension se traduit comme suit : une nouvelle ligne est ajoutée pour garder trace du changement (comme dans le type 2), la valeur de "flag" de l'ancienne ligne est remplacé (écrasement de valeur, comme dans le type 1). De plus, la dernière valeur est sauvegardé dans l'attribut "valeur_historique" (comme dans le type 3).

sk_id	id	nom	grade_actuel	grade_historique	date_debut	date_fin	Flag
A	1	Benali	MCA	MCA	25/09/2017	30/06/2018	N
B	1	Benali	Prof	MCA	30/06/2018	31/12/9999	Y

TABLE 4.7: Changement de type 6 : combinaison de méthodes

4.7.2 Dimensions à évolution rapide (RCD)

Dans cette catégorie, une dimension est dite à évolution rapide si elle subit des changements très fréquents que l'on souhaite sauvegarder. Les changements fréquents entraînent une croissance rapide de la dimension. Si par exemple, l'historisation se base sur les changements de type 2, cela aura un impact sur la maintenance et la performance au fur et à mesure que la dimension grandit.

Prenons un exemple pour illustrer l'impact de ces changements. Considérons la dimension patients où il y a 10 000 lignes. En moyenne, chaque patients change ses informations une dizaine de fois en une année. Si le changement de type 2 est appliqué, il y aura 10000 * 10 = 100 000 lignes. En imaginant que la table compte 1 million de lignes, il sera très difficile de gérer la situation avec le type 2. Nous utilisons dans ce cas une approche de dimension qui change rapidement.

Parmi les solutions proposées pour pallier à ce problème, nous détaillerons ici une des plus utilisées qui est : l'isolation des attributs qui évoluent rapidement dans une dimension à part. Le résultat est obtenu en deux étapes :

— **Étape 1 :** Identifier les attributs qui évoluent rapidement et les isoler dans une table séparée. La dimension contenant ces attributs est aussi appelée *Junk dimension*. Les valeurs de chaque attribut seront divisées en un ensemble d'intervalles. La Junk dimension contiendra le produit cartésien de toutes les valeurs de ces attributs. Chaque combinaison est identifiée par une clé.

  Prenons la dimension patient. Les attributs comme patient_id, nom, sexe, situation maritale ne changeront pas ou changent très rarement. Mais les attributs comme le poids et l'IMC (indice de masse corporelle) change tous les mois en fonction de la visite du patient à l'hôpital. Nous devons donc séparer les attributs de poids et IMC de la table des patients. Ci-dessous un exemple de la junk dimension résultante :

— **Étape 2 :** Relier la nouvelle Junk dimension et la dimension mère. La relation directe en insérant la clé de la Junk dimension comme clé étrangère dans

id	poids	imc
1	30-40	≤ 18.5
2	30-40	18.5 à 25
3	30-40	≥ 25
4	40-50	≤ 18.5
...	...	...

la dimension mère, causera l'incrémentation du volume de données de la dimension mère. En effet, à chaque changement dans la Junk dimension, il faudra ajouter une nouvelle ligne dans la dimension mère. Pour pallier à ce problème, la solution proposée est de mettre en place une table intermédiaire contenant la relation entre les deux. Cette table est appelée *Mini dimension* et contiendra les deux clés des deux dimensions ainsi que les informations relatives aux dates effectives et à la valeur courante. Ci-dessous la représentation de la relation entre les trois dimensions.

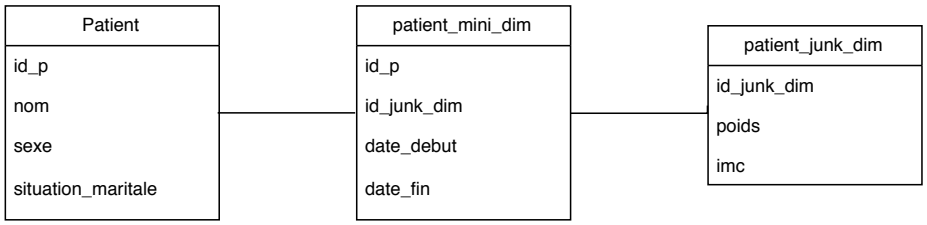


FIGURE 4.7: RCD

4.8 Les bonnes pratiques

Nous regroupons dans cette section quelques points importants assurant la bonne conception d'un entrepôt de données :

— Avant de se lancer dans la conception d'un entrepôt de données, il faut au préalable déterminer les points suivants :

1. Choisir le processus à modéliser.
2. Choisir le grain des faits. Décider ce que représente une ligne de la table de faits. Le niveau de détail : transaction individuelle, récapitulatifs journaliers, mensuels,...etc.
3. Identifier les dimensions qui s'appliquent aux lignes de la table de faits. Le choix des dimensions résulte du grain choisi.
4. Identifier les mesures de fait qui renseignent la table de faits.

— La dimension temps doit toujours être présente dans un modèle d'entrepôt de données.

— Dans une table de fait, la concaténation des clés étrangères (pointant vers les différentes dimensions) doit garantir l'unicité d'une ligne.

— Les mesures doivent être dans la mesure du possible des valeurs numériques et additives.

— Une table de fait ne doit jamais être mise à jour (sauf correction). Un rapport généré sur le même ensemble de données à un instant T ou T+x doit contenir exactement les mêmes informations.

— Une table de fait ne doit pas contenir de ligne artificielle. Exemple, pour un étudiant ayant raté l'examen de la première session (avec justificatif), aucune ligne avec la note 0 ne doit être ajoutée dans le SID.

— Il faut respecter la première forme normale (unicité de la clé et valeurs atomiques des champs).

— Il est préférable d'utiliser une clé artificielle de type auto-incrément plutôt qu'une clé logique (composée de 1 ou plusieurs champs) et ceci afin de faciliter la maintenance due à l'évolution des dimensions et de faciliter les jointures.

— Il ne faut jamais supprimer une ligne d'une table de dimension. Pour ne pas afficher les lignes obsolètes, il est préférable d'utiliser des champs flag pour spécifier l'état de la ligne.

— Il faut éviter les relations n vers n en ce qui concerne les dimensions. Un département appartient à une région, un produit à une catégorie... Si un produit appartient à plusieurs catégories, cela risque de complexifier les analyses. Il est possible de définir deux dimensions distinctes et de les relier grâce à une table d'association.

Chapitre 5

Intégration dans un entrepôt de données

5.1 Introduction

L'intégration de données est l'étape la plus fastidieuse et la plus longue dans la mise en place d'un système d'information décisionnel. Intégrer un ensemble de données hétérogènes et disparate tant par la forme que par le format, dans un environnement homogène, est une étape bien complexe. Lors de cette étape les données sont transformées et filtrées pour représenter une source d'information homogène, commune et fiable. la performance du SID est étroitement liée à la qualité d'intégration de données. Il est à noter que l'étape d'intégration de données ne se limite pas au domaine décisionnel. Elle est plus générale et peut être appliquée pour différents besoins : réunir et requêter plusieurs systèmes d'informations opérationnels, faire communiquer des applications qui ont été faites en silo (indépendamment les unes des autres), etc.

5.2 Approches d'intégration

Plusieurs approches ont été développées en fonction des besoins d'intégration. Nous présentons dans cette section les approches les plus utilisées.

5.2.1 Extract Transform and Load (ETL)

C'est l'approche la plus utilisée dans la mise en place d'un entrepôt de données. Dans cette approche, l'intégration se fait en trois étapes :

— L'extraction des données à partir des sources.

— La transformation des données qui consiste à nettoyer, agréger les données pour les intégrer dans un schéma prédéfini.

— Le chargement de données dans la cible (l'entrepôt de données).

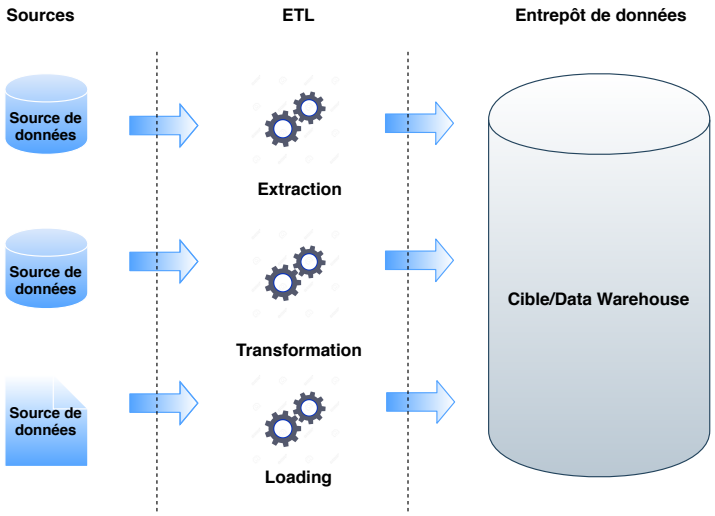


FIGURE 5.1: Extract, Transform and Load

Avantages et inconvénients

+ Peut traiter une quantité importante de données dans une même exécution.
+ Permet des transformations et des agrégations complexes sur les données.
+ Productivité améliorée grâce aux différents outils proposés (interfaces graphiques simplifiées...).
– Exige de l'espace mémoire pour effectuer les transformations.
– Latence des données entre la source et la cible.
– Unidirectionnel (des sources vers la cible).

Nous nous intéresserons dans ce polycopié plus en détails à cette approche qui est la plus utilisée pour l'intégration dans un entrepôt de données.

5.2.2 Entreprise Information Integration(EII)

Dans l'approche EII, aucune intégration physique n'est effectuée. Les sources de données hétérogènes sont consolidées à l'aide d'une base de données virtuelle, de manière transparente aux applications utilisant les données. La base de données virtuelle offre une vue unifiée des données. Les utilisateurs envoient directement leur requête sur la base de données. La requête est par la suite décomposée en sous-requêtes qui seront envoyées aux sources respectives. Les réponses sont assemblées en un résultat final.

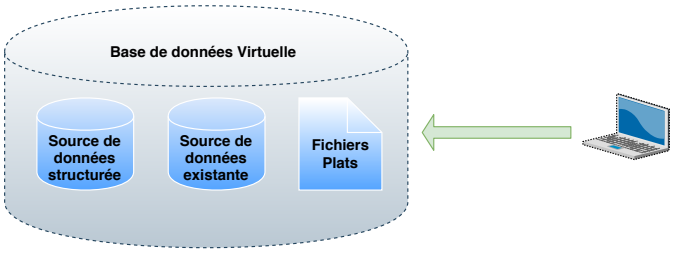


FIGURE 5.2: Entreprise Information Integration

Avantages et inconvénients

+ Fournit un accès en temps réel en lecture et en écriture (bidirectionnel).
+ Aucun déplacement de données.
+ Accélère le déploiement de la solution.
– Surcharge les systèmes opérationnels (sources).
– Transformations limitées sur les données.
– Consommation d'une grande bande passante du réseau.
– Réécritures des requêtes à chaque exécution.

5.2.3 Entreprise Application Integration(EAI)

Dans le but de faire communiquer des applications qui ont été construites dans des environnements différents et avec des technologies différentes, l'approche EAI repose sur l'intégration et le partage de données des applications à l'aide de services web (architecture SOA). Cette approche permet une communication en temps réel. Elle est utilisée également pour alimenter des entrepôts de données. Cette approche ne remplace pas un ETL.

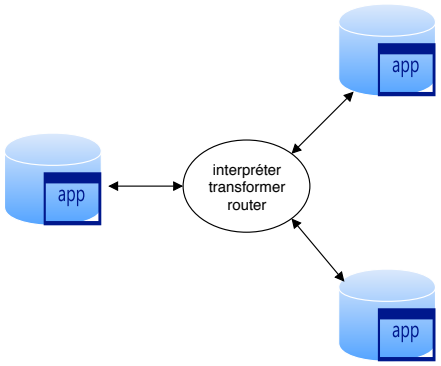


FIGURE 5.3: Entreprise Application Integration

Avantages et inconvénients

+ Facilite l'interopérabilité des applications.
+ Permet l'accès en temps-réel.
+ Permet de contrôler le flux d'information.
– Ne supporte quasiment pas de transformation et d'agrégations de données.
– Consommation de la bande passante du réseau.

Le tableau 5.1 résume les différentes approches et leurs caractéristiques.

5.3 ETL

Avant de se lancer dans l'étape d'intégration de données, il est important de définir la feuille de route et les éléments importants à prendre en considération. Nous présentons dans ce qui suit le déroulement dans l'ordre de la mise en place de cette intégration.

Critère	ETL	EII	EAI
Flux de données	Unidirectionnel	Bidirectionnel	Bidirectionnel
Latence	Journalier à mensuel	Temps réel	Temps réel
Transformation de données	Grande capacité	Moyenne capacité	Faible capacité
Contexte d'utilisation	Consolidation d'une grande quantité de données Transformations complexes	Relier un entrepôt existant avec des sources de données spécifiques Données sources volatiles et accessibles à l'aide de requêtes simples	Sources non accessibles directement Requêtes simples

TABLE 5.1: Comparaison entre les différentes approches d'intégration

1. Déterminer les données nécessaires qui serviront à alimenter l'entrepôt de données.
2. Déterminer les sources contenant ces données.
3. Préparer le staging area ou l'ODS (détaillé ci-dessous).
4. Définir les règles d'extraction des données.
5. Définir les règles de transformation et de nettoyage des données.
6. Planifier les agrégations des données.
7. Définir les procédures pour le chargement de données.
8. Chargement des tables de dimensions.
9. Chargement des tables de faits.

5.3.1 Operating Data Store et Staging Area

L'Operating Data Store (ODS) et le staging area sont des composants de l'architecture pour le stockage et la transformation de données en vue de faciliter l'intégration dans un entrepôt de données. Plusieurs définitions sont présentes dans la littérature sur le but de chacun. Ils existent, par ailleurs, plusieurs sources qui les définissent comme étant un même concept représenté sous différents noms. Il apparait que l'utilisation de l'un ou de l'autre, ou les deux en même temps dépend de la vision de l'organisation qui met en place le SID. Nous présentons par conséquent une définition (parmi tant d'autres) de ces deux concepts.

Operating Data Store L'ODS représente la zone de stockage des données sources, provenant des systèmes d'information opérationnels ou autres. Cette zone permet d'unifier les données dans un même format. Il ne sera plus question de fichiers délimités, fichiers XML ou autres. Toutes les données seront représentées sous forme de tables. L'ODS utilise le même SGBD que l'entrepôt de données. Les tables au niveau de l'ODS ne doivent subir aucune contrainte sur les données. Cette caractéristique permet de s'assurer qu'il n'y a aucun filtre pour récupérer les données. Par exemple, dans un SIO, les champs d'une table ont une taille prédéfinie (Le nom d'un client par exemple ne doit pas dépasser 30 caractères). Dans un ODS, cette contrainte ne doit pas être représentée. Les champs doivent avoir une taille maximale pour pouvoir récupérer toutes les données peu importe les changements appliqués au niveau du SIO.

Staging Area La zone Staging Area englobe tout le processus de transformation et de chargement effectués sur les données : (1) les transformations effectuées sur les données sources pour intégrer l'ODS et (2) les transformations effectuées sur les données de l'ODS pour être intégrées dans l'entrepôt.

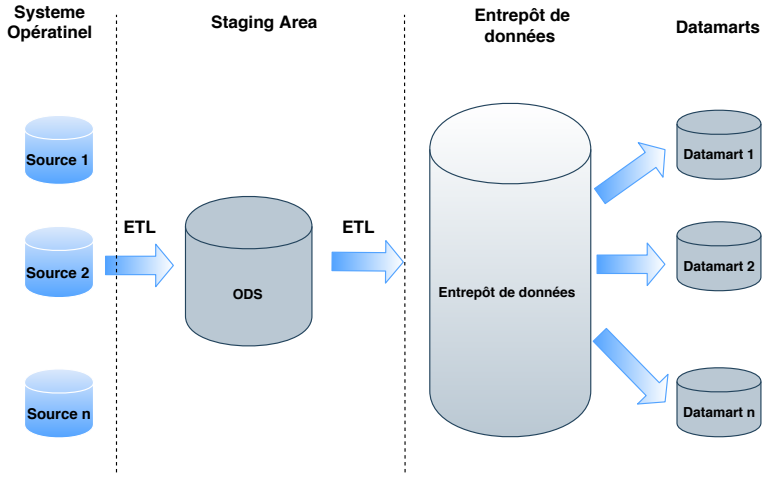


FIGURE 5.4: ODS et Staging Area

5.3.2 Extraction

L'extraction de données intervient après l'étape de conception. En effet, il faudra construire le schéma cible (tables de dimensions et faits) pour pouvoir identifier les sources potentielles pour l'alimentation de l'entrepôt. Pour chaque attribut cible, il est nécessaire de trouver la ou les sources qui correspondent. Plusieurs cas sont rencontrés lors de l'identification des sources :

— Plusieurs sources peuvent correspondre à une seule cible. Dans ce cas, il faudra choisir la source la plus pertinente et complète.

— La cible est une combinaison de plusieurs sources. Il faudra alors définir les règles de consolidation (jointure, filtre...etc).

— La cible est représentée par une partie d'une source. Il faudra alors définir les règles d'extraction et de découpage. Par exemple, la cible est le nom de la ville alors que la source contient l'adresse complète.

Après l'identification et la définition des règles de consolidation, deux modes d'extraction s'offrent à nous :

Extraction complète

Il s'agit comme son nom l'indique d'extraire toutes les données pertinentes à partir des sources. L'étape peut être coûteuse en temps, vu le nombre de données manipulées. Ce mode est généralement appliquée dans deux cas :

— Chargement initial des données dans l'entrepôt.

— Rafraîchissement complet des données dans le cas de modification dans les sources.

Extraction incrémentale

Dans ce mode, sont concernées seulement les données qui ont subit une modification ou ont été ajoutées depuis la dernières extraction. L'extraction incrémentale peut se faire de deux manières :

1. Extraction en temps réel L'extraction en temps réel se fait au moment où les changements surviennent dans les sources. Les transactions effectuées sur les sources peuvent être capturées de différentes manières :

— Utilisation des logs de transactions des bases de données. Ces logs sont à la base utilisés pour la récupération en cas de pannes.

— Utilisation des triggers (procédures déclenchées lors d'un évènement) pour recopier les données à extraire dans un fichier de sortie.

— Modification des applications sources pour transcrire tout les changements effectués dans un fichier d'extraction.

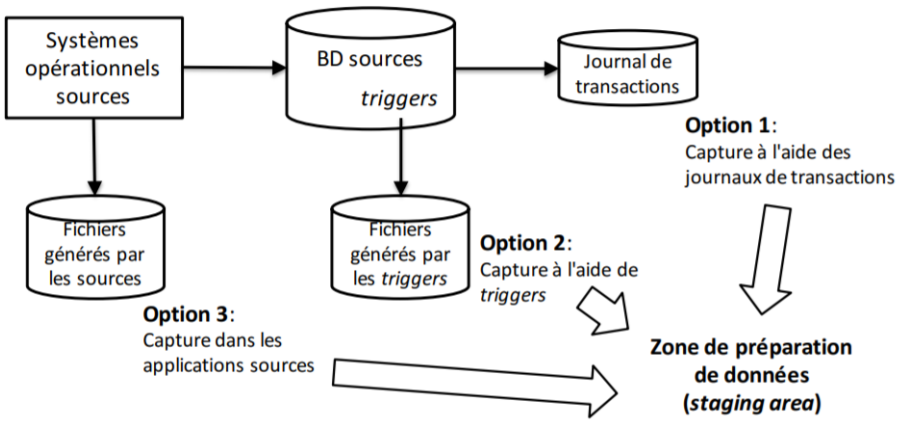


FIGURE 5.5: Extraction en temps réel

Chaque technique connaît des avantages et des inconvénients. L'utilisation de logs de transactions et de triggers ne peut malheureusement être appliquée que si les sources proviennent de bases de données, mais ont l'avantage de ne nécessiter que peu voir aucun changement dans les sources. La modification des applications sources quant à elle entraîne des coûts additionnels de développement et de maintenance, mais a l'avantage de pouvoir être utilisée sur n'importe quelle source.

2. Extraction en différée (en lot) L'extraction en différée permet de récupérer les changements réalisés sur les sources durant un intervalle prédéfini (par exemple, par heure, jour, mois...). Il est possible de capturer les modifications comme suit :

— Utilisation des timestamp : Pour chaque ajout dans le système, une estampille (timestamp) est ajoutée à la ligne. Lors de l'extraction, seulement les lignes, qui ont un timestamp plus récent que la dernière extraction, sont concernées. Cette méthode n'est pas très pratique dans le cas de suppression de lignes.

— Comparaison de fichiers : Il s'agit de comparer entre deux états successifs des sources pour pouvoir extraire seulement les différences (ajouts, modifications

et suppressions). Cette méthode exige de garder une copie de l'état des données sources de la dernière extraction et peut donc être coûteuse.

5.3.3 Transformation

C'est à cette étape du processus que les règles sont définies pour relier une cible à une ou plusieurs sources. Plusieurs types de transformations peuvent être appliqués sur les données sources pour correspondre au format des données cibles. Ci-dessous une liste non exhaustive des transformations possibles dans un outil ETL :

- Changement de format des données : changement de type, taille, etc.
- Codification des valeurs : définir des normes pour les valeurs de différentes sources. Exemple : ['homme','femme']vs['M','F'] vs ['1','0'].
- Pré-calcul des valeurs dérivées.
- Découpage de champs complexes.
- Fusion de plusieurs champs.
- Conversion des unités de mesure.
- Conversion de dates.
- Pré-calcul des agrégations.
- Etc.

5.3.4 Chargement

Le chargement consiste à transférer les données transformées vers la cible. Plusieurs options de chargements existent :

- Chargement initial : S'effectue une seule fois lors de l'activation de l'entrepôt de données. Ce chargement peut durer plusieurs heures.
- Chargement incrémental : S'effectue après un chargement initial. Il permet de charger les données extraites en temps réel ou en lots. Respecte la nature des changements (changements lents et rapides).
- Rafraîchissement complet : Est utilisé lorsque le nombre de changements est assez grand et rend par conséquent le chargement incrémental complexe.

5.4 ELT

L'ELT (Extract Load and Transform) est une solution d'intégration de données. La finalité est la même qu'une solution d'ETL mais la manière d'arriver au résultat est différente. L'étape de transformation dans une approche ETL est réalisée sur un serveur intermédiaire par des moteurs (engine) autres que ceux des SGBDs (exemple, moteur JAVA pour Talend). L'ELT quant à lui profite des fonctionnalités proposées par les SGBDs pour effectuer les transformations. Autrement dit, les données sont (1) extraites des sources, (2) chargées dans l'environnement cible puis (3) transformées. L'un des principaux attraits des outils ELT est la réduction des temps de chargement par rapport au outils ETL. Tirer parti de la capacité de traitement intégrée à une infrastructure d'entreposage de données réduit le temps de transit des données et est plus rentable.

Actuellement, la plupart des solutions d'ETL proposent des composants dits ELT.

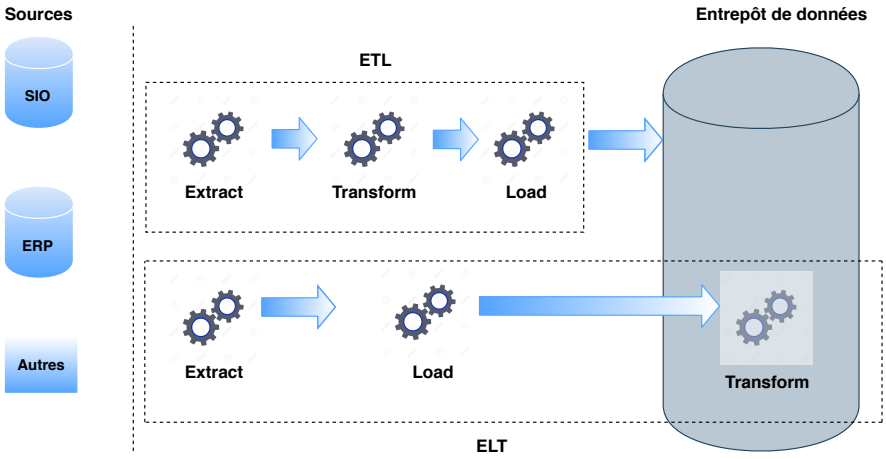


FIGURE 5.6: ETL VS ELT

Critère	ETL	ELT
Environnements	Hétérogènes	Homogènes
Transformations	Moteurs autres que SGBD	Seulement les moteurs SGBDs
Types de transformation	Très variés	Variés
Performances	Performant	Très performant
Taille de données	Importante	Très importante

TABLE 5.2: Comparaison entre les approches ETL et ELT

Chapitre 6

Stockage et interrogation d'un entrepôt de données

6.1 Introduction

La mise en place d'un système d'information décisionnel requiert l'utilisation d'outils répondant aux exigences de stockage et d'analyse de données. Autrement dit, les outils doivent respecter les caractéristiques des systèmes OLAP. Les 12 règles de Codd ont été longtemps utilisées pour décrire les principaux concepts que doivent respecter les outils OLAP.

6.1.1 Règles de Codd

Les 12 règles de Codd, disponibles sur le Web, ont pendant longtemps étaient la référence pour définir les caractéristiques qu'un outil doit avoir pour pouvoir se qualifier d'outil OLAP. Dans la réalité, la plupart des outils se définissant comme outils OLAP ne respectent pas toutes les règles. Ci-dessous les concepts clés pour un outil OLAP :

1. Conception et vue multidimensionnelle : un outil OLAP doit se baser sur un modèle multidimensionnel pour faire de l'analyse.
2. Transparence : la technologie utilisée, la conception ainsi que toutes les spécifications techniques doivent être invisibles à l'utilisateur final.
3. Accessibilité : les outils OLAP doivent permettre d'accéder aux données de façon à produire de la connaissance rapide. Une information pertinente doit être fournie en tout temps.
4. Rapidité : les montées de charges ne doivent pas freiner l'analyste. L'outil doit pouvoir supporter des requêtes complexes.
5. Architecture Client Serveur : pour un accès uniforme et des traitements plus rapides et plus complets.
6. Dimensions génériques : essayer, autant que possible, d'avoir une unicité dans la définition des concepts du modèle.
7. Gestion des matrices creuses : en mathématiques, les matrices creuses sont des matrices qui contiennent beaucoup de zéros. En informatique, il existe des algorithmes qui utilisent cette spécificité pour optimiser le stockage de ce type de matrices pour des raisons de performances. Les outils OLAP doivent avoir cette capacité d'optimisation d'espace de stockage par la gestion des matrices creuses.
8. Multi-utilisateurs : les outils OLAP sont, par définition, destinés à un accès concurrent.

9. Croisement inter dimensions illimité : l'utilisateur ne doit avoir aucune restriction quand au nombre de croisements qu'il fait entre les dimensions.
10. Intuitifs : les utilisateurs d'outils OLAP ne sont pas forcément informaticiens. Il est donc nécessaire d'offrir des solutions de haut niveaux.
11. Affichage flexible : l'utilisateur doit pouvoir aisément manipuler les résultats au format désiré.
12. Nombre illimité de dimensions et de niveaux d'agrégation.

6.1.2 Règles de FASMI

Le respect de ces règles a été remis en question en découvrant que les concepts émis ont été sponsorisé par un des fournisseurs d'outils OLAP. Un autre ensemble de caractéristiques a vu le jour et a été regroupé sous l'acronyme FASMI (Fast Analysis of Shared Multidimensional Information). La communauté actuelle s'entend pour définir un outil OLAP comme un outil respectant les caractéristiques suivantes :

— **Fast** : Le temps de réponse aux requêtes des utilisateurs ne doit pas excéder un certain délai (20 sec en moyenne). Pour cela, il est important de pré-calculer certaines données pour réduire le temps de réponse.

— **Analysis** : L'outil doit fournir aux utilisateurs la possibilité de réaliser tout type d'analyse et de calculs sur les données de l'entrepôt. L'aspect technique doit rester transparent à l'utilisateur.

— **Shared** : L'outil doit garantir la confidentialité des données et doit gérer la concurrence d'accès aux données.

— **Multidimensional** : C'est la caractéristique clé. Le système doit fournir des vues conceptuelles multidimensionnelles des données. Il doit offrir la possibilités de définir des hiérarchies.

— **Informations** : Représente la quantité de données que peut gérer l'outil et non pas uniquement la taille de données qu'il peut stocker.

Avant d'étudier les différentes manières de stocker les données, nous présentons dans la section suivante la notion clé de tout outil OLAP, le concept multidimensionnel d'un entrepôt de données.

6.2 Le cube multidimensionnel

Un hypercube OLAP (appelé cube OLAP par abus de langage) est une représentation abstraite d'informations multidimensionnelles utilisée dans les entrepôts de données. Cette structure est utilisée pour les besoins d'analyses.

Les questions métiers s'énoncent naturellement sous forme multidimensionnelle. Prenons par exemple la question suivante : "Quel est le chiffre d'affaire réalisé pour un produit X, pour la région Nord, pour le mois de juin 2018 ?" L'utilisateur cherche ici le chiffre d'affaires (mesure) selon les dimensions (ou axes) Produit, Géographique et Temps. Le cube avec sa structure multidimensionnelle et hiérarchique (capacité par exemple de passer du total Pays au détail des villes) représente de fait une structure idéale pour que l'utilisateur puisse répondre facilement et de manière autonome aux questions qu'il se pose.

Les cubes permettent d'obtenir des informations agrégées selon les besoins des utilisateurs tout en garantissant :

— Une simplicité et rapidité d'accès. Ceci est rendu possible par le fait que le cube stocke au préalable des calculs intermédiaires qui évitent de devoir tout recalculer à chaque fois. Ces calculs d'agrégats sont effectués lors de la phase alimentation du cube. De la même manière que l'alimentation d'un entrepôt, l'alimentation d'un cube peut se faire de manière complète ou incrémentale à des intervalles réguliers, en fonction de la fraîcheur des données souhaitée.

— La capacité de manipuler les données selon différentes dimensions.

Le cube est construit à partir des schémas étoiles et flocons.

Pour des raisons de visualisation, on ne peut représenter graphiquement que des hypercubes à trois dimensions mais les hypercubes peuvent contenir plusieurs dimensions.

Un cube est composé de **cellules** qui peuvent contenir plusieurs **mesures**. La localisation de la cellule est faite à travers les axes, qui correspondent chacun à une **dimension**. La dimension est composée de **membres** qui représentent les différentes valeurs de la dimension.

le cube décrit dans la figure 6.1 contient des cellules comprenant une mesure, organisées en fonction de trois dimensions nommées pièces, régions et années. La cellule grisée dans le cube est l'intersection des membres suivants :

— Le membre écrous de la dimension pièces.

— Le membre ouest de la dimension régions.

— Le membre 2019 de la dimension années.

— La mesure quantité.

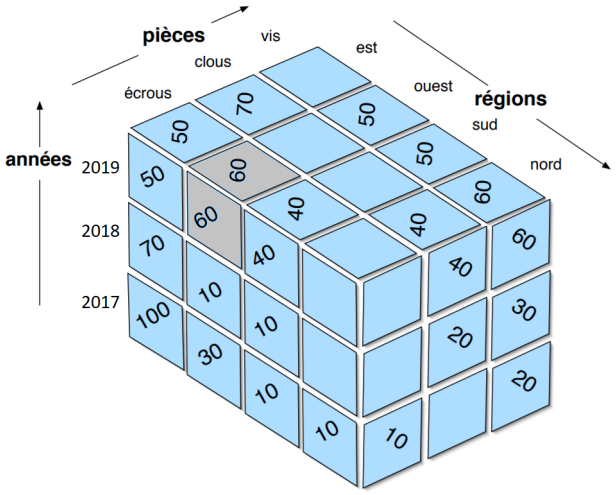


FIGURE 6.1: Exemple d'un cube multidimensionnel

On parle de treillis des cuboïdes pour décrire l'ensemble des cubes qu'il est possible de construire à partir d'un modèle ayant plusieurs dimensions. La figure 6.2 décrit l'ensemble des hypercubes qu'il est possible de construire à partir de quatre dimensions A, B, C et D.

Construire un cube contenant toutes les dimensions pour y stocker toutes les données n'est pas forcément la meilleure manière de procéder pour répondre

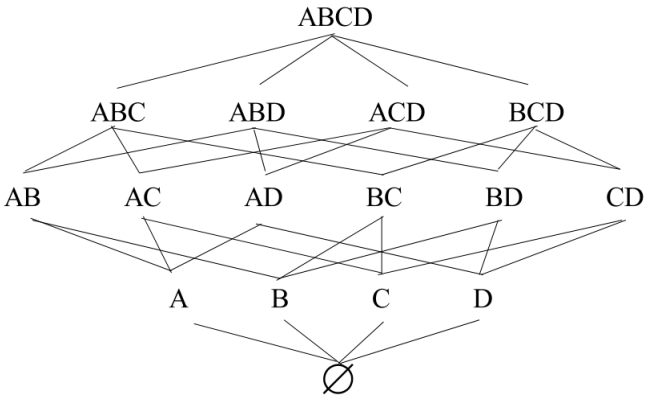


FIGURE 6.2: Treillis des cuboïdes pour 4 dimensions

rapidement à des requêtes d'analyse. Il faut savoir que la majorité des requêtes ne portent généralement que sur un ensemble réduit de données. Il est donc important lors de la conception des cubes de trouver le juste équilibre en définissant :

— Quelles données stockées.

— Quels axes d'analyse choisir.

— Quels agrégats définir.

6.2.1 Cellules feuille et non-feuille

Dans un cube, on distingue deux types de cellules en fonction de comment est obtenue la valeur de la cellule :

— **Cellule feuille :** La valeur de ce type de cellule est directement obtenue à partir de la table de faits. Les membres de dimensions utilisés pour identifier la cellule dans le cube sont des membres *feuilles*. Un membre feuille n'a pas de membre enfant d'un point de vue hiérarchique, autrement dit, c'est le grain le plus fin dans une hiérarchie.

— **Cellule non-feuille :** Une cellule peut également être identifiée à l'aide de membres non-feuilles. Un membre non-feuille est un membre qui possède un ou plusieurs membres enfants. Dans notre exemple de la figure **??**, dans la dimension temps, le membre "deuxième partie" possède les membres troisième et quatrième trimestres. La valeur de la cellule dérive le plus souvent de l'agrégation de membres enfants associés au membre non-feuille. La fonction d'agrégation pour la construction de cubes est l'une des fonctions classiques comme Avg, Sum, Min, Max, etc.

6.3 Les opérations sur le cube

Plusieurs opérations OLAP sont disponibles pour manipuler les données pour les besoins d'analyse et de visualisation. Nous présentons les plus utilisées en les catégorisant selon leur impact.

6.3.1 Sur la structure

Les opérations agissant sur la structure visent à présenter une vue différente du cube en fonction des analyses souhaitées. Parmi les plus utilisées :

— **Projection et Sélection (Slice and Dice) :** Ces opérations correspondent aux opérations relationnelles de manipulation de données. La sélection (Dice) permet d'exprimer des conditions soit sur les mesures, soit sur les membres. Elle permet de construire un sous-cube. Quant à la projection (Slice), elle permet de définir des tranches d'un cube en spécifiant soit des dimensions, soit des niveaux de granularité.

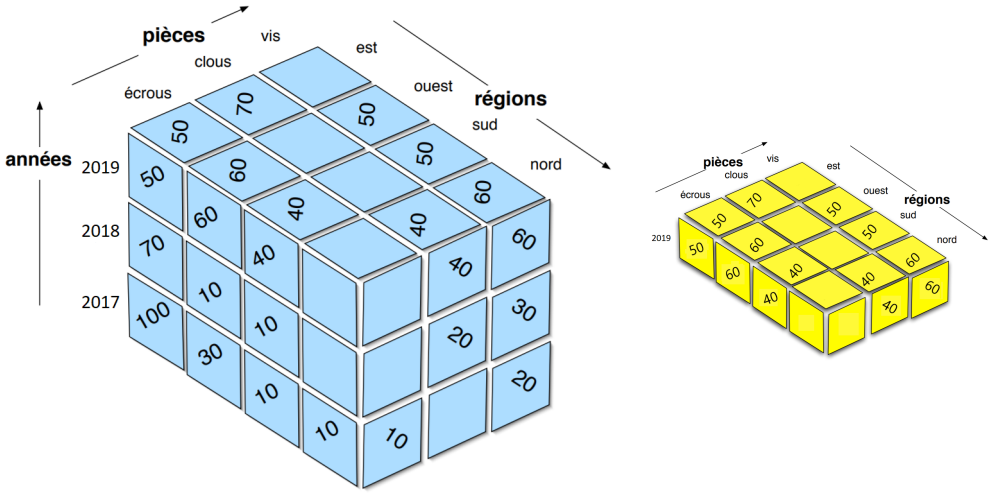


FIGURE 6.3: Projection sur la dimension temps (Année 2019)

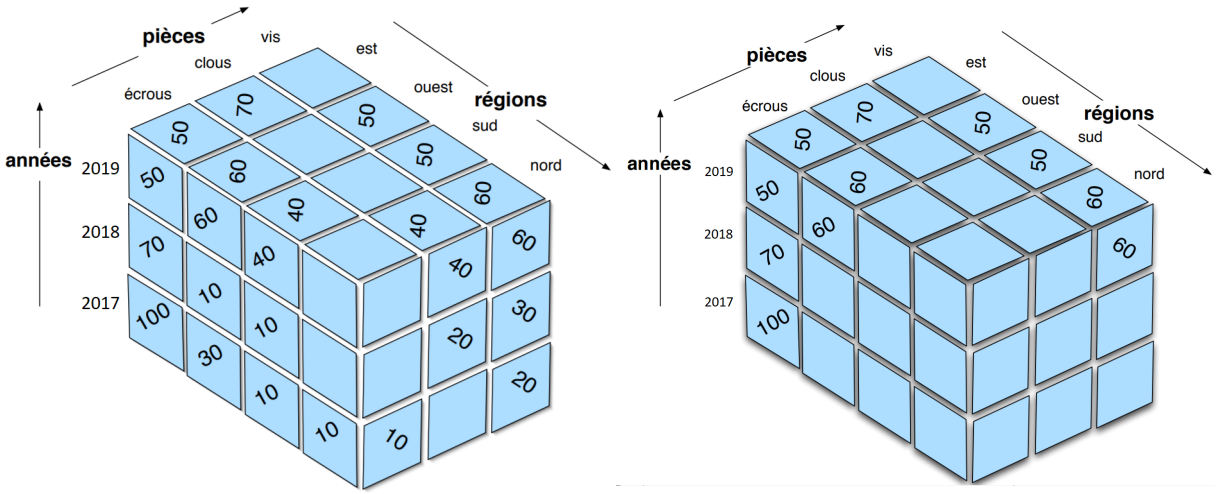


FIGURE 6.4: Sélection des ventes ≥ 50

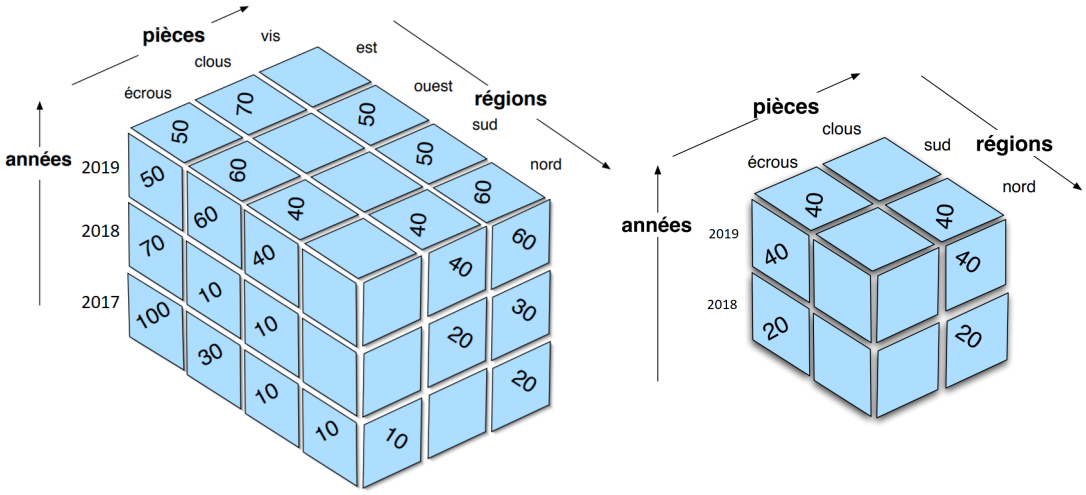


FIGURE 6.5: Sélection de plusieurs valeurs des différentes dimensions

— **Rotation (Rotate) :** Consiste à pivoter ou à effectuer une rotation du cube, de manière à présenter une vue différente des données à analyser.

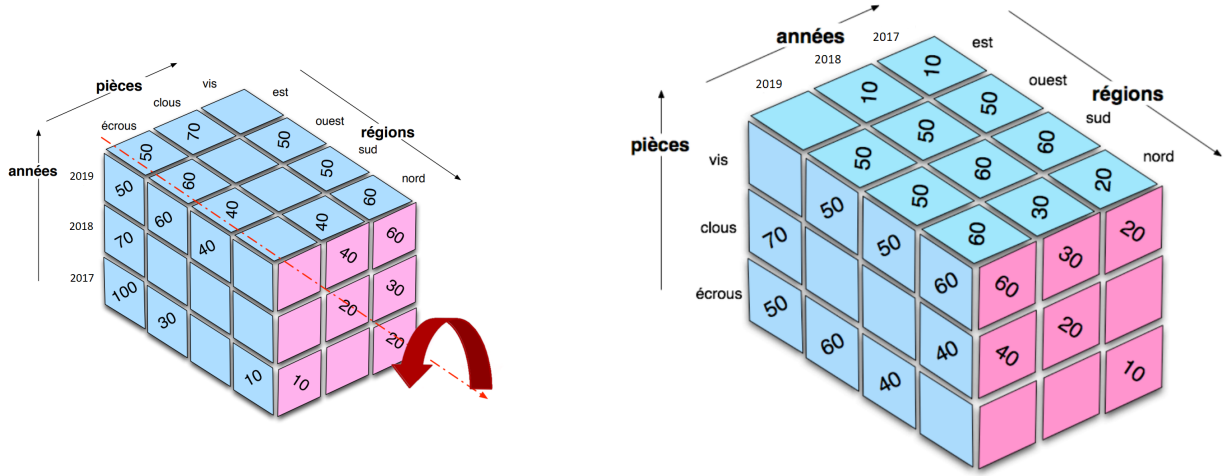


FIGURE 6.6: Rotation

— **Permutation (Switch) :** Consiste à inverser des membres d'une dimension, de manière à permuter deux tranches du cube. Certaines informations se retrouvent alors placées l'une à côté de l'autre, ce qui facilite la découverte d'un phénomène.

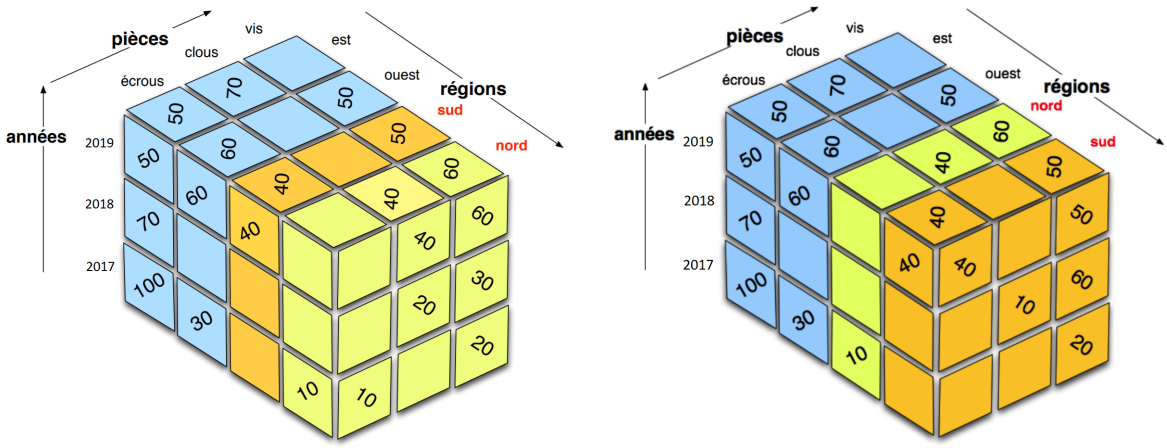
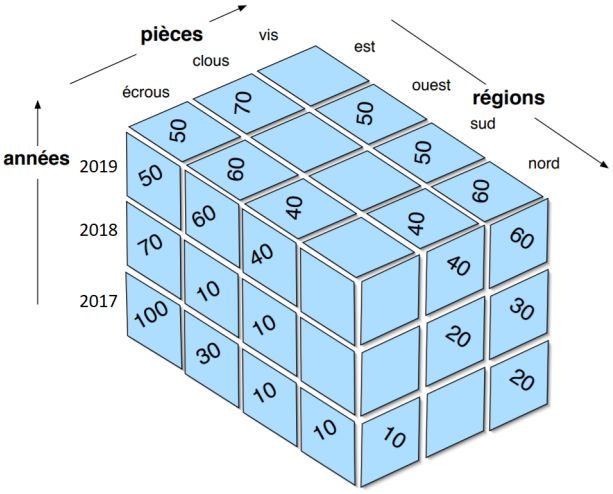


FIGURE 6.7: Permutation

— **Division (Split) :** Consiste à présenter chaque tranche du cube en passant d'une représentation tridimensionnelle à une présentation tabulaire. Le nombre de tables résultantes dépend du nombre de valeurs de la dimension choisie.



ventes (est)	2019	2018	2017
écrous	50	70	100
clous	70	70	100
vis		10	10

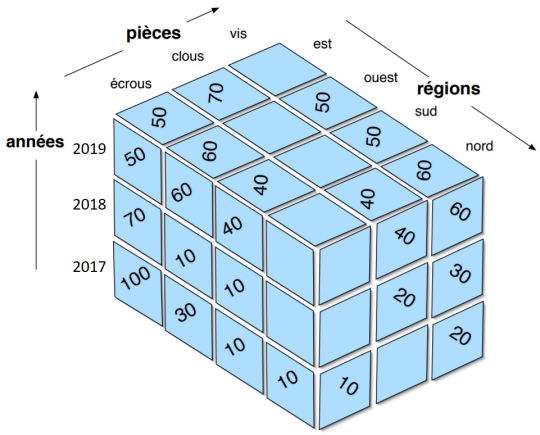
ventes (ouest)	2019	2018	2017
écrous	60	10	30
clous		10	40
vis	50	50	10

ventes (sud)	2019	2018	2017
écrous	40	10	10
clous		10	
vis	50	60	20

ventes (nord)	2019	2018	2017
écrous			10
clous	40	20	
vis	60	30	20

FIGURE 6.8: Division

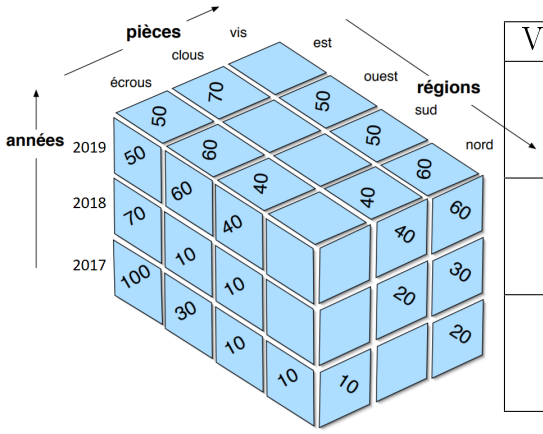
— **Emboitement (Nest) :** permet d'imbriquer les membres d'une dimension dans une autre dimension. L'intérêt de cette opération est qu'elle permet de grouper sur une même représentation bi-dimensionnelle toutes les informations (mesures et membres) d'un cube quelque soit le nombre de ses dimensions. L'opération Unnest permet de faire l'inverse.



ventes (nest)		2019	2018	2017
écrous	est	50	70	100
	ouest	60	10	30
	sud	40	10	10
	nord			10
clous	est	70	70	100
	ouest		10	40
	sud		10	
	nord	40	20	
vis	est		10	10
	ouest	50	50	10
	sud	50	60	20
	nord	60	30	20

FIGURE 6.9: Emboitement

— **Enfoncement (Push)** permet de combiner les membres d'une dimension aux mesures du cube. Autrement dit, les membres sont directement représentés dans le contenu des cellules avec les mesures. L'opération Pull permet de faire l'inverse.



Ventes (push)	est	ouest	sud	nord
écrous	2019 50 2018 70 2017 100	2019 60 2018 10 2017 30	2019 40 2018 10 2017 10	2017 10
clous	2019 70 2018 70 2017 100	2018 10 2017 40	2018 10	2019 40 2018 20
vis	2018 10 2017 10	2019 50 2018 50 2017 10	2019 50 2018 60 2017 20	2019 60 2018 30 2017 20

FIGURE 6.10: push

6.3.2 Sur le contenu

Ces opérations agissent sur la granularité des données analysées. Elles permettent de hiérarchiser la navigation entre les différents niveaux de détail d'une dimension :

— **Grain supérieur(Roll-up) :** Appelée aussi forage vers le haut, cette opération permet de représenter les données du cube à un niveau de granularité supérieur en respectant la hiérarchie de la dimension. Nous utilisons une fonction d'agrégation (somme, moyenne,...), qui est paramétrée, pour indiquer la façon de calculer les données du niveau supérieur à partir de celles du niveau inférieur.

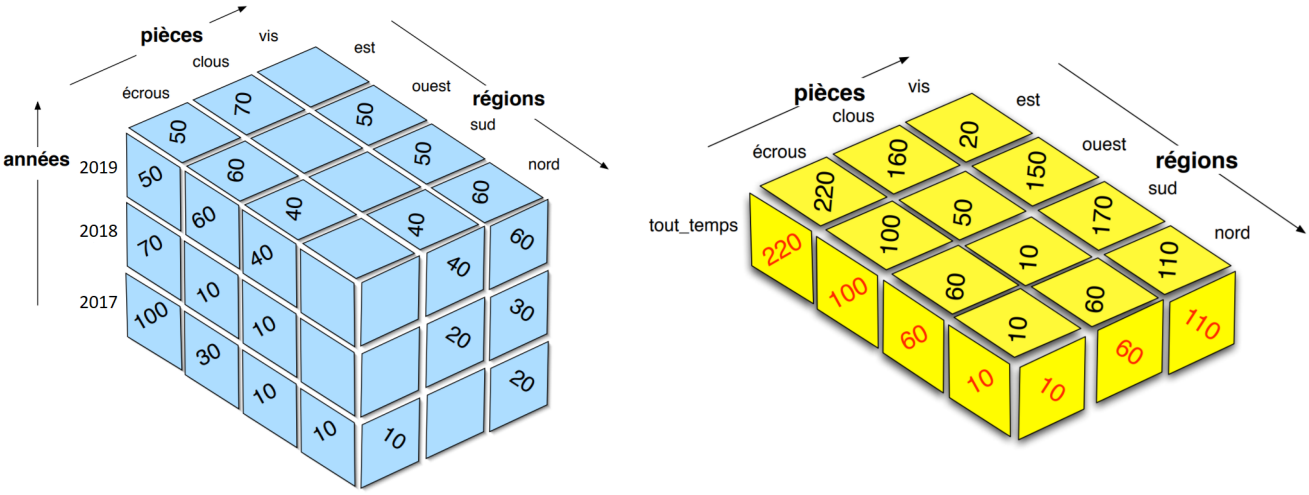


FIGURE 6.11: Grain supérieur

— **Grain inférieur (Drill-down) :** Appelée aussi forage vers le bas, cette opération permet de représenter les données du cube à un niveau de granularité inférieur, donc sous une forme plus détaillée.

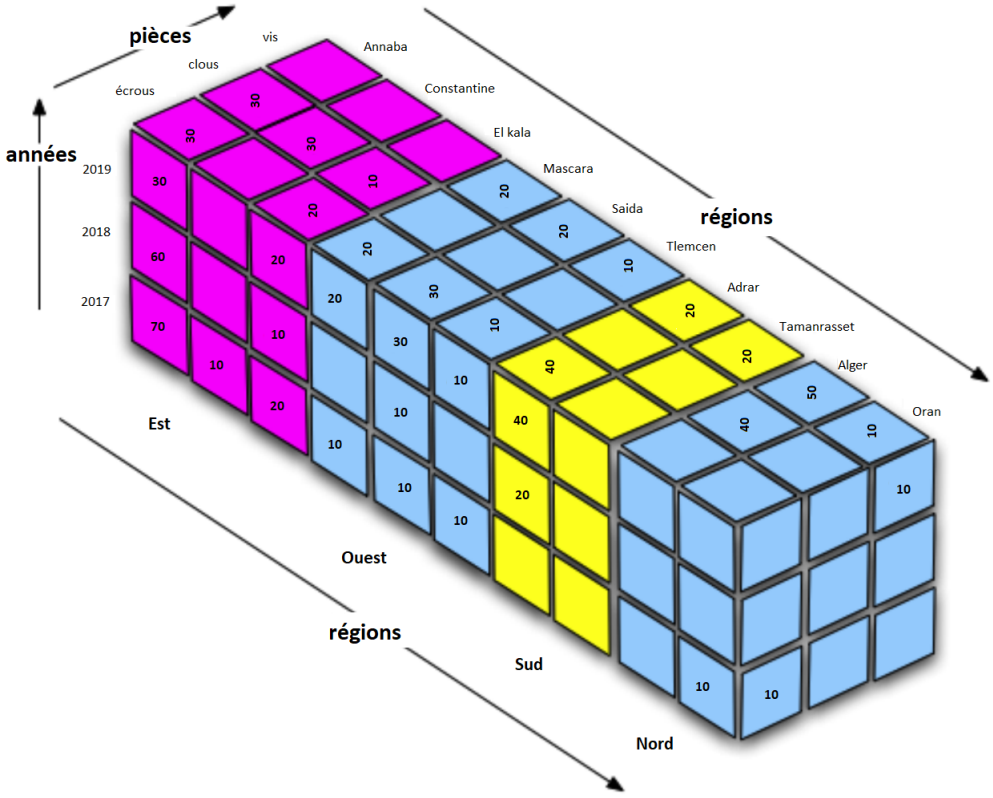


FIGURE 6.12: Grain inférieur

6.3.3 Entre cubes

— **Opérations ensemblistes (union, intersection, différence) :** Comme dans les tables relationnelles, ces opérations sont à réaliser sur des données

qui ont le même schéma.

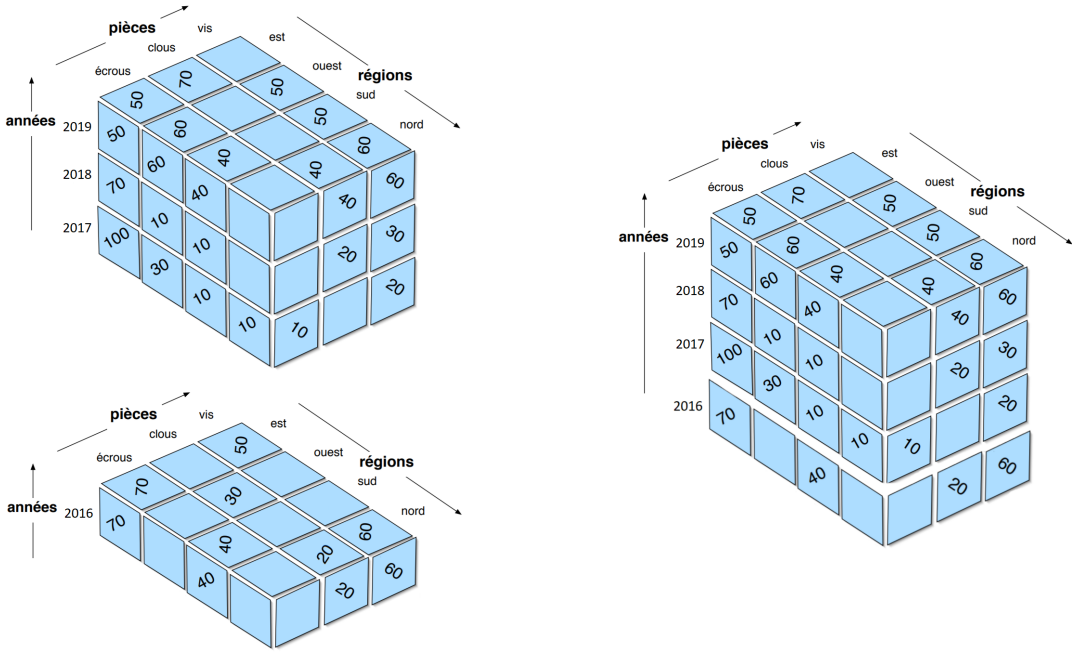


FIGURE 6.13: Union entre deux cubes

6.4 Stockage d'un entrepôt de données

Une des questions importantes dans la mise en place d'une solution OLAP est la stratégie à choisir pour le stockage physique et par conséquent les techniques de traitement de ces données. Dans les solutions proposées, Il existe trois modes principaux de stockage de l'OLAP.

6.4.1 ROLAP : Le stockage relationnel

C'est la stratégie la plus couramment utilisée pour le stockage des entrepôts. Les données sont stockées dans une base de données relationnelle. Les tables constituant le schéma en étoile(ou flocon) sont directement stockées dans la base relationnelle. Plusieurs calculs d'agrégations sont réalisés à différents niveaux.

Un moteur OLAP est ajouté comme une sur-couche à la base de données pour simuler le comportement d'une base de données multidimensionnelle. Le moteur permet entre autre de traduire automatiquement les modèles multidimensionnels en modèle de stockage relationnel. Il a également un rôle important dans l'analyse de requêtes. Il se charge de transformer une requête multidimensionnelle sur le modèle multidimensionnel en une requête relationnelle.

Le langage de requête SQL a été étendu avec de nouveaux opérateurs et fonctions pour supporter les requêtes d'analyse multidimensionnelles tels que : Cube, rollup, rank, etc. Nous présenterons quelques exemples dans la section 6.5.

La force de ce mode de stockage réside dans sa technologie relationnelle mure et un stockage de très grand volume de données. Néanmoins, le temps de réponse peut être potentiellement élevé, due aux transformations des requêtes. Il est également

impossible de réaliser certains types de requêtes qui demandent des calculs trop complexes.

Plusieurs stratégies d'optimisation sont mises en place pour rendre l'utilisation du stockage relationnel plus performante. A savoir :

— Utilisation de différentes techniques d'indexation : indexation binaire, indexe de jointure, listes inversées, etc.

— Fragmentation des tables de l'entrepôt pour l'optimisation des requêtes.

— Utilisation des vues matérialisées : Elles sont utilisées pour stocker les pré-calculs d'agrégats et par conséquents les différents cuboïdes. Le choix des agrégations à stocker est important. Il faut trouver le meilleur rapport temps d'exécution/espace de stockage. En effet, matérialiser tous les cuboïdes serait coûteux en espace alors que ne stocker que très peu serait coûteux en temps de réponse aux requêtes. Le choix de la matérialisation doit se basé sur les requêtes des utilisateurs les plus fréquentes.

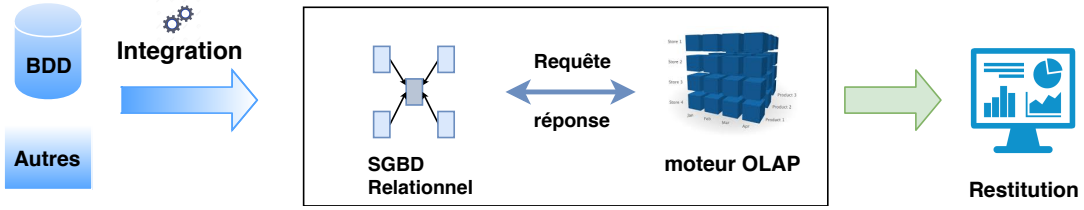


FIGURE 6.14: ROLAP

6.4.2 MOLAP : Stockage multidimensionnel

MOLAP se base sur un système purement multidimensionnel pour le stockage des cubes. Pour ce faire, il utilise la structure de tableaux. Cela permet une correspondance directe avec la vue multidimensionnelle. La deuxième caractéristique d'un système MOLAP est la réalisation de pré-agrégations et de pré-calculs de données sur tous les croisements envisageables entre les dimensions. De cette manière la restitution des données se fait de façon instantanée. Les données étant stockées, le temps gagné pendant la restitution des données est considérable. De plus, du fait de sa structure tabulaire, les calculs d'agrégats sont très rapides car cela se fait en colonne et/ou en ligne. Il n'y a aucune jointure à réaliser.

Contrairement au ROLAP qui se base sur les bases de données relationnelles (technologie qui a fait ses preuves), le MOLAP n'a pas de cadre technologique standardisé. Autrement dit, chaque produit MOLAP propose sa propre version du modèle multidimensionnel et ses stratégies de stockage.

En analogie au langage SQL adapté pour répondre aux besoins analytiques dans un système ROLAP. Plusieurs langages d'interrogation de données ont été proposés pour requêter les cubes multidimensionnels. Parmi les plus utilisés, le langage MultiDimensional eXpressions (MDX) qui sera présenté dans la section 6.5.

Dans le stockage dimensionnel, il est important de gérer la densité du cube, car toutes les cellules du cube ne contiennent pas forcement une valeur. Plusieurs techniques de compression et d'indexation sont mises en place pour réduire l'espace de stockage et rendre les accès plus performants.

Deux importants inconvénients découlent de la stratégie de stocker un grand nombre de pré-calculs :

— La Génération d'un volume trop important de données. Par conséquent, une solution MOLAP trouve rapidement ses limites en terme de performance face à des entrepôt de données volumineux.

— Les Techniques de rafraîchissement des cubes sont limitées. Il faudrait reconstruire fréquemment l'entrepôt, ce qui peut être coûteux.

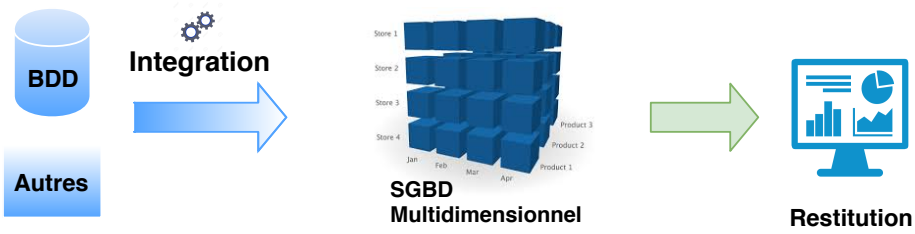


FIGURE 6.15: MOLAP

6.4.3 HOLAP : Stockage hybribe

Tout comme son nom l'indique, le mode de stockage HOLAP permet d'hybrider entre les deux points forts des deux modes ROLAP et MOLAP. Les tables de faits et de dimensions sont stockées dans un SGBD relationnel. Ce qui offre une grande capacité de stockage. Les données agrégées quant à elles sont stockées dans des tableaux. Ce qui offre de meilleurs performances en terme de réponses aux requêtes.

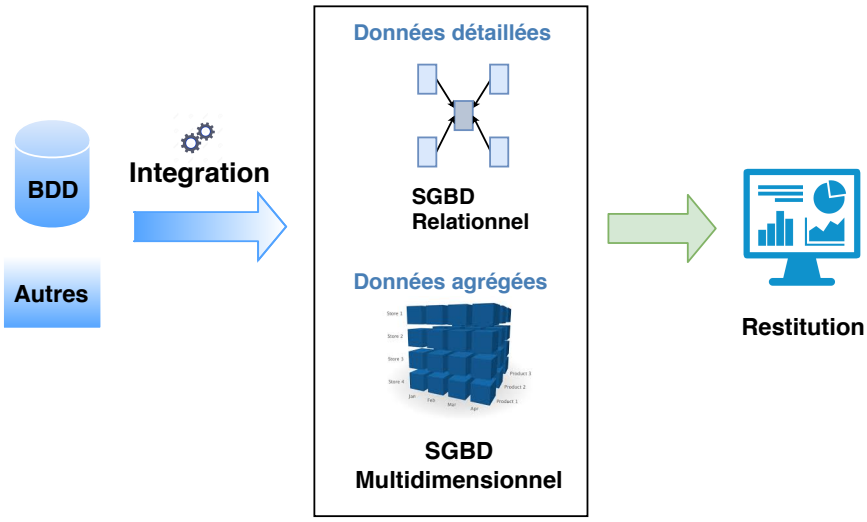


FIGURE 6.16: HOLAP

6.5 Langages d'interrogation de données

6.5.1 SQL étendu Pour l'OLAP

Le langage SQL a connu des extensions qui ont permit de répondre aux exigences des outils OLAP. Dans la version SQL99, les extensions OLAP ont été intégrées. Ces dernières permettent de créer et de naviguer dans des cubes multidimensionnelles. Nous présentons dans la suite les extensions les plus pertinentes.

Fonctions d'agrégations

Calculer la somme, la moyenne, le nombre, le max et le min se fait grâce aux fonctions d'agrégations classique de SQL. Parmi les nouvelles fonctions ajoutées, nous trouvons :

— **Fonctions Rank et Dense_Rank** : La fonction rank permet de calculer le classement d'un enregistrement comparé aux autres enregistrements basé sur les mêmes mesures. La requête suivante retourne le classement des régions par rapport aux quantité de pièces vendues.

```
SELECT region, rank() OVER   (ORDER  BY  qte DESC) AS qte
FROM ventes
ORDER BY qte
```

Il est possible d'utiliser la fonction Rank pour répondre à la question : Quel est le top 3 des régions ?

```
SELECT region, rank() OVER   (ORDER  BY  qte DESC) AS qte
FROM ventes
ORDER BY qte
FETCH FIRST 3 ROWS ONLY
```

Quant à la fonction dense_rank, elle permet de prendre en considération les cas où il y a deux tuples qui ont le même rang. Dans ce cas, le tuple suivant est de rang N+2.

— **Fonction N-tile :** Prend en entrée deux paramètres, une expression et un nombre n. La fonction calcule le domaine de l'expression en fonction des valeurs qui apparaissent dans la colonne. Ce domaine est alors divisé en n intervalles de tailles approximativement égales. La fonction N-tile permet de retourner le numéro de l'intervalle qui contient la valeur de l'expression. La requête suivante retourne le minimum, le maximum et la moyenne des quantités de pièces vendues parmi les 10% meilleures ventes.

```
SELECT Min(qte), Max(qte), Avg(qte)
FROM ventes
GROUP BY N_tile(qte,10) as Percentile
HAVING Percentile = 10;
```

— **Fonction EVERY** prend en considération les tuples que s'ils vérifient tous l'expression passée en argument. La requête suivante retourne le maximum et

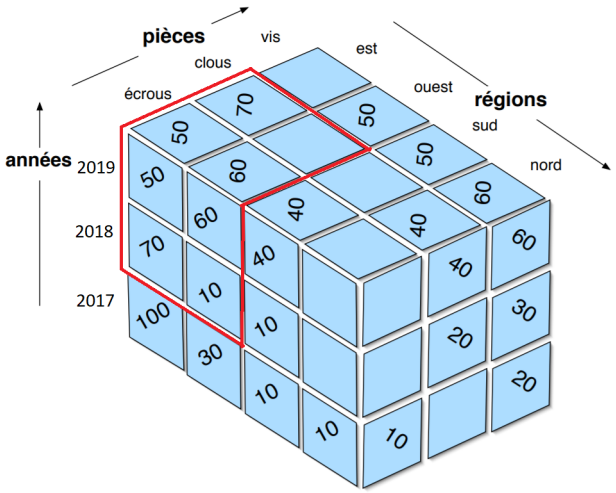
le minimum de quantité de pièces vendues pour chaque région enregistrant un taux de ventes important.

```
SELECT COUNT(*), MAX(qte), MIN(qte)
FROM ventes
GROUP BY region Having EVERY (qte > 2000000)
```

— Dans la même logique les **fonctions ANY et SOME** permettent de prendre en considération les tuples si au moins un vérifie l'expression passée en argument.

Fonctions de la clause GROUP BY

Pour mieux comprendre les différentes fonctions, nous déroulerons les différents résultats sur un sous ensemble des données du cube ventes :



Pour des raisons de simplificiation, nous prendrons en considération que les ventes des pièces *écrous* et *clous*, dans les régions *est* et *ouest* des années *2019* et *2018*.

— **ROLLUP** permet de calculer des agrégats (SUM, COUNT, MAX, MIN, AVG) à tous les niveaux d'agrégation sur une hiérarchie de dimensions. Il calcule également le total général selon l'ordre de gauche à droite dans la clause Group By. Prenons l'exemple suivant :

```
SELECT annee, region, piece, SUM(qte)
FROM ventes
GROUP BY ROLLUP (annee, region, piece)
```

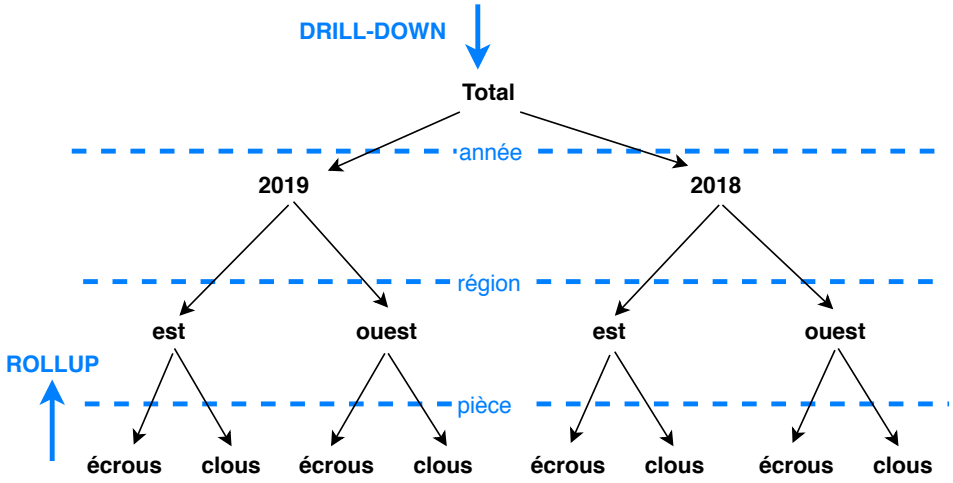


FIGURE 6.17: Rollup sur pièce,région et année

Années	Région	Pièces	Sum(qte)	Signification
2019	est	écrous	50	
2019	est	clous	70	
2019	est	-	120	Total est/2019
2019	ouest	écrous	60	
2019	ouest	-	60	Total ouest/2019
2019	-	-	180	Total 2019
2018	est	écrous	70	
2018	est	clous	70	
2018	est	-	140	Total est/2018
2018	ouest	écrous	10	
2018	ouest	clous	10	
2018	ouest	-	20	Total ouest/2018
2018	-	-	160	Total 2018
-	-	-	340	Grand Total

TABLE 6.1: Résultat du ROLLUP dans un tableau

— **CUBE** Comme la fonction ROLLUP, la fonction CUBE calcule des agrégats (SUM, COUNT, MAX, MIN, AVG) à différents niveaux d'agrégation, mais de plus, permet de calculer toutes les combinaisons d'agrégations possibles d'un ensemble de colonnes de regroupement.

```
SELECT annee, region, piece, SUM(qte)
FROM ventes
GROUP BY CUBE (annee, region, piece)
```

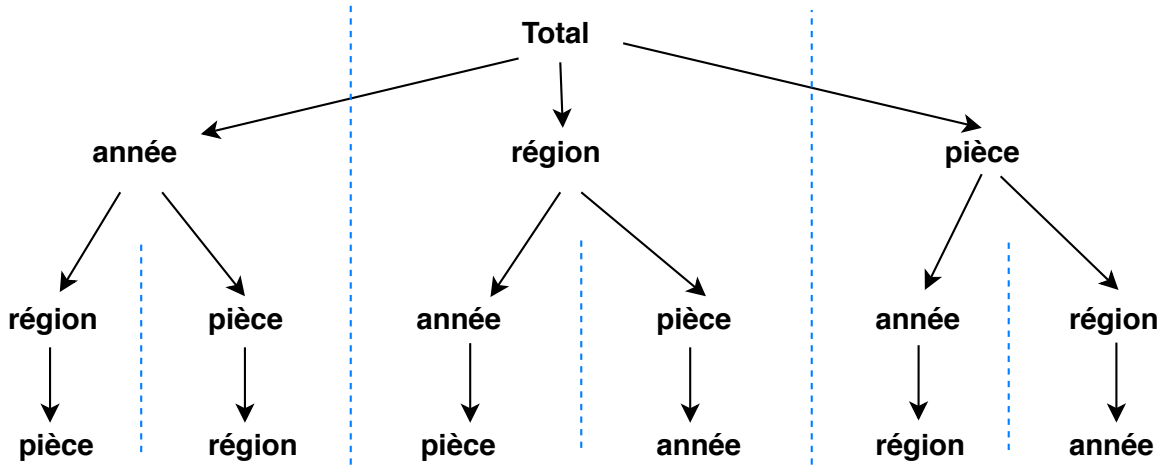


FIGURE 6.18: Cube sur pièce,région et année

Années	Région	Pièces	Sum(qte)	Signification
2019	est	écrous	50	
2019	est	clous	70	
2019	est	-	120	Total est/2019
2019	ouest	écrous	60	
2019	ouest	-	60	Total ouest/2019
2019	-	écrous	110	Total écrous/2019
2019	-	clous	70	Total clous/2019
2019	-	-	180	Total 2019
2018	est	écrous	70	
2018	est	clous	70	
2018	est	-	140	Total est/2018
2018	ouest	écrous	10	
2018	ouest	clous	10	
2018	ouest	-	20	Total ouest/2018
2018	-	écrous	80	Total écrous/2018
2018	-	clous	80	Total clous/2018
2018	-	-	160	Total 2018
-	est	écrous	120	Total écrous/est
-	est	clous	140	Total clous/est
-	est	-	260	Total est
-	ouest	écrous	70	Total écrous/ouest
-	ouest	clous	10	Total clous/ouest
-	ouest	-	80	Total ouest
-	-	écrous	190	Total écrous
-	-	clous	150	Total clous
-	-	-	340	Grand Total

TABLE 6.2: Résultat du CUBE dans un tableau

— **GROUPING SETS** Permet de spécifier des groupements multiples.

```
SELECT annee, region, piece, SUM(qte)
FROM ventes
GROUP BY GROUPING SETS((annee, region),(annee, piece))
```

Années	Région	Pièces	Sum(qte)	Signification
2019	est	-	120	Total est/2019
2019	ouest	-	60	Total ouest/2019
2019	-	écrous	110	Total écrous/2019
2019	-	clous	70	Total clous/2019
2018	est	-	140	Total est/2018
2018	ouest	-	20	Total ouest/2018
2018	-	écrous	80	Total écrous/2018
2018	-	clous	80	Total clous/2018

TABLE 6.3: Résultat du GROUPING SETS dans un tableau

6.5.2 MDX : MultiDimensional eXpressions

MDX, est un langage de requêtes OLAP pour les bases de données multidimensionnelles. Il est utilisé pour naviguer dans les bases multidimensionnelles et définir des requêtes sur tous leurs objets (dimensions, hiérarchies, niveaux, membres et cellules). Une requête MDX retourne un rapport à plusieurs dimensions consistant en un ou plusieurs tableaux 2D. Il a été utilisé par de nombreux outils OLAP.

La syntaxe de MDX ressemble à celle de SQL par ses mots clé SELECT, FROM, WHERE, mais leurs sémantiques sont différentes : SQL construit des vues relationnelles alors que MDX construits des vues multidimensionnelles des données. La tableau 6.4 illustre l'analogie entre les deux concepts :

Multidimensionnel (MDX)	Relationnel (SQL)
Cube	Table
Niveau	Colonne
Dimension	Ensemble de colonnes liées
Mesure	Colonne
Membre de dimension	Valeur

TABLE 6.4: Analogie entre les langages MDX et SQL

Le but de cette section n'étant pas de présenter la syntaxe complète du langage MDX, nous présentons dans ce qui suit quelques concepts clés et la structure générale d'une requête MDX. Pour ce faire, nous présenterons des exemples simples sur un cube de données contenant les résultats des étudiants dans les différentes facultés d'une université.

— Dans un cube, une dimension peut contenir un ou plusieurs niveaux (levels). Par exemple, la dimension temps peut avoir comme niveaux : Année, mois et jour.

— Un niveau est lui même composé de membres. Par exemple, le niveau Année contient les membres : 2017, 2018. En MDX, le choix du niveau s'exprime comme suit :

```
[Annee].[2017]
[Annee].[Mois].[Mars]
```

— Il est possible de trouver plusieurs mesures dans un cube. En MDX, les mesures sont les membres de la dimension Measures. Si aucune mesure n'est spécifiée dans la requête, c'est la mesure définie par défaut qui est retournée.

```
[Measures].[noteCC]
[Measures].[noteExam]
```

— Un tuple est une suite de plusieurs membres entre parenthèses séparées par une virgule.

```
([Annee].[2017],[Faculte].[MathInfo])
```

— Un set est un ensemble ordonné de tuples entre accolades. Les sets sont utilisés pour retourner différentes combinaison de données. Il peut retourner deux membres différents, deux mesures différentes,... Le but étant de juxtaposer les résultats pour inférer de la connaissance. Dans Les exemples suivants, il est possible de comparer le taux de réussite des étudiants dans les deux facultés, ou bien de comparer entre les résultats dans les contrôles continu et les examens d'une même faculté. L'utilisation des Sets permet d'exprimer plusieurs types de combinaison.

```
{
 ([Annee].[2017],[Faculte].[Chimie]),
 ([Annee].[2017],[Faculte].[MathInfo])
}

{
  ([Measures].[NoteCC],[Faculte].[Chimie]),
  ([Measures].[noteExam],[Faculte].[Chimie])
}
```

Syntaxe de base d'une requête

```
Select [<spécification des axes>...]
From [<spécification d'un ou plusieurs cubes>]
Where [<spécification des filtres>]
```

Spécification des axes La clause Select permet de déterminer les axes du résultat de la requête. Un Set suivi du mot clé ON suivi d'un nom d'axe spécifique. Ce dernier

fait référence à un numéro d'ordre s'il y a plus de 2 axes retournés, ou simplement aux noms d'axes explicites COLUMNS et ROWS.

```
SELECT {([Measures].[NoteCC], [Faculte].[Chimie]),
        ([Measures].[NoteCC], [Faculte].[MathInfo])} ON COLUMNS,
       {([Annee].[2017]),
        ([Annee].[2018])} on rows
FROM [Reussite]
```

ou

```
SELECT {([Measures].[NoteCC], [Faculte].[Chimie]),
        ([Measures].[NoteCC], [Faculte].[MathInfo])} ON AXIS(0),
       {([Annee].[2017]),
        ([Annee].[2018])} on AXIS(1)
FROM [Reussite]
```

Cette requête retourne la moyenne des Note de contrôle continus dans les faculté MathInfo et Chimie pour les années 2017 et 2018.

Spécification d'un ou plusieurs cubes La clause From permet de spécifier le ou les cubes nécessaires pour la création du résultat. Dans le cas de plusieurs cubes, une jointure multidimensionnelle est appliquée. Il faut pour cela, qu'il y ai au moins une dimension commune entre les cubes (l'équivalent des attributs de jointures en SQL).

Spécification de filtres Comme en SQL, la clause Where permet de spécifier les filtrer sur les cellules. Les filtres sont aussi appelés Slices et ont le même rôle que l'opérateur Slice défini plus haut. Supposons ne vouloir que les moyennes concernant les étudiants de sexe féminin. Le résultat est obtenu grâce à la requête suivante :

```
SELECT {([Measures].[NoteCC], [Faculte].[Chimie]),
        ([Measures].[NoteCC], [Faculte].[MathInfo])} ON COLUMNS,
       {([Annee].[2017]),
        ([Annee].[2018])} on rows
FROM [Reussite]
WHERE ([Etudiant].[Sexe].[F])
```

Comme dans SQL, il est possible d'appliquer certaines fonctions prédéfinies. Ci-dessous une liste non exhaustive des fonctions les plus utilisées :

Fonctions sur les membres

Il existe plusieurs fonctions qui permettent de naviguer dans les hiérarchies de dimensions.

— Children : Retourne tous les membres du niveau immédiatement en dessous de celui-ci. Par exemple, les enfants du membre 2017 sont tous les mois de l'année.

```
[Annee].[2017].Children
```

— Members : Retourne les membres du niveau choisi. Par exemple, l'instruction suivante affichera toutes les années.

```
[Annee].Members
```

— Parent, FirstChild, LastChild, NextMember, Prev Member : Permettent de naviguer verticalement et horizontalement au sein d'une dimension.

— Etc.

Fonctions sur les sets

Les fonctions utilisées sur les sets sont les fonctions classiques appliquées sur les ensembles.

— Order : Permet de trier les éléments d'un set.

— Head, Tail : permettent de retourner les premiers et les derniers élément d'un Set. L'exemple suivant permet de retourner les deux premiers membres de la dimension Année

```
Head([Annee].Members,2)
```

— Filter : Il est possible d'appliquer un filtre sur les set. L'instruction suivante retourne que les mois où les moyennes de contrôle continu étaient supérieures à 15

```
Filter([Annee].Children, ([Measures].[NoteCC]) > 15)
```

— Intersect, Union et CrossJoin

— Subset, TopCount...

— Etc.

Fonctions avancées

De plus des fonctions de base, le langage MDX offre des fonctions avancées permettant une analyse plus poussée.

— **ParallelPeriod** : Permet de réaliser des analyses comparatives. Cela consiste à présenter à côté d'une mesure, la même mesure sur une période parallèle qui lui est antérieure.

— **Descendants** : Permet de réaliser des calculs cumulatifs sur une période de temps définie. Plusieurs paramètres peuvent être utilisés pour affiner les analyses.

— **IIF** : Permet de définir des tests conditionnels sur les mesures. Les conditions sont généralement exprimées sur l'évolution des mesures [Evolution], le pourcentage de l'évolution [%] et les performances réalisées [Performances]. Il existe des fonctions prédéfinies qui calculent chacune de ces métriques.

Chapitre 7
Conclusion

Avant de commencer tout projet décisionnel, il est important de fixer au préalable les objectifs souhaités et d'avoir une vue globale sur les données sources. Ces deux éléments sont à la base du choix de la conception et de l'architecture à mettre en place.

La phase la plus fastidieuse dans la mise en place d'un entrepôt de données est l'étape d'alimentation. La définition des règles de transformation et d'extraction conditionne la qualité du produit final. S'ajoute à cela, la phase d'assainissement de données qui peut être chronophage. Il est très important de s'assurer de la qualité et la fiabilité des données. En effet, un système d'information décisionnel basé sur des données non fiables peut être plus néfaste que la non mise en place du système.

Nous avons présenté dans ce polycopié, les différents concepts clés à connaître pour mettre en place un système d'information décisionnel. Nous avons introduit l'importance d'un tel système et les enjeux de sa réalisation. Nous nous sommes focalisés sur les notions de base, qui sont : le fait, la dimension et leurs caractéristiques. Un panorama des différentes architectures est présenté ainsi que les modèles conceptuels d'un entrepôt de données. Nous avons mis l'accent sur les différentes solutions mises en place pour assurer l'historisation de données. Cette dernière représente une des caractéristiques principales d'un système d'information décisionnel. Nous avons présenté, par la suite, les différentes approches d'intégration de données et détaillé l'approche ETL qui représente l'approche la plus utilisée dans le domaine décisionnel.

Le dernier point dans la mise en place d'un système d'information décisionnel est le choix de la stratégie de stockage. Nous avons vu que chaque stratégie comporte des avantages mais également des inconvénients. Le choix dépend des ressources du projet décisionnel. Pour finir, nous avons présenté deux langages de requêtes permettant d'interroger les données de l'entrepôt. Il est à rappeler qu'il existe plusieurs solutions pour exploiter les données de l'entrepôt, des plus simples comme les requêtes au plus complexes comme l'application d'algorithme de fouille de données. Dans notre polycopié, nous nous sommes intéressés uniquement aux langages de requêtes.

Bibliographie

[Bou18] Omar Boussaïd. Support de cours : Systèmes d'information décisionnels. 2018.

[Den08] Marie-Chantal Denis. *Conception et réalisation d'un entrepôt de données institutionnel dans une perspective de support à la prise de décision.* PhD thesis, Université du Québec à Trois-Rivières, 2008.

[ESP15] Bernard ESPINASSE. Supports de cours. 2015.

[FN13] Emmanuel Ferragu and Philippe Nieuwbourg. *Modélisation des systèmes d'information décisionnels techniques de modélisation conceptuelle et relationnelle des entrepôts de données.* Vuibert, Paris, 2013.

[GR09] Matteo Golfarelli and Stefano Rizzi. *Data Warehouse Design : Modern Principles and Methodologies.* McGraw-Hill, Inc., 1 edition, 2009.

[JLVV13] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, and Panos Vassiliadis. *Fundamentals of data warehouses.* Springer Science & Business Media, 2013.

[KR03] Ralph Kimball and Margy Ross. *Entrepôts de données : guide pratique de modélisation dimensionnelle.* Vuibert informatique, 2003.

[Neg18] Elsa Negre. Support de cours : Entrepôts de données. 2018.

[Sol08] Lydie Soler. Support de cours : Les entrepôts de données. 2008.

[Tri05] Serna Encinas María Trinidad. *Entrepôts de données pour l'aide à la décision médicale : conception et expérimentation.* PhD thesis, Université Joseph-Fourier-Grenoble I, 2005.

[VZ14] Alejandro Vaisman and Esteban Zimányi. *Data Warehouse Systems : Design and Implementation.* Springer, Heidelberg, 2014.