Année Académique 2022- 2023

Ministère de l'Economie Numérique, des Télécommunication et de l'Innovation

ESATIC
ECOLE SUPÉRIEURE AFRICAINE DES TIC

Cours d'informatique décisionnel

Business Intelligence

Cycle Master (2)

Spécialité :
Système Informatique et Génie Logiciel
(SIGL)

Enseignant : Dr BROU Pacôme, Maître-Assistant
(Mathématiques appliquées, Optimisation & Telecom)
Direction de la Recherche et de l'Innovation Technologique
Laboratoire des Sciences et Technologie de l'Information et de la Communication

Chapitre 1 : Introduction aux Entrepôts de Données

1. Introduction

Formalisé au début des années 1990, le concept d'Entrepôt de Données (ED) en anglais Datawarehouse (DW) est devenu la clé de voûte de ce qui est appelé aujourd'hui «l'information décisionnelle ». Son fort développement a été poussé par un effet de mode renforcé par l'occasion saisie par les fournisseurs du marché informatique (constructeurs, intégrateurs de systèmes, éditeurs de logiciels), qui y ont trouvé le moyen de proposer des nouveaux produits remplissant plus ou moins bien les fonctions demandées par les utilisateurs.

Les systèmes d'information (SI) ont connu une longue suite d'innovations.

- Sur les infrastructures (en particulier les structures client/serveur, intranet/internet)
- Sur les outils (en particulier les bases de données relationnelles et le développement des langages objets).

Mais l'entrepôt de données ne s'inscrit pas dans cette séquence : L'entrepôt n'est pas une nouvelle plateforme technologique. Il ne s'agit pas uniquement d'outils et de techniques. Avant tout, l'entrepôt de données doit considérer les besoins de l'entreprise. Son développement demande une approche métier.

Il faut cependant se méfier d'une démarche simpliste de présentation :

- L'expertise associée à la notion de démarche métier n'est pas une simple connaissance des produits du marché,
- Bien qu'il s'agisse de traiter des grands volumes de données, les performances du système ne s'appuient pas nécessairement sur l'utilisation de machines ou d'architectures massivement parallèles,
- Un entrepôt de données n'est pas simplement un SGB haut de gamme,
- Un entrepôt de données ne se résume pas à un logiciel de présentation de données, même si on peut donner à cette partie du système une importance stratégique (partie visible, donc partie vedette).

En résumé, la définition d'un entrepôt de données ne se réduit pas à une interface de présentation associée à une base de données. Ce n'est pas faux, mais la clé est dans la pertinence des contenus : l'adéquation des données au mécanisme de décision.

2. Présentation

Un système d'information décisionnel (SID) viable implique avant tout la mise en place de deux éléments essentiels :

- Un modèle de données spécifique et
- Une infrastructure d'alimentation

La construction de ces modèles sera étudiée plus en détail dans les chapitres suivants. Leur élaboration est l'œuvre de génie logiciel et d'intégration.

Un système d'Information Décisionnel est bien un projet stratégique, ce qui ne signifie pas que c'est nécessairement un projet de taille démesurée. D'une façon réaliste les volumes de données sont plus de l'ordre de la centaine de giga-octets que du tera-octet.
Un SID ne se définit pas de façon académique, on peut cependant donner quelques caractéristiques fondamentales, en examinant les objectifs qu'il doit servir. Ces objectifs sont

définis par l'expression des besoins des gestionnaires d'entreprises pour réaliser des tâches de prise de décision. La réalisation de ces tâches nécessite :

- d'accéder aux (montagnes de) données, qui existent dans leur société,
- de faire des tris, des regroupements, des coupes en tranche, en dés et en toutes sortes de façon dans les données,
- d'accéder facilement et directement aux données qui leurs sont nécessaires,
- de montrer seulement ce qui est important,
- d'obtenir des valeurs cohérentes et comparables même lorsque les sources sont différentes,
- d'obtenir des valeurs fiables et objectives qui permettent de prendre des décisions.

On peut alors transformer l'expression de ces besoins, sous forme d'une sorte de cahier des charges d'un entrepôt de données, d'une part relativement aux fonctions qu'il doit remplir et d'autre part, relativement à sa structure.

L'idée de constituer une base de données orientée sujet, intégrée, contenant des informations datées, non volatiles et exclusivement destinées aux processus d'aide à la décision fut dans un premier temps accueillie avec une certaine perplexité.

Mais l'économie actuelle en a décidé autrement. Les entreprises sont confrontées à une concurrence de plus en plus forte, des clients de plus en plus exigeants, dans un contexte organisationnel de plus en plus complexe et mouvant.

Pour faire face aux nouveaux enjeux économiques, l'entreprise doit anticiper. L'anticipation ne peut être efficace qu'en s'appuyant sur de l'information pertinente. Cette information est à la portée de toute entreprise qui dispose d'un capital de données gérées par ses systèmes opérationnels et qui peut en acquérir d'autres auprès de fournisseurs externes.

Mais actuellement, les données sont surabondantes, non organisées, dans une perspective décisionnelle et éparpillées dans de multiples systèmes hétérogènes.

Pourtant, les données représentent une mine d'informations. Il devient fondamental de rassembler et d'homogénéiser les données afin de permettre d'analyser les indicateurs pertinents pour faciliter les prises de décisions.

Pour répondre à ces besoins, le nouveau rôle de l'informatique est de définir et d'intégrer une architecture qui serve de fondation aux applications décisionnelles : l'entrepôt de données ou Datawarehouse.

3. Pourquoi un Entrepôt de Données (Data Warehouse) ?

3.1. La problématique des Entreprises

L'entreprise construit un système décisionnel pour améliorer sa performance. Elle doit décider et anticiper en fonction de l'information disponible et capitaliser sur ses expériences.

Depuis plusieurs dizaines d'années, une importante masse d'informations est stockée sous forme informatique dans les entreprises. Les systèmes d'information sont destinés à garder la trace d'événements de manière fiable et intègre. Ils automatisent de plus en plus les processus opérationnels.

Parallèlement, les entreprises réalisent la valeur du capital d'information dont elles disposent. Au-delà de ce que l'informatique leur apporte en terme fonctionnel, elles prennent conscience de ce qu'elle pourrait apporter en terme de contenu informationnel.

Considérer le système d'information sous cet angle en tant que levier pour accroître leur compétitivité et leur réactivité n'est pas nouveau. Par contre, étant donné l'environnement concurrentiel actuel, cela devient une question de survie.

L'informatique a un rôle à jouer, en permettant à l'entreprise de devenir plus entreprenante et d'avoir une meilleure connaissance de ses clients, de sa compétitivité ou de son environnement.

Il est intéressant de calculer les retours sur investissement rendus publics. Ils se calculent rarement en termes de baisse de coûts, mais en terme de gains. Par exemple, ils permettent un meilleur suivi des ventes, une meilleure compréhension des habitudes d'achats des clients, d'une adaptation des produits à une clientèle mieux ciblée.

A ce titre, le Datawarehouse doit être rapproché de tous les concepts visant à établir une synergie (association) entre le système d'information et sa stratégie.

3.2. La réalité des systèmes d'information

A première vue, les systèmes opérationnels seraient des mines d'or informationnelles. En fait, il n'en est rien.

Les données contenues dans ces systèmes sont :

- Eparpillées : il existe souvent de multiples systèmes, conçus pour être efficace pour les fonctions sur lesquelles ils sont spécialisés.
- Peu structurées pour l'analyse : la plupart des systèmes informatiques actuels ont pour objet de conserver en mémoire l'information, et sont structurés dans ce but.
- Focalisées pour améliorer le quotidien : toutes les améliorations technologiques se sont focalisées pour améliorer cette capacité en termes de volume, qualité, rapidité d'accès. Il manque très souvent la capacité à nous donner les moyens de tirer parti de cette mémoire pour prendre des décisions.
- Utilisées pour des fonctions critiques : la majorité des systèmes existants est conçue dans le but unique de nous servir avec des temps de réponse corrects.

Le Tableau *Tab1* Présente les différences entre les données opérationnelles et les données décisionnelles.

Données opérationnelles	Données décisionnelles
Orientées application, détaillées, précise au moment de l'accès	Orientées activité (thème, sujet), condensées, représente des données historiques
Mise à jour interactive possible de la part des utilisateurs	Pas de mise à jour de la part des utilisateurs
Accédées de façon unitaire par une personne à la fois	Utilisées par l'ensemble des analystes, gérées par sous-ensembles
Cohérence atomique	Cohérence globale
Haute disponibilité en continu	Exigence différente, haute disponibilité ponctuelle
Unique (pas de redondance en théorie)	Peuvent être redondantes
Structure statique, contenu variable	Structure flexible
Petite quantité de données utilisées par un traitement	Grande quantité de données utilisées par les traitements
Réalisation des opérations au jour le jour	Cycle de vie différent
Forte probabilité d'accès	Faible probabilité d'accès
Utilisées de façon répétitive	Utilisée de façon aléatoire

Tab 1. Différences entre données du système de production et données décisionnelles

S'il existe effectivement des informations importantes, il n'en est pas moins nécessaire de construire une structure pour les héberger, les organiser et les restituer à des fins d'analyse.

Cette structure est « l'entrepôt de données » ou le Data Warehouse . Ce n'est pas une usine à produire l'information, mais plutôt un moyen de la mettre à la disposition des utilisateurs de manière efficace et organisée.

La mise en œuvre du Datawarehouse est un processus complexe. L'objectif à atteindre est de recomposer les données disponibles pour en donner :

- Une vision intégrée et transversale aux différentes fonctions de l'entreprise,
- Une vision métier au travers de différents axes d'analyse,
- Une vision agrégée ou détaillée suivant le besoin des utilisateurs.

Le Data Warehouse permet la mise en place d'un outil décisionnel s'appuyant sur les informations pertinentes pour l'entreprise, centrées sur le métier utilisateur.

3.3 Pourquoi pas un SGBD ?

Fonctions d'un SGBD (*Fig.1*)

- Systèmes transactionnels (OLTP),
- Permettre d'insérer, modifier, interroger rapidement, efficacement et en sécurité les données de la base,
- Sélectionner, ajouter, mettre à jour, supprimer des tuples,
- Répondre à de nombreux utilisateurs simultanément.

Fonctions d'un DW (*Fig.1*)

- Aider à la prise de décision (OLAP)
- Regrouper, organiser des informations provenant de sources diverses

- Intégrer et stocker les données pour une vue orientée métier
- Retrouver et analyser l'information rapidement et facilement

	OLTP	DW
Utilisateurs	Nombreux Employés	Peu Analystes
Données	Alphanumériques Détaillées / atomiques Orientées application Dynamiques	Numériques Résumées / agrégées Orientées sujet Statiques
Requêtes	Prédéfinies	« one-use »
Accès	Peu de données (courantes)	Beaucoup d'informations (historisées)
But	Dépend de l'application	Prise de décision
Temps d'exécution	Court	Long
Mises à jour	Très souvent	Périodiquement

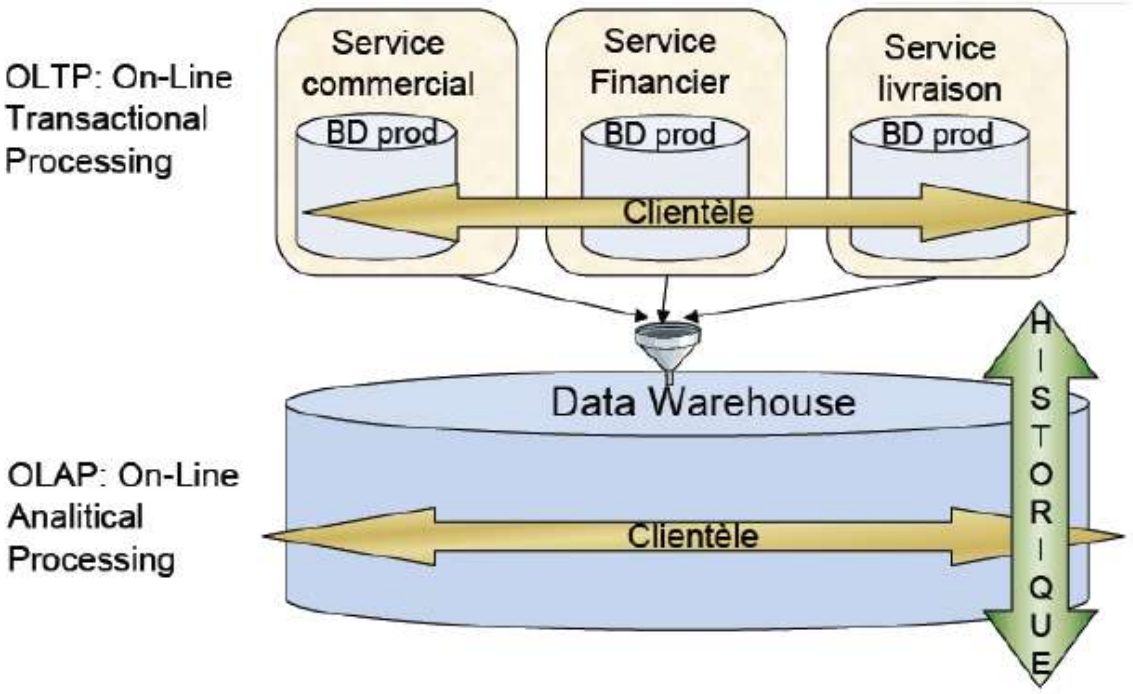


Fig. 1. OLTP vs DW

3.4. Les fonctions attendues d'un entrepôt de données

- L'entrepôt de données doit rendre les données de l'organisation facilement accessibles. Le contenu de l'entrepôt doit être facile à comprendre. Les données doivent être parlantes et leur signification évidente pour l'utilisateur. Pour être lisible, le contenu doit être étiqueté de manière significative.
- Les outils d'accès doivent être simples et faciles à utiliser, avec des temps de réponses minimes. Ils doivent permettre de séparer et combiner les données de toutes sortes de façons.
- L'entrepôt de données doit présenter l'information de manière cohérente. Les données doivent être crédibles, même si elles sont assemblées à partir de plusieurs sources d'information. Si deux mesures portent le même nom, elles doivent vouloir dire la même chose. Inversement, si deux mesures ne veulent pas dire la même chose, elles doivent avoir des noms différents. La cohérence implique une qualité des données élevée. Elle suppose que l'on a tenu compte de toutes les données, qu'elles sont complètes.
- L'entrepôt de données doit être adaptable et résistant aux changements. Les besoins des utilisateurs, les conditions d'activité, les données et la technologie sont en perpétuelle évolution. Ces modifications doivent être prises en compte par l'entrepôt de données, sans remettre en cause les données existantes. Elles ne doivent pas invalider les données existantes et les applications ne doivent pas être modifiées ou bouleversées lorsque les utilisateurs posent de nouvelles questions ou que de nouvelles données sont adjointes à l'entrepôt. Si les données descriptives de l'entrepôt doivent être modifiées, il faut pouvoir en rendre compte convenablement.
- L'entrepôt de données doit être protégé. Il contient de précieuses informations sur l'entreprise, ce qu'elle vend, à qui, à quel prix, quelles sont ses interrogations, etc. Il doit donc posséder un contrôle d'accès rigoureux aux informations confidentielles de l'organisation.
- L'entrepôt de données sert de socle à la prise de décision. Il doit contenir des données servant à étayer ces décisions.
- L'entrepôt de données doit être accepté par la communauté des utilisateurs.

4. Qu'est-ce qu'un entrepôt de données ?

4.1. Un entrepôt de données est une base de données

- Consolidant les données de bases de données opérationnelles,
- Utilisée en consultation et mise à jour périodiquement,
- Organisée pour permettre le traitement de requêtes "analytiques" plutôt que "transactionnelles" (OLAP par rapport à OLTP).
  - OLTP: On-line transaction processing. Petites transactions consistant en une recherche d'informations et, souvent, une mise à jour.
  - OLAP: On-line analytical processing. Grosses transactions impliquant une fraction importante des données réalisant, par exemple, un calcul statistique.

4.2. Intérêt de l’entrepôt de données

- Vision transversale de l’entreprise
- Intégration de différentes bases
- Données non volatiles (pas de suppression)
- Historisation
- Organisation vers prise de décision

4.3. Un projet complexe

- Rassembler des données hétérogènes
- Les homogénéiser et les restructurer
- Vérifier leur fiabilité
- Les éditer (publier)

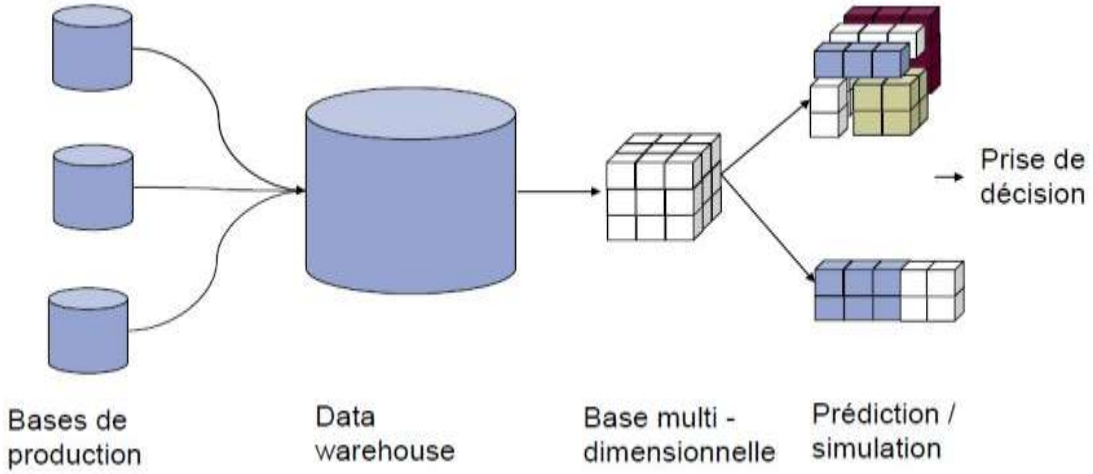


Fig. 2. Processus de prise de Décision

4.4. Exemples

a) Utilisation d'un entrepôt par une entreprise de distribution.

- Les données de vente sont enregistrées dans les différents magasins (OLTP).
- Chaque nuit, les données des différents magasins sont transférées dans un entrepôt de données au siège de la firme.
- Les données de l'entrepôt sont utilisées pour mettre au point des stratégies commerciales, des campagnes de promotion . . .

b) Marketing et gestion de la relation client (CRM)

- Qui sont mes clients?
- Pourquoi sont-ils mes clients?
- Comment les conserver ou les faire revenir?
- Ces clients sont-ils intéressants pour moi ?

c) Analyse des ventes dans les grandes surfaces utilisateurs de la partie décisionnelle du SI

d) Marketing, Actions Commerciales

- Où placer ce produit dans les rayons?
- Comment cibler plus précisément le mailing (communication publicitaire) concernant ce produit ?

4.5. Exemples de requêtes OLAP

a) Quel est le nombre de paires de chaussures vendues par le magasin X en mai 2003

b) Comparer les ventes avec le même mois de 2001 et 2002.

c) Quelles sont les composantes des machines de production ayant eu le plus grand nombre d'incidents imprévisibles au cours de la période 1992-97 ?

4.6. Objectifs

Toutes les données qu'elles proviennent du système de production de l'entreprise ou qu'elles soient achetées vont devoir être organisées, coordonnées, intégrées et stockées, pour donner à l'utilisateur une vue intégrée et orientée métier.

L'objectif d'un Data Warehouse est une sorte de point focal stockant en un endroit unique toute l'information utile provenant des systèmes de production et des sources externes.

Avant d'être chargée dans le Data Warehouse, l'information doit être extraite, nettoyée et préparée. Puis, elle est intégrée et mise en forme de manière compréhensible pour être comprise par l'utilisateur.

5. Types de Données dans un Data Warehouse

5.1. Définition

De nombreuses définitions ont été proposées, soit académiques, soit par des éditeurs d'outils, de bases de données ou par des constructeurs, cherchant à orienter ces définitions dans un sens mettant en valeur leur produit. La définition la plus consensuelle est celle de Bill Inmon, père fondateur du Data Warehouse, en 1990:

"*Un entrepôt de données (data Warehouse) est une collection de données thématiques, intégrées, non volatiles et historisées pour la prise de décisions*".

5.2. Les données sont thématiques

La vocation du Data Warehouse est de prendre des décisions autour des activités majeures de l'entreprise. Les données sont ainsi structurées autour de thèmes, ce qui facilite l'analyse transversale. Pour éviter le doublonnage des données, on regroupe les différents sujets dans une structure commune. Ainsi, si le sujet client contient des informations dans les sujets marketing, ventes, analyse financière, on regroupera ces trois sujets au sein du thème client. Dès lors, chaque donnée n'est présente qu'à un endroit et le Data Warehouse joue bien un rôle de point focal.

5.3. Les données sont intégrées

Elles proviennent de sources hétérogènes. Avant d'être intégrées dans le Data Warehouse, les données doivent êtres mises en forme et unifiées afin d'avoir un état cohérent. Par exemple, la consolidation de l'ensemble des informations concernant un client donné est nécessaire pour donner une vue homogène de ce client. Une donnée doit avoir une description et un codage unique (cohérence, normalisation, maîtrise de la sémantique, prise en compte des contraintes référentielles et des règles de gestion).

5.4. Les données sont historisées et non volatiles

- **Données historisées**
  Suivre dans le temps l'évolution des différentes valeurs des indicateurs → couches de données

- **Données non volatiles (**Traçabilité → non suppression)

  La non volatilité des données est en quelque sorte une conséquence de l'historisation. Une même requête effectuée à quelques mois d'intervalle en précisant la date de référence de l'information recherchée donnera le même résultat.

  Le Tableau *Tab2* présente les principales différences entre le système de production et le data Warehouse.

Critère	Système de production	Datawarehouse
Niveau de détail des informations utilisateurs	Très détaillé	Synthétique, parfois détaillé
Utilisateurs	Une ou quelques fonctions de l'entreprise	plusieurs fonctions de l'entreprise
Données figées	Non-évolution en temps réel	Oui-archivage
Historique	Non	Oui
Opérations sur les données	Mise à jour/consultation	Consultation uniquement

Tab. 2. Différences entre Système de Production et Data Warehouse

6. Les concepts de base

6.1 La structure

Un Data Warehouse se structure en quatre classes de données, organisées selon un axe historique et un axe synthétique (*Fig3*).

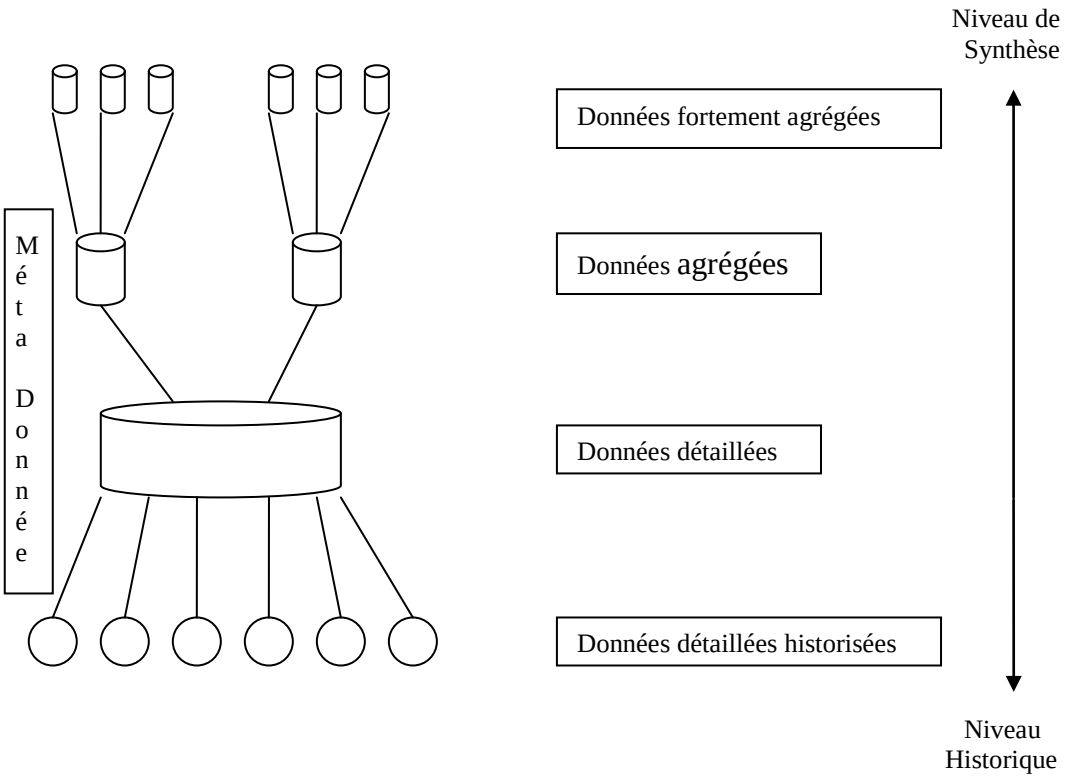


Fig. 3. Structure d'un Data Warehouse

- **Les Données Détaillées**

  Elles reflètent les événements les plus récents. Les intégrations régulières des données issues des systèmes de production vont habituellement être réalisées à ce niveau.

  Les volumes à traiter sont plus importants que ceux gérés en transactionnel. Attention: le niveau de détails géré dans le Data Warehouse n'est pas forcément identique au niveau de détails géré dans les systèmes opérationnels. La donnée insérée dans le Data Warehouse peut être déjà une agrégation ou une simplification d'informations tirées du système de production.

  Exemple: l'étude du panier de la ménagère nécessite de stocker le niveau de finesse du ticket de caisse.

- **Les Données Agrégées**

  Elles correspondent à des éléments d'analyse représentatifs des besoins utilisateurs. Elles constituent déjà un résultat d'analyse et une synthèse de l'information contenue dans le système décisionnel, et doivent être facilement accessibles et compréhensibles. La facilité d'accès est apportée par des structures multidimensionnelles qui permettent aux utilisateurs de naviguer dans les données suivant une logique intuitive, avec des performances optimales. (Certains SGBD du marché sont conçus pour faciliter la mise en place des agrégations et la navigation au sein de celles-ci).

La définition complète de l'information doit être mise à la disposition de l'utilisateur pour une bonne compréhension. Dans le cas d'un agrégat, l'information est composée du contenu présenté (moyenne des ventes, …) et de l'unité (par mois, par produit,…).

- **Les Métadonnées**

  Elles regroupent l'ensemble des informations concernant le Datawarehouse et les processus associés. Elles constituent une véritable aide en ligne permettant de connaître l'information contenue dans le Datawarehouse. Elles sont idéalement intégrées dans un référentiel.

  Les principales informations sont destinées :

  - A l'utilisateur (sémantique, localisation).
  - Aux équipes responsables des processus de transformation des données du système de production vers le Datawarehouse (localisation dans les systèmes de production, description des règles, processus de transformation).
  - Aux équipes responsables des processus de création des données agrégées à partie des données détaillées.
  - Aux équipes d'administration de la base de données (structure de la base implémentant le Datawarehouse).
  - Aux équipes de production (procédures de changement, historique de mise à jour,…)

- **Les Données Historisées**

  Un des objectifs du Datawarehouse est de conserver en ligne les données historisées. Chaque nouvelle insertion de données provenant du système de production ne détruit pas les anciennes valeurs, mais crée un nouvelle occurrence de la donnée. Le support de stockage dépend du volume des données, de la fréquence d'accès, du type d'accès. Les supports les plus couramment utilisés sont les disques, les disques optiques numérique, les cassettes.

  La logique d'accès aux données la plus utilisée est la suivante : les utilisateurs commencent à attaquer les données par le niveau le plus agrégé, puis approfondissent leur recherche vers les données les plus détaillées (Drill Drown).

  L'accès des données se fait également directement par les données détaillées et historisées, ce qui conduit à des brassages de données lourds, demandant des machines très puissantes.

  Le Datawarehouse est une réussite dans une entreprise lorsque le nombre d'utilisateur accédant aux données de détail augmente.

6.2 Les Architectures

Pour implémenter un Data Warehouse, trois types d'architectures sont possibles :

- L'architecture réelle,
- L'architecture virtuelle,
- L'architecture remote.

a) L'architecture réelle

Elle est généralement retenue pour les systèmes décisionnels. Le stockage des données est réalisé dans un SGBD séparé du système de production. Le SGBD est alimenté par des extractions périodiques. Avant le chargement, les données subissent d'importants processus d'intégration, de nettoyage, de transformation (*Fig4*).

L'avantage est le fait de disposer de données **préparées** pour les besoins de la décision et répondant aux objectifs du Datawarehouse.

L'inconvénient est d'une part le coût de stockage supplémentaire et d'autre part le temps d'accès en réel.

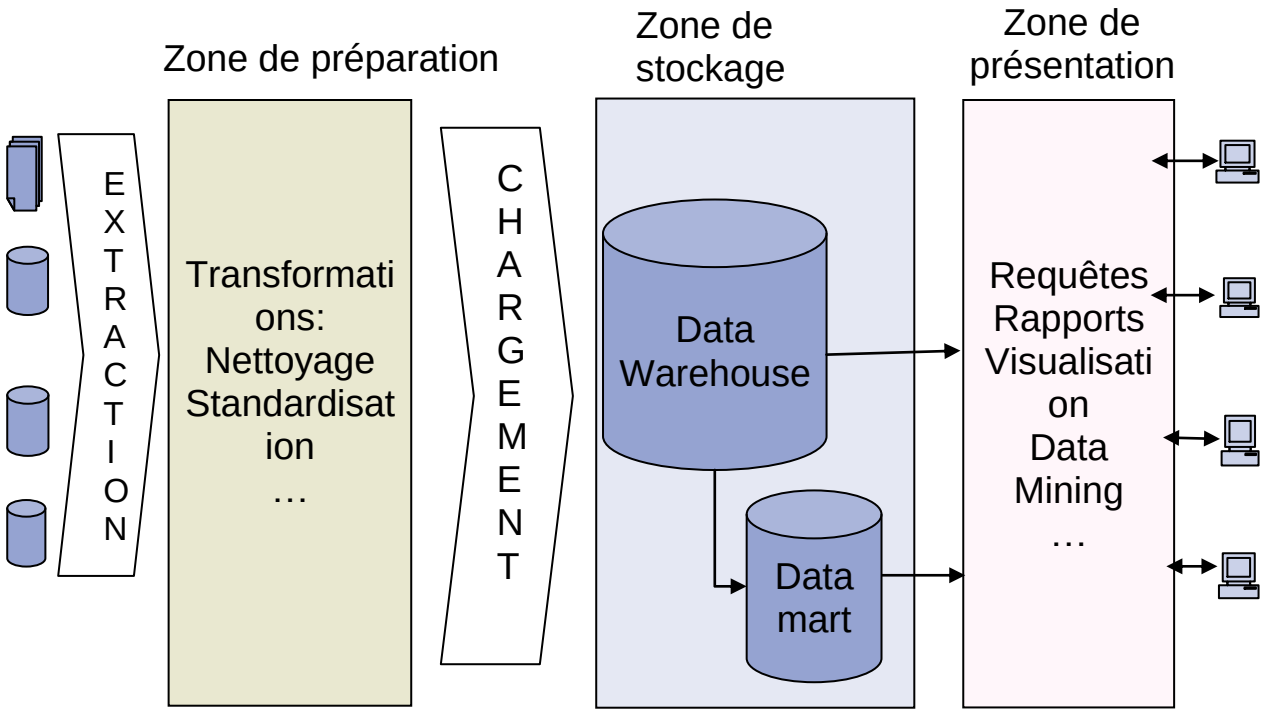


Fig. 4. Architecture d'un Data Warehouse

- **Les Flux de Données**
  - Flux entrant
  - Extraction: multi-source, hétérogène
  - Transformation: filtrer, trier, homogénéiser, nettoyer
  - Chargement: insertion des données dans l'entrepôt
  - Flux sortant
  - Mise à disposition des données pour les utilisateurs finaux
- **Les Différentes Zones de l'Architecture**
  - Zone de préparation (Staging area)
  - Zone temporaire de stockage des données extraites
    - Réalisation des transformations avant l'insertion dans le DW
    - Nettoyage
    - Normalisation…
    - Données souvent détruites après chargement dans le DW
  - Zone de stockage (DW, DM)
    - On y transfère les données nettoyées

- Contient les données de l'entreprise

– Zone de présentation

- Donne accès aux données contenues dans le DW
- Peut contenir des outils d'analyse programmés: Rapports, Requêtes…

b) L'architecture virtuelle

Cette architecture n'est pratiquement pas utilisée pour le Datawarehouse. Les données résident dans le système de production. Elles sont rendues visibles par des produits middleware ou par des passerelles.

Il en résulte deux avantages : pas de coût de stockage supplémentaire et l'accès se fait en temps réel.

L'inconvénient est que les données ne sont pas préparées.

c) L'architecture REMOTE

C'est une combinaison de l'architecture réelle et de l'architecture virtuelle. Elle est rarement utilisée.

L'objectif est d'implémenter physiquement les niveaux agrégés afin d'en faciliter l'accès et de garder le niveau de détail dans le système de production en y donnant l'accès par le biais de middleware ou de passerelle.

d) Synthèse

Les différents éléments d'appréciation sont repris dans le tableau récapitulatif *Tab. 3*.

	Architecture réelle	Architecture virtuelle	Architecture remote
Utilisation	Retenue pour les systèmes décisionnels	Rarement utilisée	Rarement utilisée
Stockage	SGBD séparé du système de production, alimenté par es extractions périodiques	Données résidant dans le système de production	Combinaison des architectures réelles et virtuelles
Avantages	Données préparées pour les besoins de la décision	Pas de coût de stockage supplémentaire, accès en temps réel	
Inconvénients	Coût de stockage supplémentaire, manque d'accès temps réel	Données non préparées	

Tab. 3. Tableau de Synthèse des Architectures de Data Warehouse

7. Les Magasins de Données (Datamarts)

Le Datamart minimise la complexité informatique. Il est donc plus facile de se concentrer sur les besoins utilisateurs.

Définition

Le Datamart est une base de données moins coûteuse que le Datawarehouse, et plus légère puisque destinée à quelques utilisateurs d'un département. Il séduit plus que le data Warehouse les candidats au décisionnel.

C'est une petite structure très ciblée et pilotée par les besoins utilisateurs. Il a la même vocation que le Data Warehouse (fournir une architecture décisionnelle), mais vise une problématique précise avec un nombre d'utilisateurs plus restreint.

En général, c'est une petite base de données (SQL ou multidimensionnelle) avec quelques outils, et alimentée par un nombre assez restreint de sources de données.

Mais pour réussir, il y a quelques précautions à prendre, gage de son évolutivité vers le Data Warehouse.

Donc le Datamart peut préparer au Datawarehouse. Mais il faut penser grand, avenir, et adopter des technologies capables d'évoluer.

En résumé un Datamart est :

- Un sous-ensemble d'un entrepôt de données
- Destiné à répondre aux besoins d'un secteur ou d'une fonction particulière de l'entreprise
- Point de vue spécifique selon des critères métiers

8. Histoire

Les principales dates à retenir construisant l'histoire de l'Entrepôt de données sont les suivantes :

- Années 1960 - General Mills et l'Université Dartmouth, dans un projet conjoint, créent les termes "faits" et "dimensions".
- 1983 - Teradata introduit dans sa base de données managériale un système exclusivement destiné à la prise de décision.
- 1988 - Barry Devlin et Paul Murphy publient l'article "Une architecture pour les systèmes d'information financiers" ("An architecture for a business and information systems") où ils utilisent pour la première fois le terme "Datawarehouse".
- 1990 - Red Brick Systems crée Red Brick Warehouse, un système spécifiquement dédié à la construction de l'Entrepôt de données.
- 1991 - Bill Inmon publie Building the Datawarehouse (Construire l'Entrepôt de Données).
- 1995 - Le Datawarehousing Institute, une organisation à but lucratif destinée à promouvoir le datawarehousing, est fondé.
- 1996 - Ralph Kimball publie The Data Warehouse Toolkit (La boîte à outils de l'Entrepôt de données).

CHAPITRE 2 : Modélisation et Implémentation d'un Entrepôt de Données

1. Modélisation dimensionnelle

Les données d'un Data Warehouse sont de deux types:

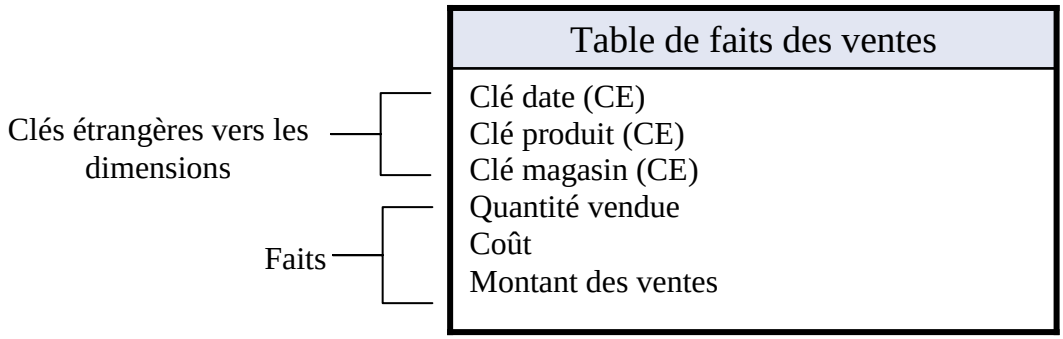
- Les faits: grosse accumulation de données reprenant des faits simples. Exemple: chiffres de ventes,
- Les données "dimensionnelles": données en quantité réduite, souvent statiques qui précisent des informations sur les éléments apparaissant dans les faits.

La modélisation dimensionnelle sert à modéliser l'activité que l'on souhaite analyser.

1.1. Table des faits (clé multiple)

- Table principale du modèle dimensionnel
- Contient les données observables (les faits ou un agrégat de faits) sur le sujet étudié selon divers axes d'analyse (les dimensions), autrement dit elle Contient un ou plusieurs faits numériques qui se produisent pour la combinaison de clés définissant chaque enregistrement

Exemple



Fait: Ce que l'on souhaite mesurer: Quantités vendues, …
Table des faits : Contient les clés des axes d'analyse (dimension) et les faits.

1.2. Typologie des faits

- **Additif**: additionnable suivant toutes les dimensions
  - Quantités vendues, chiffre d'affaire
  - Peut être le résultat d'un calcul:
  - Bénéfice = montant vente - coût
- **Semi additif**: additionnable suivant certaines dimensions
  - Solde d'un compte bancaire:
  - Pas de sens d'additionner sur les dates car cela représente des instantanés d'un niveau
  - Σ sur les comptes: on connaît ce que nous possédons en banque
- **Non additif**: fait non additionnable quelque soit la dimension
  - Prix unitaire: l'addition sur n'importe quelle dimension donne un nombre dépourvu de sens

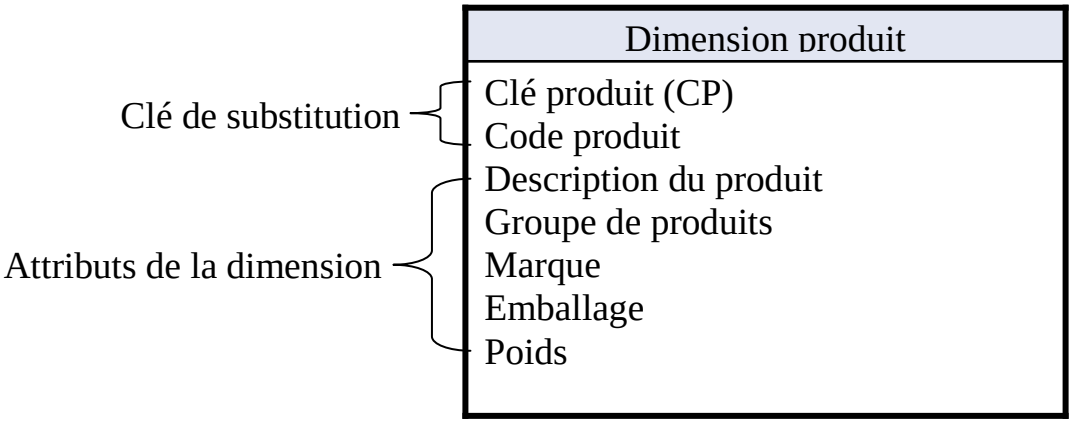
1.3. Table "Dimension" (Dimension = Axe d'analyse)

- Axe d'analyse selon lequel vont être étudiées les données observables (faits)
- Contient les détails sur les faits

- Sa clé est primaire et correspond à l'un des composants de la clé multiple de la table de faits.

Exemple: Client, Produit, Période de temps, …

Une table Dimension contient souvent un grand nombre de colonnes, un ensemble d'informations descriptives des faits et beaucoup moins d'enregistrements qu'une table de faits.



a) Dimensions et Indicateurs

- **Dimensions**
  - Produit
  - Client
  - Vendeur
  - Date
- **Indicateur**
  - Chiffre d'affaires

Une dimension prend une liste de valeurs, un indicateur est un nombre.

b) Hiérarchie de dimensions

Mois ⟶ ***Semaine*** ⟶ ***Jour***

2. Les types de modèles

2.1. Modèle en flocons (snowflake schema)

La représentation directe d'un contexte dimensionnel dans une base de données relationnelle est un réseau de tables jointes selon un schéma en flocon. Dans ce mode de représentation l'association conceptuelle qui contient les faits devient la table de faits, et chacune des entités dimensionnelles devient une table distincte (Fig. 6).

La table de faits contient en plus des indicateurs significatifs qu'elle comporte par définition, un ensemble de clés étrangères, dont chacune assure la liaison avec la table du niveau le plus fin de chaque dimension.

La table des faits est généralement une très grande table, puisqu'elle comporte autant d'enregistrements qu'il existe de combinaisons pertinentes entre les tables "Dimension". Dans le cas de la figure 1, le nombre d'enregistrements de la table de faits "Activité" peut théoriquement être égal au produit du nombre d'Etablissements par le nombre de Produits et par le nombre de Jours de l'historique mémorisé.

C'est une borne maximum, car il n'y a pas nécessairement eu d'activité pour chaque combinaison possible. Même si le nombre d'activité réelle est une faible proportion de ce maximum, la table de faits a pratiquement toujours une taille supérieure d'un ordre de

grandeur à la taille de la plus grande table dimensionnelle. Elle occupe en général 95 à 99 % du volume total de la base de données.

La génération de clés techniques est impérative. Pour être logiquement connectée, une table de faits doit posséder une clé pour chaque dimension. Dans chaque enregistrement d'une table de faits, les clés prennent une place importante. Si la table de faits possède des centaines de milliers, voire de millions, d'enregistrements, l'espace occupé par les clés dans la base de données est loin d'être négligeable. Il faut donc chercher à minimiser cet espace.

Il ne faut donc pas chercher à utiliser les clés signifiantes, qui ne sont pas faites pour économiser de la place, mais pour signifier quelque chose. Il faut utiliser les clés techniques numériques, générées éventuellement lors du chargement dans la base de diffusion (ou dans l'entrepôt de données).

Le format des clés doit être homogène, le plus petit possible compte-tenu de la cardinalité de chaque table. Il faut penser également aux possibilités d'extension de la base en volume. (Fig. 5).

En résumé, le modèle en flocons contient:

- Une table de fait et des dimensions décomposées en sous hiérarchies
- Plusieurs niveaux de Tables de Dimension
- La table de dimension de niveau hiérarchique le plus bas est reliée à la table de fait. On dit qu'elle a la granularité la plus fine.

Avantages

- Normalisation des dimensions
- Économie d'espace disque

Inconvénients

- Modèle plus complexe (jointures)
- Requêtes moins performantes

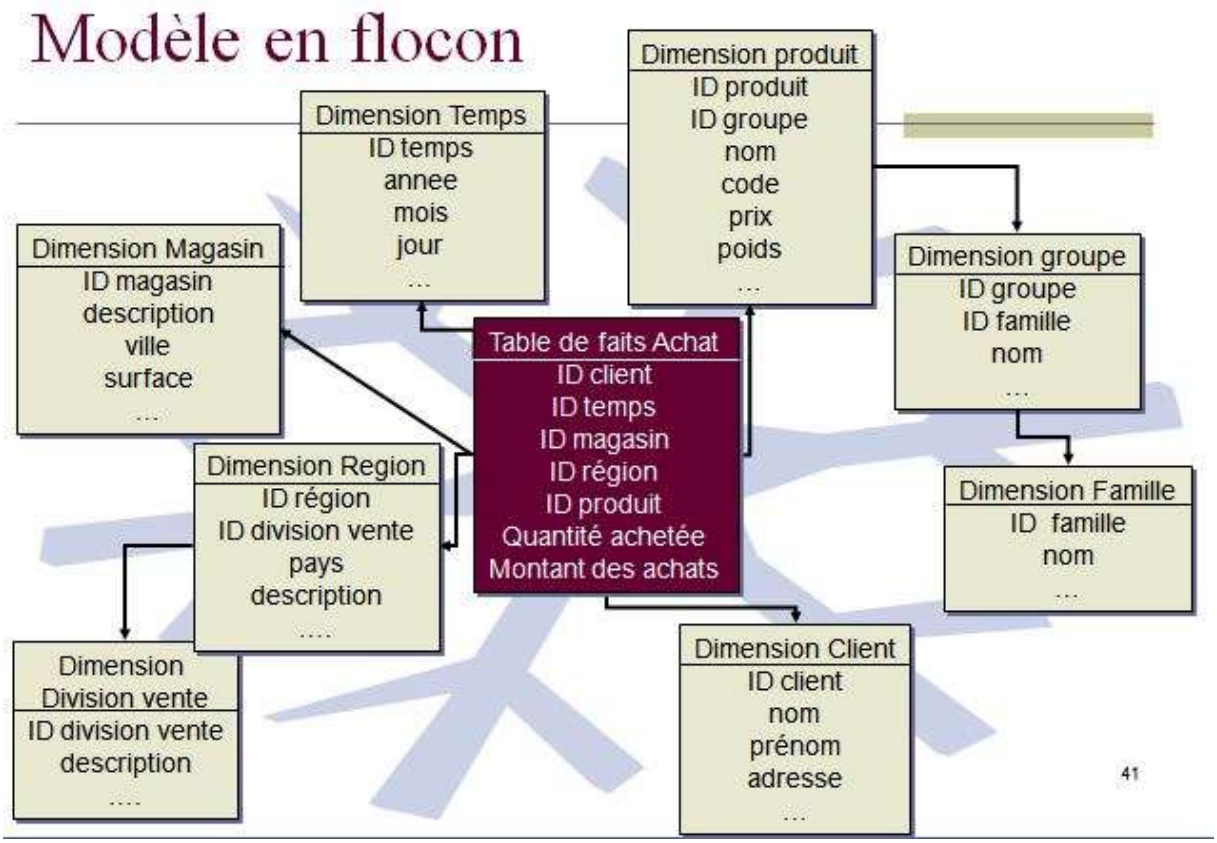


Fig. 5. Exemple de Schéma en flocons

2.2. Modèle en étoile (star schema)

Dans la pratique on préfère une forme dénormalisée du schéma en flocon : le schéma en étoile.

La figure 6 représente le schéma en étoile, dérivé du même modèle que le schéma en flocons de la figure 5.

Le schéma en étoile ne comporte, en plus de la table de faits qu'une table par dimension. Cette simplification est obtenue au prix d'une forte dénormalisation.

Dans la dimension « Client », par exemple, toutes les propriétés descriptives de l'Entreprise et du Groupe sont regroupées dans la même table que les propriétés de l'Etablissement. Dans le cas d'un groupe contrôlant 100 établissements, la description du Groupe sera répétée dans 100 enregistrements.

Ce modèle est donc générateur d'une forte redondance mais:

- La redondance des données ne compromet pas la cohérence d'une base ne subissant pas de mise à jour transactionnelle,
- L'espace occupé par les tables dimensionnelles est insignifiant par rapport au volume de la table de faits,
- Toutes les tables dimensionnelles ont une liaison directe avec la table de faits. Quelle que soit la complexité des dimensions, le nombre de tables pouvant être impliquées dans une requête, en plus de la table de faits, est inférieur ou égal au nombre de dimensions du contexte. Le temps d'exécution d'une requête est indépendant du niveau hiérarchique des propriétés conditionnelles invoquées.

En résumé, le modèle en étoile contient:

- Une table de fait centrale.
- Un ensemble de tables de Dimension (1 niveau).
- Les dimensions n'ont pas de liaison entre elles.

Avantages

- Facilité de navigation
- Nombre de jointures limité

Inconvénients:

- Redondance dans les dimensions
- Toutes les dimensions ne concernent pas les mesures

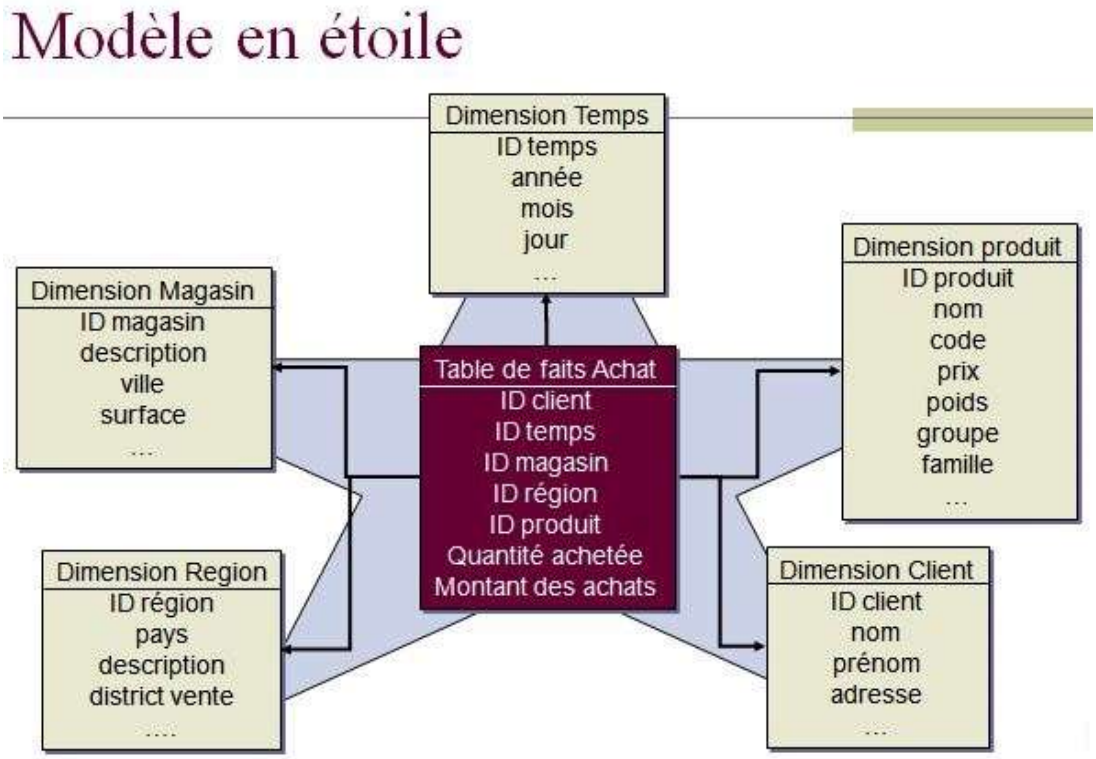
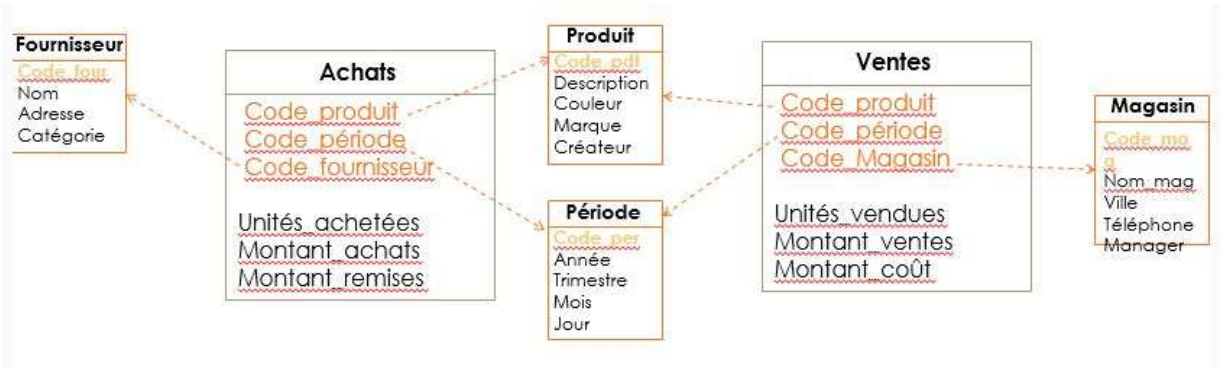


Fig. 6. Exemple de Modèle en Etoile

2.3 Le Modèle en Constellation de faits

Ce modèle est un ensemble de schémas en étoiles et/ou en flocon dans lesquels les tables de faits se partagent certaines tables de dimensions. C'est de cette accumulation que découle un modèle en constellation.



3. BD Multidimensionnelles et OLAP

3.1. Introduction

La vocation d'un entrepôt de données est l'analyse de données pour l'aide à la décision dans les entreprises. La modélisation multidimensionnelle est la base des entrepôts de données et de l'analyse multidimensionnelle (OLAP). Donc, la modélisation multidimensionnelle est une réponse à un besoin analytique.

Les bases de données relationnelles, modélisées selon les principes classiques de normalisation, s'adaptent très mal à un contexte analytique (OLAP). En analyse, l'utilisateur doit disposer d'un modèle relativement intuitif et capable de stocker le résultat de nombreux calculs d'agrégation.

L'intérêt pour l'analyse de données s'est développé énormément ces dernières années. Les entreprises se sont rendues compte de l'efficacité de la technologie OLAP (On-line Analytical Processing) dans l'analyse et l'exploration des données. Cette technologie est utilisée dans les systèmes d'aide à la décision. Le plus souvent, ces systèmes sont basés sur des techniques d'entreposage de données pour exploiter la grande masse d'informations disponibles dans les entreprises à des fins d'analyse et d'aide à la décision.

La modélisation multidimensionnelle propose donc d'analyser des *indicateurs* numériques (par exemple chiffre d'affaires, nombre d'individus, ratios, etc.) dans un contexte précisé par le croisement de plusieurs *dimensions* d'analyse (par exemple temps, géographie, organisation, produits, ...).

Par exemple, considérons les trois dimensions Temps, Pays et Produits, utilisées pour analyser les ventes. L'indicateur Chiffre d'Affaires sera calculable sur l'ensemble des combinaisons possibles entre ces trois axes. L'ensemble des combinaisons possibles peut être représenté par un cube.

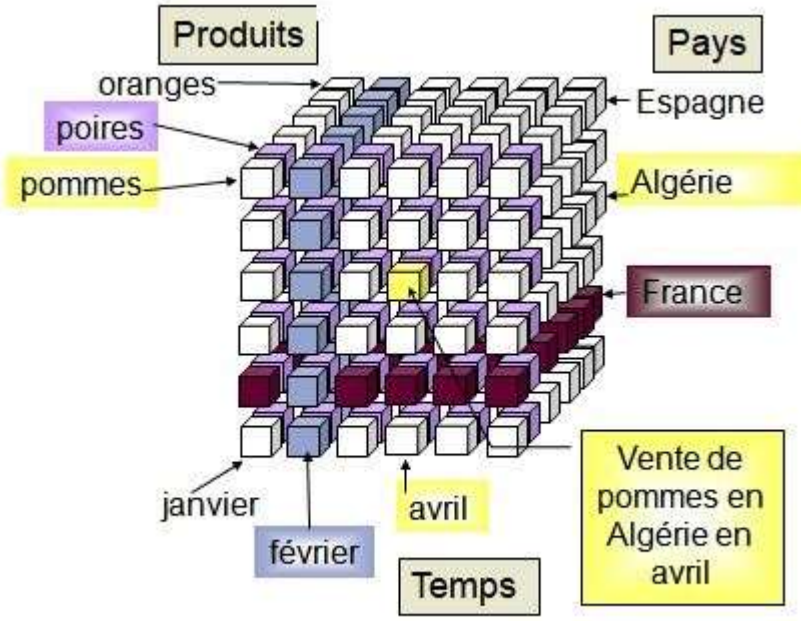


Fig. 7. Exemple de Cube de Données

Au-delà de trois dimensions, cela devient mathématiquement un hypercube (qu'il est beaucoup plus difficile de représenter graphiquement). Une base de données multidimensionnelle typique peut donc s'envisager comme un hypercube d'une dizaine de dimensions comprenant plusieurs millions de cellules (on parle plus couramment de *Cube OLAP*).

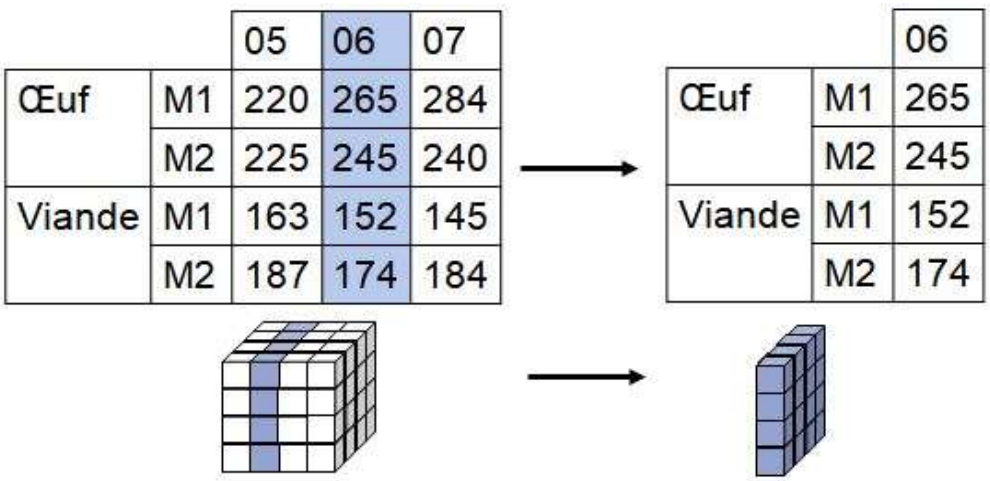
3.2. OLAP (On-Line Analytical Processing)

L'OLAP ou Online Analytical Processing est une technique informatique d'analyse multidimensionnelle, qui permet aux décideurs, d'avoir accès rapidement et de manière interactive à une information pertinente présentée sous des angles divers et multiples, selon leurs besoins particuliers. L'OLAP est donc une technique dont les fonctionnalités servent à faciliter l'analyse multidimensionnelle: opérations réalisables sur l'hypercube pour extraire les données.

a) Opérations OLAP

- **Opérations Agissant sur la Structure**

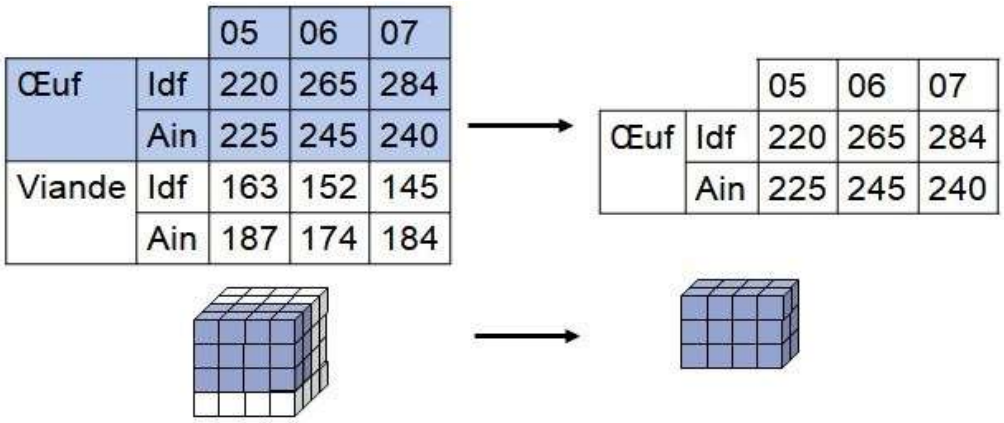
  Tranchage (slicing): Consiste à ne travailler que sur une tranche du cube. Une des dimensions est alors réduite à une seule valeur.



		05	06	07
Œuf	M1	220	265	284
	M2	225	245	240
Viande	M1	163	152	145
	M2	187	174	184

		06
Œuf	M1	265
	M2	245
Viande	M1	152
	M2	174

Extraction d'un bloc de données (dicing): Consiste à travailler uniquement sur un sous-cube.



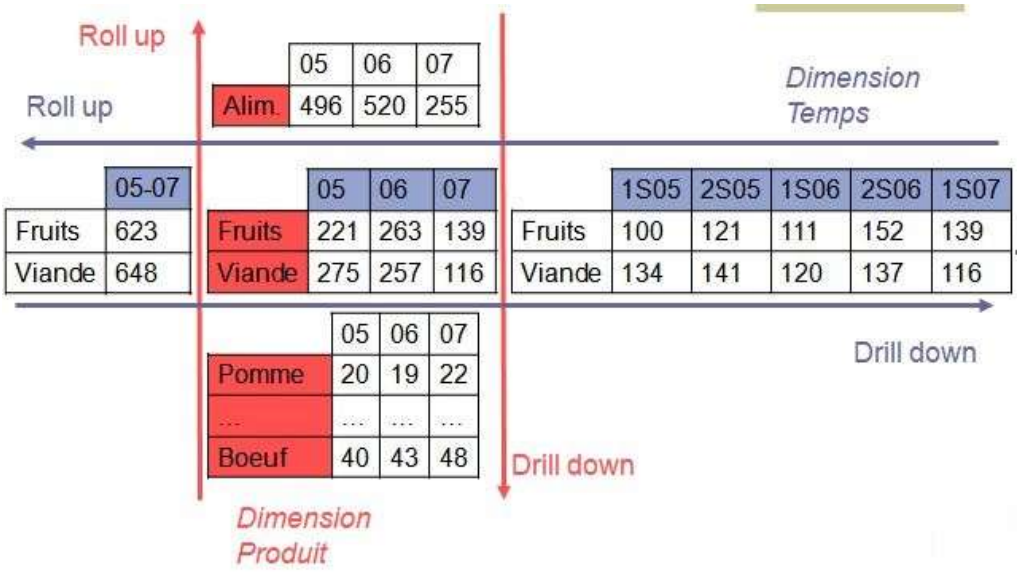
- **Opérations agissant sur la Granularité**

  Forage vers le haut (roll-up): « dézoomer »

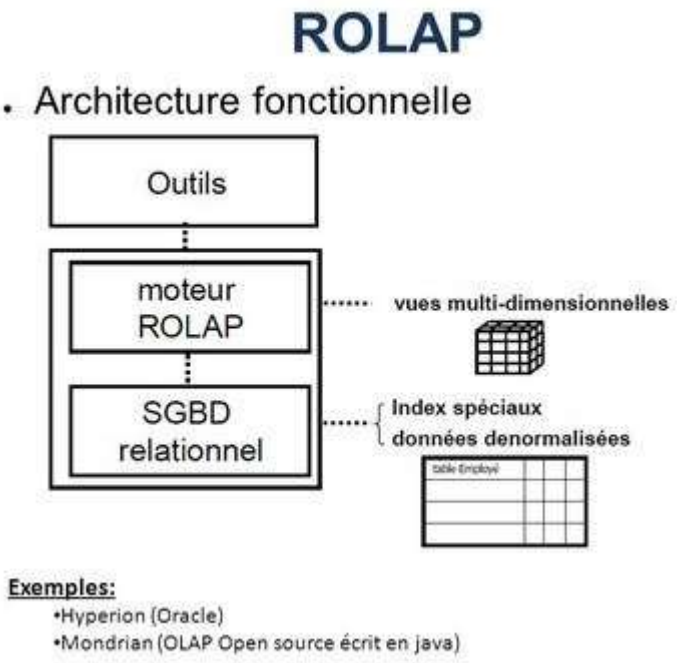
  - Obtenir un niveau de granularité supérieur
  - Utilisation de fonctions d'agrégation

  Forage vers le bas (drill-down): « zoomer »

  - Obtenir un niveau de granularité inférieur
  - Données plus détaillées



- **ROLAP: Relational OLAP**



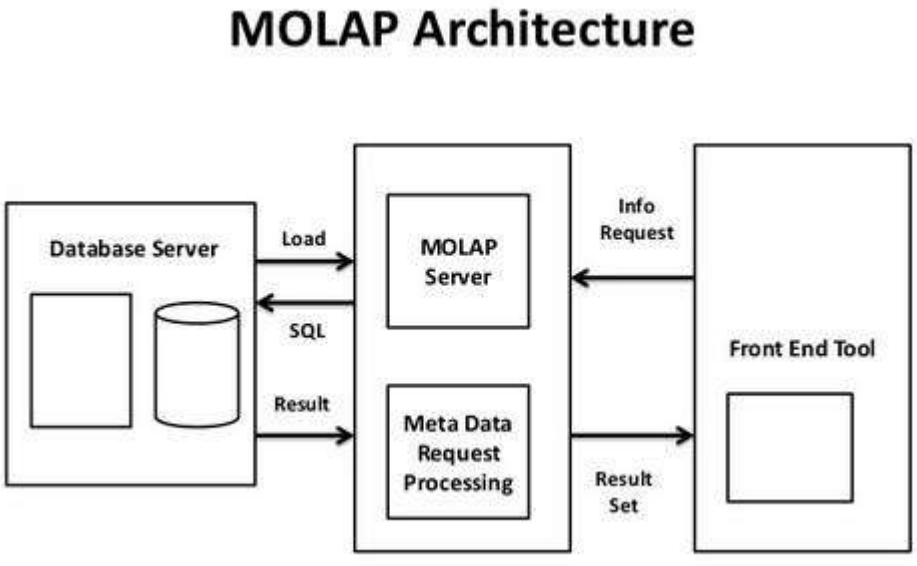
L'obtention des données se fait via des tables relationnelles et des jointures qui vont avec celles-ci. Donc, la requête créée sera relativement complexe, selon la granularité, et, sera d'une longueur plus ou moins importante. Comme le résultat n'est pas stocké, à chaque consultation, la requête devra être relancée à chaque consultation la requête devra être relancée.

Les différents inconvénients de la méthode ROLAP: Le temps de réponse est d'une longueur assez conséquente étant donné que les requêtes fonctionnent via des tables. Les bases sont donc utilisées à chaque relance du rapport.

Les avantages de la méthode ROLAP: Le coût est relativement faible, en effet, cette méthode utilise des ressources déjà existantes comme des ressources matérielles, des licences etc.

Exemples: Microsoft Analysis Services, Oracle 10g, MetaCube d'Informix, Mondrian de Pentaho, DSS Agent de MicroStrategy.

- **MOLAP: Multidimentional OLAP**



On stock les données dans un CUBE qui est en fait une base de données multidimensionnelles. De cette façon, le concept de relationnel n'est plus présent. Pré calculer tous les croisements envisageables est l'objectif de cette base de données multidimensionnelle, de cette manière la restitution des données se fait de façon instantanée. Les données étant stockées, le temps gagné pendant la restitution des données sera considérable.

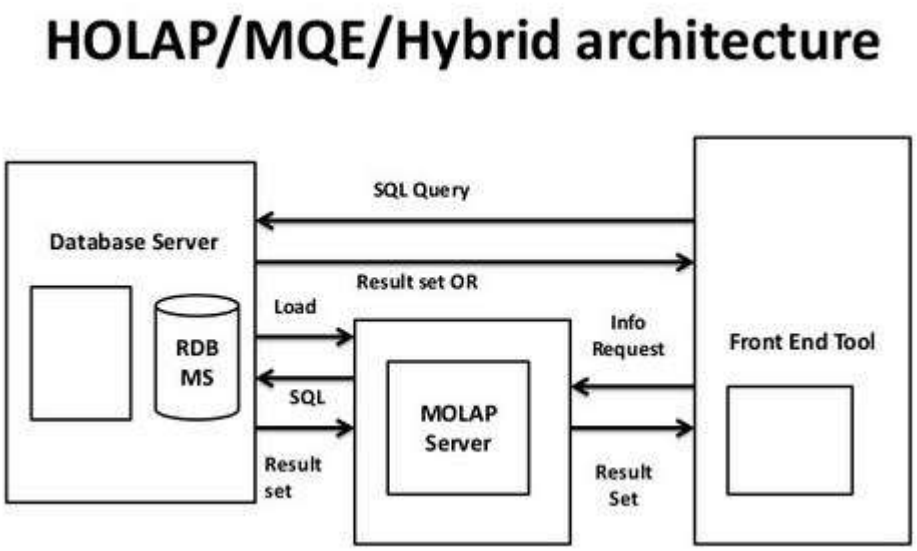
Inconvénients des cubes MOLAP : Le coût est important, en effet, elle nécessite souvent des licences pour les bases multidimensionnelles et des coûts pour le développement des CUBES.

Avantage des cubes MOLAP: Le temps de réponse est extrêmement court car la totalité des données sont stockées au sein d'un CUBE.

Remarque: Les bases de données multidimensionnelles possèdent leur propre langage permettant de faire des requêtes, appelé le MDX, qui est l'équivalent du SQL utilisé pour les bases de données relationnelles.

Exemples: Board M.I.T., Essbase, IBM TM1, Jedox Palo, icCube server, Infor Alea, Microsoft Analysis Services, Oracle OLAP.

- **HOLAP: Hybrid OLAP**



L'HOLAP est un mélange du ROLAP et du MOLAP. Les cubes HOLAP sont donc Hybrides. On se sert du MOLAP lorsque l'on veut accéder aux données agrégées. Si l'on souhaite arriver à un niveau de détail plus important, nous utilisons le ROLAP.

Par exemple, les données sont stockées et accessible via un Cube multidimensionnel, mais on fait également de la restitution via un outil de reporting comme SSRS par exemple. L'utilisateur pourra donc avoir accès à un rapport contenant les données issues du CUBE ainsi qu'à un autre rapport détaillé contenant les données en provenance de tables, cette fois relationnelles.

Inconvénients de la méthode HOLAP: Elle est inutilisable en cas de complexité trop élevée des rapports ou qu'ils fassent appel à trop de croisements de données. **Avantages de la méthode HOLAP**: Un investissement financier moindre que la méthode MOLAP, en effet la partie développement sera beaucoup moins importante. De plus le temps de réponse est relativement court.

Exemple: Oracle OLAP, Microsoft Analysis Services.

CHAPITRE 3 : Alimentation du Data Warehouse

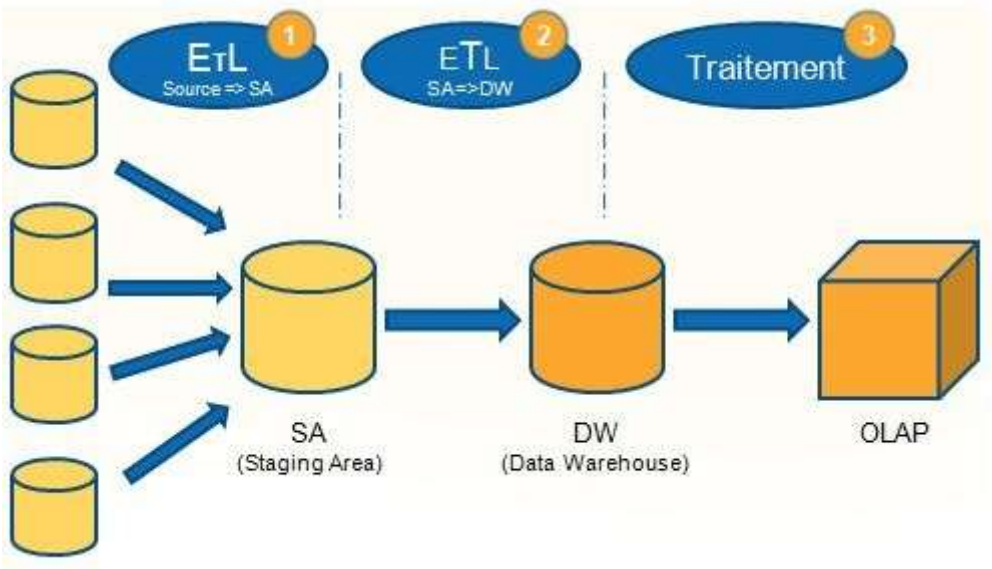
1. Introduction

Après la conception de l'ED, on a besoin d'Acquérir des données pour l'alimentation ou la mise à jour régulière de l'entrepôt.

Besoin d'un outil pour automatiser les chargements de l'entrepôt : ETL

ETL, acronyme de *Extraction, Transformation, Loading* (ou : *data pumping*), en français (Extraction, Transformation et Chargement) est un système de chargement de données depuis les différentes sources d'information de l'entreprise (hétérogènes) jusqu'à l'entrepôt de données.

Il est important de savoir que la réalisation de l'ETL constitue 70% d'un projet décisionnel en moyenne. Et ce n'est pas pour rien, ce système est complexe et ne doit rien laisser s'échapper, sous peine d'avoir une mauvaise information dans l'entrepôt, donc des données fausses, donc inutilisables.



Les sources de données peuvent être de plusieurs types:

- **Enterprise ressource planning (ERP)**

  Gèrent les processus opérationnels d'une entreprise (ex: ressources humaines, finances, distribution, approvisionnement, etc.).

- **Customer Relationship Management (CRM)**

  Gèrent les interactions d'une entreprise avec ses clients (ex: marketing, ventes, après-vente, assistance technique, etc.).

- **Systèmes Legacy (systèmes légataires)**

  Matériels et logiciels dépassés mais difficilement remplaçables.

- **Point of sale (POS) (point de vente)**

  Matériels et logiciels utilisés dans les caisses de sorties d'un magasin.
- **Externes**

  Données concurrentielles achetées, données démographiques.

  Les sources de données peuvent présenter des problèmes tels que:

- Sources diverses et hétérogènes;
- Sources sur différentes plateformes et OS;
- Applications *legacy* utilisant des BD et autres technologies obsolètes;
- Historique de changement non-préservé dans les sources;
- Qualité de données douteuse et changeante dans le temps;
- Structure des systèmes sources changeante dans le temps;
- Incohérence entre les différentes sources;
- Données dans un format difficilement interprétable ou ambigu.

2. Définitions

Les processus ETL sont les composants les plus critiques et les plus importants d'une infrastructure décisionnelle. Bien que cachés de l'utilisateur de la plate-forme décisionnelle, les processus ETL rassemblent les données à partir des systèmes opérationnels et les prétraitent pour les outils d'analyse et de reporting. La précision et la vitesse de la plate-forme décisionnelle toute entière dépendent des processus ETL

Ils regroupent plusieurs étapes, qui ont pour objet de transférer des données depuis les applications de production vers les systèmes décisionnels :

L'alimentation se déroule en 3 phases. Ces trois étapes décrivent une mécanique cyclique qui a pour but de garantir l'alimentation du Datawarehouse en données homogènes, propres et fiables.

2.1. Extraction des données

But: Extraction de données des applications et des bases de données de production (ERP, CRM, SGBDR, fichiers, ...)

L'extraction est la première étape du processus d'apport de données à l'entrepôt de données. Extraire, cela veut dire lire et interpréter les données sources et les copier dans la zone de préparation en vue de manipulations ultérieures.

Elle consiste en :

- Cibler les données,
- Appliquer les filtres nécessaires,
- Définir la fréquence de chargement,

Lors du chargement des données, il faut extraire les nouvelles données ainsi que les changements intervenus sur ces données. Pour cela, il existe trois stratégies de capture de changement :

- **Colonnes d'audit**: la colonne d'audit, est une colonne qui enregistre la date d'insertion ou du dernier changement d'un enregistrement. Cette colonne est mise à jour soit par des triggers ou par les applications opérationnelles, d'où la nécessité de vérifier leur fiabilité.
- **Capture des logs**: certains outils ETL utilisent les fichiers logs des systèmes sources afin de détecter les changements (généralement logs du SGBD). En plus de l'absence de cette fonctionnalité sur certains outils ETL du marché, l'effacement des fichiers logs engendre la perte de toute information relative aux transactions.
- **Comparaison avec le dernier chargement**: le processus d'extraction sauvegarde des copies des chargements antérieurs, de manière à procéder à une comparaison lors de

chaque nouvelle extraction. Il est impossible de rater un nouvel enregistrement avec cette méthode. L'extraction des données des sources hétérogènes nécessite d'identifier les sources utiles et de comprendre les schémas.

2.2. Transformation des données

But: Rendre cohérentes les données issues de différentes sources C'est une suite d'opérations qui a pour but de rendre les données cibles homogènes pour être traitées de façon cohérente.

Cette opération se solde par la production d'informations dignes d'intérêt pour l'entreprise et les données sont donc prêtes à être entreposées.
La transformation de ces données vise à les réconcilier entre les différentes sources, pour effectuer des calculs ou du découpage de texte, pour les enrichir avec des données externes et aussi pour respecter le format requis par les système cibles (Troisième Forme Normale, Schéma en Etoile, etc.)

Avant de commencer, il faut visualiser le schéma d'un entrepôt et sa façon de fonctionner (gérer l'historique, dimensions, faits, etc.). Le but du jeu est de faire rentrer toutes les données de l'entreprise dans ce modèle, les données doivent donc être :

- **Dé-normalisées**: Dans un DW (datawarehouse), avoir des doublons n'est pas important; avoir un schéma en troisième forme normale est même déconseillé. Il faut que les données apparaissent là où elles doivent apparaître.
- **Nettoyées**: Dans un système de production, les utilisateurs introduisent les données. Les risques d'erreurs sont donc élevés. Ces erreurs ont des répercutions directes sur les analyses. Il faut pouvoir détecter et corriger ces erreurs.
- **Contextualisées**: Imaginez un système de production, où les informations sur l'activité du personnel sont enregistrées, et un système de RH où les informations personnelles des employés sont stockées. Un entrepôt de données possède une vision universelle, les informations relatives à un employé ne seront stockées qu'une seule dans une seule dimension.

La transformation consiste donc en:

- Une détection et correction d'erreurs,
- Une discrétisation, réduction, normalisation,
- Une détection de redondance, fusion, intégration,
- Une transformation dans le modèle cible.

Exemple

Donnés sources	**données cibles (après intégration)**
Source 1 : male, femelle	**m, f**
Source 2 : 1, 0	**m, f**
Source 3 : Masculin, féminin	**m, f**

Donnés sources	**données cibles (après intégration)**
Source 1 : $150,000	**1 050 000 DA**
Source 2 : 16 000 €	**1 600 000 DA**
Source 3 : 200.000 RS	**4 000 000 DA**

2.3 Chargement

But: Introduire les données dans l'entrepôt.

C'est la dernière phase de l'alimentation d'un entrepôt de données, le chargement est une étape indispensable. Elle reste toutefois très délicate et exige une certaine connaissance des structures du système de gestion de la base de données (tables et index) afin d'optimiser au mieux le processus.

Le chargement est l'étape la plus complexe, il s'agit ici d'ajouter les nouvelles lignes, voir si des lignes ont été modifiées et faire une gestion d'historique, voir si des lignes ont été supprimées et le mentionner dans l'entrepôt, tout en faisant attention de ne pas charger des données en double.

La latence des processus d'ETL varie, du mode batch (traitement effectué par lot) (parfois mensuel ou hebdomadaire, le plus souvent quotidien) jusqu'au quasi-temps réel avec des rafraîchissements plus fréquents (toutes les heures, toutes les minutes, etc.).

3. Conception d'un ETL

Il n'existe pas de méthodes de conception d'ETL. Chaque entreprise possède ses propres systèmes, sa propre logique de fonctionnement et sa propre culture. Un ETL va prendre toutes les données de l'entreprise et les charger dans un DW.

3.1. Comment procéder?

Deux cas sont à prendre en compte, le chargement initial et les chargements incrémentiels.

Le chargement initial est effectué au tout premier chargement de l'entrepôt et dans des cas spéciaux comme après la perte des données de l'entrepôt. Dans ce cas, on charge toutes les données de l'entreprise dans l'entrepôt.

Le chargement incrémentiel est le fait d'ajouter des données à un entrepôt existant, c'est l'opération qui va se répéter dans le temps (chaque jour par exemple). Il faudra faire attention dans ce cas à ne charger que les informations nouvelles, et ne pas charger deux fois la même information.

3.2. Comment sont mes sources ?

Avant de faire un ETL, il faut bien étudier les sources de données. En effet, c'est d'après les sources que les stratégies de chargement vont se faire.

3.2.1 Politiques de l'alimentation

Le processus de l'alimentation (rapatriement des données) peut se faire de différentes manières. Le choix de la politique de chargement dépend des disponibilités et des accessibilités des sources de données. Ces politiques sont:

- **Push:** dans cette méthode, la logique de chargement est dans le système de production. Il "pousse" les données vers le Staging quand il en a l'occasion. L'inconvénient est que si le système est occupé, il ne poussera jamais les données.
- **Pull:** au contraire de la méthode précédente, le Pull " tire " les données de la source vers le Staging. L'inconvénient de cette méthode est qu'elle peut surcharger le système s'il est en cours d'utilisation.
- **Push-pull:** vous le devinez, c'est le mélange des deux méthodes. La source prépare les données à envoyer et prévient le Staging qu'elle est prête. Le Staging va récupérer les données. Si la source est occupée, le Staging fera une autre demande plus tard.

Staging (ou zone de préparation) est le terme désignant l'endroit où se fait l'ETL. C'est une machine dédiée à ce travail dans la plupart des cas. Considérez le *Staging* comme une zone tampon entre les sources de données et l'entrepôt.

Une fois la bonne stratégie choisie (en fonction des spécificités de l'entreprise), il est temps de se poser les questions fondamentales qui dessineront les caractéristiques de votre système:

- Quelle est la disponibilité de mes sources de données ?
- Comment y accéder ?
- Comment faire des chargements incrémentiels ?
- Quel est le temps d'un chargement incrémentiel moyen, ai-je la possibilité de recharger des données dans le cas où mon processus de chargement échoue ?
- Quelle politique vais-je utiliser dans le cas d'échec de chargement ?

Ces questionnements aideront à établir une stratégie de chargement des données sources dans le *Staging*.

3.3. Comment traiter les données ?

Maintenant que les données sont dans le Staging, il va falloir les nettoyer. C'est l'opération la plus importante du processus. En effet, une erreur dans un champ affecte forcément les analyses (exemple de Canada et Canonda). Voici les questions à se poser à cette étape :

- Quels sont les champs les plus sujets à erreurs ?
- Ai-je les moyens de corriger les erreurs automatiquement ?
- Comment permettre à un utilisateur de corriger les erreurs ?
- Quelle politique vais-je utiliser pour le traitement des erreurs.
- Comment montrer à l'utilisateur final que des données n'ont pas été totalement chargées à cause d'erreurs ?

3.4. Comment charger les données dans l'entrepôt ?

La dernière mission de l'ETL, charger les données dans le DW. Le point critique dans cette étape est qu'il faut avoir, à tout moment, un contrôle total sur les processus. Exemple : pendant un projet de construction d'entrepôt, vous commencez à automatiser les chargements incrémentiels. Mais un jour, la machine tombe en panne au beau milieu du chargement, c'est-à-dire qu'une partie des données a été chargée et une autre non. Que faire ??

Et bien si vous n'aviez pas prévu cela, vous n'avez qu'à vider la base et la recharger, avec toutes les pertes d'historique que cela implique, ou sinon prendre le temps et chercher une à une, les informations qui ont été chargées. Voici les questions qu'il faut se poser pour cette étape :

- Que faire si un chargement échoue ?
- Ai-je les moyens de revenir à l'état avant le chargement ?
- Puis-je revenir dans le temps d'un chargement donné ?
- Comment valider mon chargement, comment détecter les erreurs ?

3.5. Les métadonnées

C'est une des clés du succès de tout projet décisionnel. Les méta-données, en informatique décisionnelle, sont des informations décrivant notre environnement décisionnel. Il ne s'agit pas seulement des informations concernant le schéma des entrepôts ou la politique d'attribution de noms aux champs de l'entrepôt, mais de tout ce qui, de près ou de loin, peut ajouter de la compréhension aux chiffres présentés.

En effet, il est peut être pertinent pour notre l'analyste de savoir que la colonne prix qu'il est en train d'analyser provient des archives et non des données courantes. Il est peut être utile aussi de savoir que les chiffres devant nos yeux sont issus d'un chargement qui a échoué mais qu'on a réussi à recharger correctement. Il est important pour le grand patron d'une entreprise d'avoir une petite info bulle qui lui indique que les données de son tableau de bord sont ceux de l'avant-veille car le chargement ne s'est pas bien déroulé. Imaginez la catastrophe si le décideur prenait des décisions sur des données erronées !!

Il est très important, dans un environnement décisionnel, de non seulement documenter tout le projet, mais de rendre aussi disponible toutes ses méta-données aux utilisateurs de l'environnement pour générer encore plus de connaissance. Car n'oubliez pas que le but finalement est de créer de la connaissance.

3.6. Comment contrôler cet ETL ?

Dans tout ce que vous ferez dans la vie, le contrôle est la clé du succès. Le non contrôle amenant inévitablement le risque et le risque entraînant l'erreur. Si vous ne savez pas d'où provient une erreur, il est fort probable qu'elle soit du côté d'un élément dont vous n'avez pas le contrôle. Le meilleur système étant celui qui laisse le moins de place au risque. Intéressons nous à comment contrôler un ETL, quels sont les points clés à surveiller et, surtout, que faire lorsqu'un élément ne fait pas son travail correctement.

Les ETL sont, malheureusement, la plus grande faiblesse des environnements décisionnels. Tout est critique et sujet à contrôle dans un ETL. Sans oublier que le contrôle sans action n'est pas utile !

Comment retourner en arrière, c'est la principale question qu'on se pose et à laquelle on essaie de répondre quand on conçoit un ETL. Comment retourner en arrière si un chargement s'arrête brusquement, comment revenir à l'état initial si les données semblent incohérentes, comment valider mes transformations…

Le but de tous ces questionnements est de préserver l'intégrité et la " vérité " de l'entrepôt de données. Car c'est le seul point de défaillance de l'environnement. (à part la phase de chargement, tout le système est en lecture seule).

4. Outils ETL

4.1. Éléments à prendre en compte lors du choix de l'ETL

Les solutions d'ETL existent, sont nombreuses et répondent à toutes les demandes de performance et de portefeuille.

Cependant, devant un choix si diversifié, on se retrouve un peu perdu : Open Source ou payant, solution intégrée ou indépendante, sous-traitance ou développement. Les éléments à prendre en compte dans le choix de votre ETL sont les suivants :

- **Taille de l'entreprise**: j'entends par là taille des structures. S'il s'agit d'une multinationale avec des milliers de succursales à travers le monde, on ira plus pour une solution complète et, en général, très coûteuse. Si on est une PME, on optera plutôt pour des solutions payantes (comme Microsoft Integration Services) assurant un certain niveau de confort sans impliquer des mois de développement.
- **Taille de la structure informatique**: une entreprise avec une grosse structure informatique pourra se permettre d'opter pour une solution Open Source et la personnaliser selon les besoins de l'entreprise. Une PME ne pourra sûrement pas faire cela.
- **Culture d'entreprise**: évidemment, si une entreprise a une culture de l'Open Source très prononcée, l'application d'une solution payante risquera fortement de subir un phénomène de rejet.

- **Maturité des solutions**: il existe des solutions bien rodées, qui fonctionnent bien et qui bénéficient d'un bon retour d'expérience, c'est en général les plus chères (Business Objects, Oracle, SAP). Il existe d'autres solutions, moins matures, bénéficiant d'un " effet de mode " et qui semble offrir de très bonnes performances (Microsoft). Enfin, il existe des solutions Open Source qui, de part leur jeunesse, n'offrent pas autant de flexibilité et de facilité de mise en œuvre que les solutions précédemment citées. Il faudra compter avec le temps pour que ces solutions émergent et arrivent à un niveau de maturité acceptable…

5. Les Challenges de l'ETL

L'implémentation de processus d'ETL efficaces et fiables comprend de nombreux challenges.

- Les volumes de données sont en croissance exponentielle, et les processus d'ETL doivent traiter des quantités importantes de données granulaires (produits vendus, appels téléphoniques, transactions bancaires, etc.). Certains systèmes décisionnels sont mis à jour de façon incrémentale, alors que d'autres sont rechargés dans leur totalité à chaque itération.
- Alors que les systèmes d'information se complexifient, la variété des sources de données s'accroît également. Les processus d'ETL doivent disposer d'une large palette de connecteurs à des progiciels (ERP, CRM, etc.), bases de données, mainframes, fichiers, Services Web etc.
- Les structures et applications décisionnelles incluent des Data Warehouses, des Datamarts, des applications OLAP - pour l'analyse, le reporting, les tableaux de bord, le scorecarding, etc. Toutes ces structures cibles présentent des besoins différents en termes de transformation de données, ainsi que des latences différentes.

Les transformations des processus d'ETL peuvent être **très complexes**. Les données doivent être agrégées, parsées, calculées, traitées statistiquement, ...

6. Solutions d'Intégration Open Source pour l'ETL

Les solutions d'intégration de données *Talend* sont optimisées pour les besoins ETL de l'entreprise.

Exemple: Talend Open Studio (TOS)

Talend est la solution d'intégration de données Open Source permettant de répondre avec efficacité à un très large éventail de besoins: alimentation de Data Warehouse, synchronisation de bases de données, transformation de fichiers de divers formats (XML, VSAM, délimités, positionnels...), ...

Talend Open Studio sait déjà se connecter à un nombre plutôt impressionnant de bases de données, sait manipuler un grand nombre de formats de données. Cependant, il rester encore à étendre et ce par exemple à vos applications qui doivent gagner en ouverture vers tout autre logiciel.

Exemples de d'Outils ETL

- Oracle Warehouse Builder;
- IBM Infosphere Information Server;
- Microsoft SQL Server Integration Services (SSIS);
- SAS Data Integration Studio.

En résumé, Le secret d'un bon ETL réside dans sa complétude et dans son exhaustivité dans la prise en charge des données depuis les sources de données jusqu'à l'entrepôt.

7. La phase de restitution

C'est la dernière étape d'un projet Data Warehouse, soit son exploitation, soit restitution des résultats. Elle se fait par le biais d'un ensemble d'outils analytiques développés autour du Data Warehouse. Les outils de restitution sont la partie visible offerte aux utilisateurs. Par leur biais,

les analystes sont à même de manipuler les données contenues dans les entrepôts et les marchés de données. Les intérêts de ces outils sont l'édition de rapports et la facilité de manipulation. En effet, la structure entière du système décisionnel est pensée pour fournir les résultats aux requêtes des utilisateurs, dans un temps acceptable et sans connaissance particulière dans le domaine de l'informatique.

On distingue à ce niveau plusieurs types d'outils différents :

- Les outils de reporting et de requêtes
- Les outils d'analyse OLAP
- La phase de Datamining

Les outils de reporting et de requêtes permettent la mise à disposition de rapports périodiques, pré-formatés et paramétrables par les opérationnels. Ils offrent une couche d'abstraction orientée métier pour faciliter la création de rapports par les utilisateurs eux-mêmes en interrogeant le Data Warehouse grâce à des analyses croisées. Ils permettent également la production de tableaux de bord avec des indicateurs de haut niveau pour les managers, synthétisant différents critères de performance.

Le tableau de bord est un ensemble d'indicateurs peu nombreux conçus pour permettre aux gestionnaires de prendre connaissance de l'état et de l'évolution des systèmes qu'ils pilotent et d'identifier les tendances qui les influenceront sur un horizon cohérent avec la nature de leurs fonctions.

Les tableaux de bord sont prédéfinis et consultables à l'écran, ils génèrent des états tels que les histogrammes.

Les outils d'analyse OLAP permettent de traiter des données et de les afficher sous forme de cubes multidimensionnels et de naviguer dans les différentes dimensions. Cet agencement des données permet d'obtenir immédiatement plusieurs représentations d'un même résultat, en une seule requête sous une approche descendante des niveaux agrégés vers les niveaux détaillés (Drill-down, Roll-up).

Les outils de Datamining offrent une analyse plus poussée des données historisées permettant de découvrir des connaissances cachées dans les données comme la détection de corrélations et de tendances, l'établissement de typologies et de segmentations ou encore des prévisions. Le Datamining est basé sur des algorithmes statistiques et mathématiques, et sur des hypothèses métier.

8. Maintenance et expansion

La mise en service du Data Warehouse ne signifie pas la fin du projet, car un projet Data Warehouse nécessite un suivi constant compte tenu des besoins d'optimisation de performance et ou d'expansion. Il est donc nécessaire d'investir dans les domaines suivants:

- Support: Assurer un support aux utilisateurs pour leur faire apprécier l'utilisation de l'entrepôt de données. En outre, la relation directe avec les utilisateurs permet de détecter les correctifs nécessaires à apporter.
- Formation: il est indispensable d'offrir un programme de formation permanant aux utilisateurs de l'entrepôt de données.
- Support technique: un entrepôt de données est considéré comme un environnement de production. Naturellement le support technique doit surveiller avec la plus grande vigilance les performances et les tendances en ce qui concerne la charge du système.
- Management de l'évolution: il faut toujours s'assurer que l'implémentation répond aux besoins de l'entreprise. Les revues systématiques à certain point de contrôle sont un outil clé pour détecter et définir les possibilités d'amélioration. En plus du suivi et de la maintenance du Data Warehouse, des demandes d'expansion sont envisageables pour de nouveaux besoins, de nouvelles données ou pour des améliorations.

Ces travaux d'expansion sont à prévoir de façon à faciliter l'évolution du schéma du Data Warehouse.

9. Exercice

Une entreprise de fabrication de vaisselle jetable souhaite mettre en place un système d'information décisionnel sous la forme d'un Datamart pour observer son activité de ventes au niveau des différents lieux de distributions de ses articles et cela dans plusieurs villes. Ces lieux de distributions sont identifiés par leur enseigne, leur type (en fonction de leur surface), leur adresse (code postal et ville), leur département et leur région. Les informations relatives aux ventes sont: une période en mois, en trimestre et année. Les ventes sont observées par le nombre d'articles selon le type, et le chiffre d'affaire.

1. Quel est le fait à observer?

2. Quels sont les axes d'analyse et les mesures (indicateurs)?

3. Construire le modèle en étoile de ce Datamart.

4. Modifier ce modèle en un modèle en flocon. Modéliser explicitement les hiérarchies des dimensions.

5. Donner une représentation en sommant les éléments selon les hiérarchies représentées en 4.

CHAPITRE 4 : Fouille de Données (Data Mining)

1. Introduction

L'expression "Data Mining" serait apparue pour la première fois dans les années 60. Employée par les statisticiens, elle critiquait les pratiques de recherches de corrélations des données sans hypothèse de départ. Terme péjoratif, il était souvent associé à l'expression "Data Fishing" en opposition aux méthodes scientifiques de la communauté des statisticiens. Obtenant des résultats encourageants et faisant fi des critiques, les chercheurs en base de données ont persisté dans cette voie.

L'expression "Data Mining" réapparaît dans les années 80 quand certains chercheurs, notamment Rakesh Agrawal (chef de projet au centre de recherche IBM d'Almaden), commencent à travailler sur de volumineuses bases de données, persuadés de pouvoir valoriser les informations qu'elles contiennent.

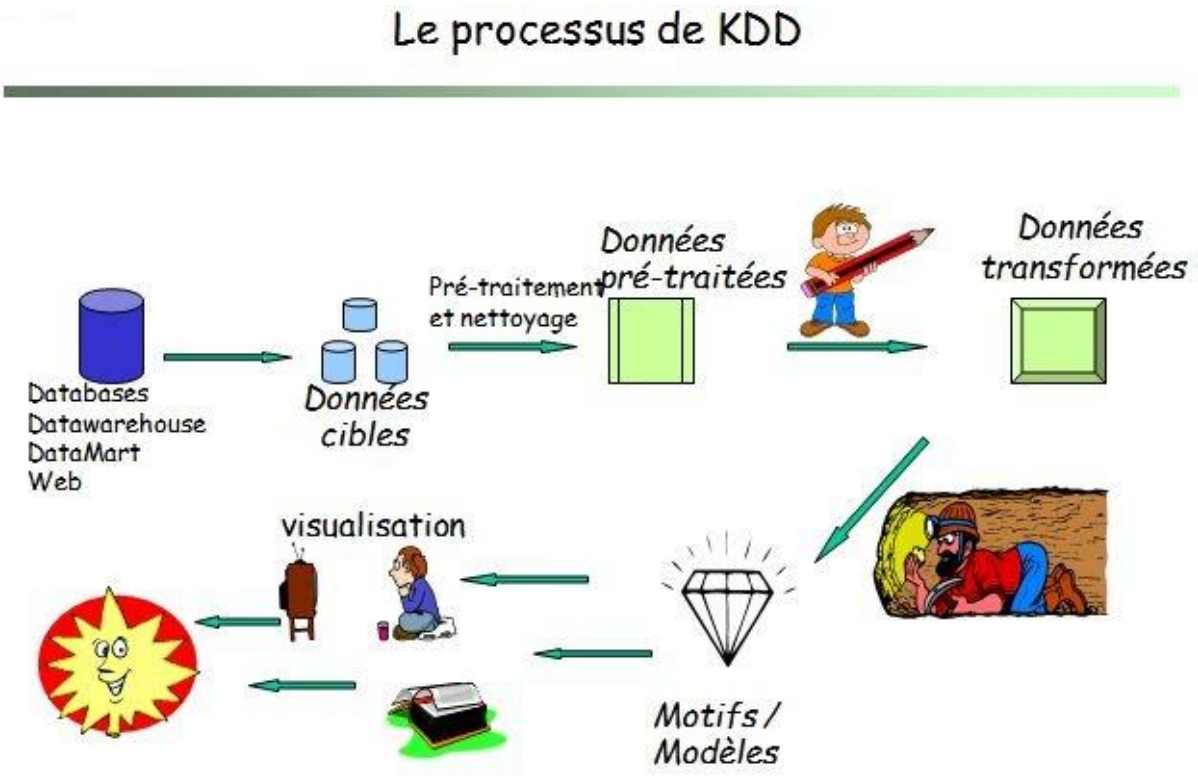
L'apparition du microprocesseur et des bases de données ayant rendu accessible l'informatisation aux moyennes, grandes entreprises et administrations, d'importantes bases de données sont générées. Il est intéressant de noter que les bases de données volumineuses des années 80 "pesaient" 1 Mb (millions de bytes) alors qu'aujourd'hui, elles font plusieurs terabytes (mille milliards de bytes).

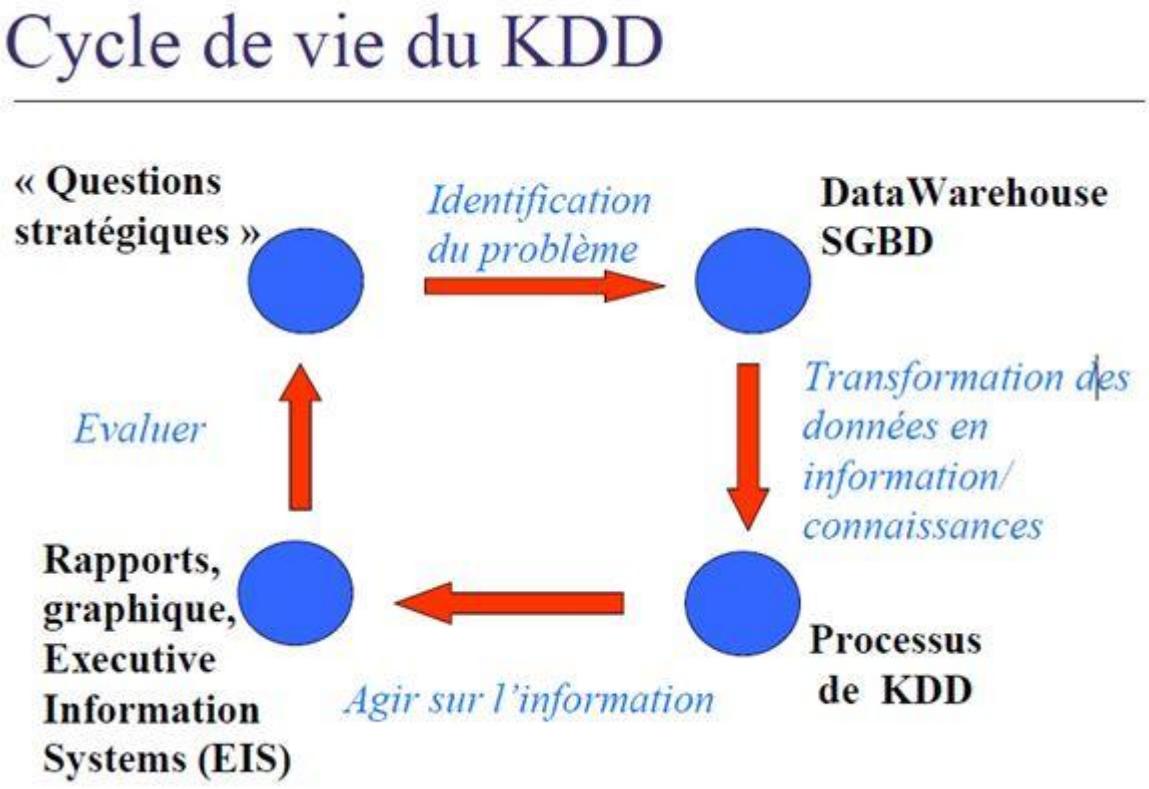
Toujours dans les années 80, Gregory Piatetsky-Shapiro, travaillant pour GTE Laboratories, s'intéresse à l'extraction automatique des données. Il est interpellé par l'étude présentée par Gio Wiederhold à la conférence de Los Angeles en 1987, "Extracting Knowledge From Data". Pour cette étude, Gio Wiederhold a développé un programme nommé Rx Project afin d'analyser la base de données de l'hôpital de Stanford (contenant l'historique de 50 000 patients) recherchant des éventuelles corrélations (effets secondaires) sur les médicaments administrés. Ce fut un succès, Rx a découvert de nombreux effets secondaires totalement inconnus jusqu'alors.

Gregory Piatetsky-Shapiro est alors persuadé que le concept est prometteur et tente d'en convaincre sa direction … En vain … En 1989, il décide d'organiser un atelier sur la découverte de connaissance dans les bases de données, espérant stimuler la recherche dans ce domaine et convaincre enfin sa direction. Se pose alors un dilemme, quel nom donner à son atelier? "Data Mining" est toujours considéré comme un terme péjoratif, "Mining" n'est pas suffisamment explicite, "Database Mining" est déjà utilisé (déposé par la société HNC Software pour Database Mining Workstation) … Il choisit alors "Knowledge Discovery in Databases (KDD)" qui souligne bien l'aspect découverte et insiste particulièrement sur la découverte de connaissances.

La communauté de recherche de découverte de connaissances s'est ensuite développée. Le premier atelier KDD-89, réunissant 50 participants, a généré des conférences internationales, de nombreux cours et des publications.

En 1997, est lancé le premier journal spécialisé "Data Mining and Knowledge Discovery Journal", publiée par Kluwers. (Aujourd'hui, ce journal est publié par Springer).





2. KDD et Data Mining:

Le KDD (ou ECD en français) est "un processus non trivial d'identification de structures inconnues, valides et potentiellement exploitables dans les bases de données (Fayyad*, 1996)".

A l'origine, le Data Mining était une des étapes du KDD (l'étape 3). Aujourd'hui, le Data Mining est utilisé pour désigner tout le processus d'extraction des données.

Actuellement, les termes Data Mining et KDD sont utilisés comme synonyme, Data Mining étant plus usité et populaire.

Quelques définitions de "Data Mining":

- L'extraction d'informations originales, auparavant inconnues, potentiellement utiles à partir de données (Frawley et Piatesky-Shapiro))
- La découverte de nouvelles corrélations, tendances et modèles par le tamisage d'un large volume de données (John Page)
- Un processus d'aide à la décision où les utilisateurs cherchent des modèles d'interprétation dans les données (Kamran Parsaye)
- Torturer l'information disponible jusqu'à ce qu'elle avoue (Dimitris Chorafas)

Ce processus est découpé en six étapes: préparation, nettoyage, enrichissement, codage, fouille et validation. L'enchaînement des différentes étapes est présenté dans la figure 2.1.

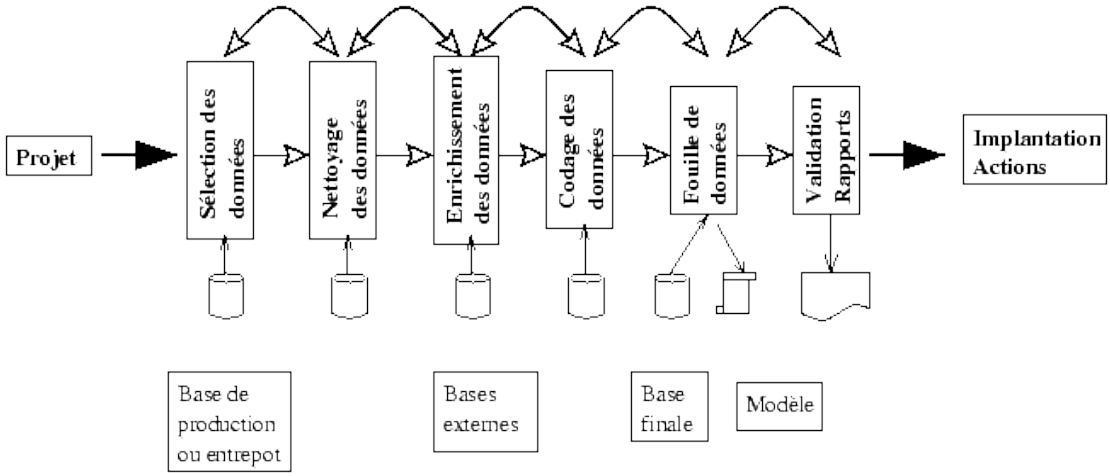


Figure 2.1 : Etapes du processus d'ECD

Ces différentes étapes servent de cadre de référence lorsque l'on doit traiter des données.

Exemple: Cet exemple est issu du livre de P. Adriaans et D. Zantige.

Un éditeur vend 5 sortes de magazines: Sport, Voiture, Maison, Musique et BD. Il souhaite mieux étudier ses clients pour découvrir de nouveaux marchés ou vendre plus de magazines à ses clients habituels. Les questions qu'il se pose sont:

1. *"Combien de personnes ont pris un abonnement à un magazine de sport cette année?"*
2. *"A-t-on vendu plus d'abonnements de magazines de sport cette année que l'année dernière ?"*
3. *"Est-ce que les acheteurs de magazines de BD sont aussi amateurs de sport ?"*
4. *"Quelles sont les caractéristiques principales de mes lecteurs de magazine de voiture ?"*
5. *"Peut-on prévoir les pertes de clients et prévoir des mesures pour les diminuer ?"*

Les questions qui sont proposées ici sont de natures différentes et mettent en jeu des processus différents.

De simples requêtes SQL sont suffisantes pour répondre aux deux premières questions. Les données recherchées dans ce cas ne sont que le résultat d'un calcul simple sur un ou des groupes d'enregistrements. Ce qui distingue ces deux premières questions, c'est la notion de temps et la comparaison. Pour établir cette comparaison, les données doivent être présentes dans la base, ce qui n'est pas toujours le cas pour les bases opérationnelles. Nous pouvons donc supposer que la requête 1 est réalisable sans technique particulière à partir des données opérationnelles sous réserve d'indexations suffisantes des tables concernées. La seule difficulté est de ne pas pénaliser le serveur transactionnel par des requêtes trop longues.

La requête 2 nécessite de conserver toutes les dates de souscription même pour les abonnements résiliés. Nous pouvons imaginer aussi que l'utilisateur émettra une grande variété de requêtes de ce type. Nous supposons alors que ces questions seront résolues à l'aide d'outils de création de requêtes multidimensionnelles de type OLAP.

La question 3 est un exemple simplifié de problème où l'on demande si les données vérifient une règle. La réponse pourrait se formuler par une valeur estimant la probabilité que la règle soit vraie. Souvent, des outils statistiques sont utilisés. Cette question peut être généralisée. On pourrait chercher des associations fréquentes entre acheteurs de magazine pour effectuer des actions promotionnelles. On pourrait également introduire une composante temporelle pour chercher si le fait d'être lecteur d'un magazine implique d'être, plus tard, lecteur d'un autre magazine.

La question 4 est beaucoup plus ouverte. La problématique ici est de *trouver* une règle et non plus de la vérifier ou de l'utiliser. C'est pour ce type de question que l'on met en œuvre des outils de fouille de données et donc le processus décrit ci-dessous.

La question 5 est également une question ouverte. Il faut, pour la résoudre, disposer d'indicateurs qui pourraient être: les durées d'abonnement, les délais de paiement, ... Ce type de question a une forte composante temporelle qui nécessite des données historiques. Elle a de nombreuses applications dans le domaine bancaire, par exemple.

Le processus de KDD utilise des outils de création de requêtes, des outils statistiques et des outils de fouille de données. Là encore, nous nous apercevons qu'une très grande partie des problèmes de décision se traite avec des outils simples. La fouille de données quand elle est nécessaire, suit souvent une analyse des données simple (OLAP) ou statistique.

2.1. Sélection des Données:

Dans notre exemple, nous avons identifié quelques objectifs précis, exprimés sous forme de questions.

La préparation des données consiste dans un premier temps à obtenir des données en accord avec les objectifs que l'on s'impose. Ces données proviennent le plus souvent de bases de production ou d'entrepôts. Les données sont structurées en champs typés (dans un domaine de définition).

Dans l'exemple, nous partons d'une liste des souscriptions d'abonnements avec cinq champs (voir figure 2.2).

client	Nom	Adresse	Date d'abonnement	Magazine
23134	Ahmed	Rue Abane Ramdane, Alger	7/10/96	Voiture
23134	Ahmed	Rue Abane Ramdane, Alger	12/5/96	Musique
23134	Ahmed	Rue Abane Ramdane, Alger	25/7/95	BD
31435	Ali	Cité Emir Abdelkader, Cne	11/11/11	BD
43342	Med	Cité des Annassers, Annaba	30/5/95	Sport
25312	Nabil	Rue Hassiba Benbouali, Alger	25/02/98	NULL
43241	Samir	NULL	14/04/96	Sport
23130	Ahmed	Rue Abane Ramdane, Alger	11/11/11	Maison

Figure 2.2 : Obtention des données

Ces données sont tout d'abord copiées sur une machine adéquate, pour des questions de performance, mais surtout parce qu'elles seront modifiées.

L'obtention des données est souvent réalisée à l'aide d'outils de requêtage (OLAP, SQL,...).

Il faut, dès à présent, noter que l'on ne peut résoudre des problèmes que si l'on dispose des données nécessaires. Il semble, par exemple, impossible de s'attaquer à la question 5 de notre exemple avec les données dont nous disposons.

2.2. Nettoyage des Données:

Il y a évidemment de nombreux points communs entre nettoyage avant l'alimentation des entrepôts (1.2.3) et avant la fouille. Bien sûr, lorsque les données proviennent d'un entrepôt, le travail est simplifié mais reste néanmoins nécessaire : nous avons maintenant un projet bien défini, précis et les données doivent être les plus adaptées possibles.

Reprenons quelques remarques citées dans le chapitre précédent:

Doublons, erreurs de saisie

Les doublons peuvent se révéler gênants parce qu'ils vont donner plus d'importance aux valeurs répétées. Une erreur de saisie pourra à l'inverse occulter une répétition. Dans notre exemple, les clients numéro 23134 et 23130 sont certainement un seul et même client, malgré la légère différence d'orthographe.

Intégrité de domaine

Un contrôle sur les domaines des valeurs permet de retrouver des valeurs aberrantes.

Dans notre exemple, la date de naissance du client 23130 (11/11/11) semble plutôt correspondre à une erreur de saisie ou encore à une valeur par défaut en remplacement d'une valeur manquante.

Informations manquantes

C'est le terme utilisé pour désigner le cas où des champs ne contiennent aucune donnée. Parfois, il est intéressant de conserver ces enregistrements car l'absence d'information peut être une information (ex: détection de fraudes). D'autre part, les valeurs contenues dans les autres champs risquent aussi d'être utiles.

Dans notre exemple, nous n'avons pas le type de magazine pour le client 25312. Il sera écarté de notre ensemble. L'enregistrement du client 43241 sera conservé bien que l'adresse ne soit pas connue.

client	Nom	Adresse	Date d'abonnement	Magazine
23134	Ahmed	Rue Abane Ramdane, Alger	7/10/96	Voiture
23134	Ahmed	Rue Abane Ramdane, Alger	12/5/96	Musique
23134	Ahmed	Rue Abane Ramdane, Alger	25/7/95	BD
31435	Ali	Cité Emir Abdelkader, Cne	NULL	BD
43342	Med	Cité des Annassers, Annaba	30/5/95	Sport
43241	Samir	NULL	14/04/96	Sport
23130	Ahmed	Rue Abane Ramdane, Alger	NULL	Maison

Figure 2.3 : Après nettoyage

Là encore, le langage SQL est utilisé pour la recherche de doublons et des informations manquantes.

2.3. Enrichissement des Données

On peut avoir recours à d'autres bases, achetées ou produites en un autre lieu, pour enrichir nos données. L'opération va se traduire par l'ajout de nouveaux champs en conservant souvent le même nombre d'enregistrements.

Une première difficulté ici est de pouvoir relier des données qui parfois sont hétérogènes. Des problèmes de format de données apparaissent et des conversions sont souvent nécessaires. Une deuxième difficulté est l'introduction de nouvelles valeurs manquantes ou aberrantes et la phase de nettoyage sera certainement de nouveau utile.

Dans l'exemple, supposons que nous ayons accès à des informations sur les clients (figure 2.4).

client	Date de naissance	Revenus	Propriétaire	Voiture
Ahmed	13/1/50	20 000 F	Oui	Oui
Ali	21/5/70	12 000 F	Non	Oui
Med	15/06/63	9 000 F	Non	Non
Samir	27/03/47	15 000 F	Non	Oui

Figure 2.4 : Enrichissement

2.4. Codage des Données

A ce stade du processus, les choix sont particulièrement guidés par l'algorithme de fouille utilisé et des ajustements des choix de codage sont souvent nécessaires.

Regroupements

Certains attributs prennent un très grand nombre de valeurs discrètes. C'est typiquement le cas des adresses. Lorsqu'il est important de considérer ces attributs pour la fouille de données il est obligatoire d'opérer des regroupements et ainsi obtenir un nombre de valeurs raisonnable.

Dans l'exemple, nous regroupons les adresses en 2 régions: Paris, province.

Attributs discrets

Les attributs discrets prennent leurs valeurs (souvent textuelles) dans un ensemble fini donné. C'est le cas de la colonne *magazine* dans notre exemple qui peut prendre les valeurs *Sport, BD, Voiture, Maison, Musique.*

Deux représentations sont possibles pour ces données : une représentation verticale telle qu'elle est présentée en figure 2.3 ou une représentation horizontale ou éclatée (figure 2.5).

	Sport	BD	Voiture	Maison	Musique
23134	0	1	1	0	1
31435	0	1	0	0	0
43342	1	0	0	0	0
43241	1	0	0	0	0

Figure 2.5 : Codage des attributs discrets

La représentation horizontale est plus adaptée à la fouille de données et certains calculs sont simplifiés. Par exemple, la somme de la colonne sport que divise le nombre d'enregistrements calcule le pourcentage de clients ayant souscrit un abonnement à un magazine de sport. Notons que la date d'abonnement dépend du type de magazine. De façon générale, la modification présentée en figure 2.5 peut induire une perte d'information pour tous les champs en dépendance fonctionnelle avec le champ transformé. Nous choisissons de transformer le champ *date d'abonnement* en *date du plus vieil abonnement*. Il est à noter que cette transformation ne nous permet plus espérer générer des règles de la forme : un acheteur de BD s'abonne à un magazine de musique dans les deux ans qui suivent.

Dans notre exemple, le même codage en deux valeurs 0 et 1 sera réalisé avec les champs Oui/Non issus de l'enrichissement.

Changements de type

Pour certaines manipulations, comme des calculs de distance, des calculs de moyenne, il est préférable de modifier les types de certains attributs. Dans notre exemple, la date de naissance et la date d'abonnement ne permettent pas d'effectuer simplement des comparaisons, des différences. Nous pouvons les convertir en âge ou en durée.

Uniformisation d'échelle

Certains algorithmes, comme la méthode des plus proches voisins, sont basés sur des calculs de distance entre enregistrements. Des variations d'échelle selon les attributs sont autant de perturbations possibles pour ces algorithmes. Des échelles très "étirées" vont artificiellement rendre des dimensions trop "vides".

C'est typiquement le cas pour le champ Revenus dans notre exemple. Les centaines de francs ne sont pas significatives ici. Nous convertissons donc les revenus en les divisant par mille. L'intervalle de valeurs est alors dans la même échelle que les dates de naissance et les durées d'abonnement.

	Sport	BD	Voiture	Maison	Musique	Date naissance	Revenus	Propriétaire	Voiture	Paris?	Durée d'abonnement
23134	0	1	1	0	1	50	20	Oui	Oui	1	4
31435	0	1	0	0	0	30	12	Non	Oui	0	NULL
43342	1	0	0	0	0	37	9	Non	Non	0	5
43241	1	0	0	0	0	53	15	Non	Oui	NULL	4

Figure 2.6 : Codage des attributs discrets et normalisation.

2.5. Fouille de Données

La fouille de données est le cœur du processus car elle permet d'extraire de l'information des données. Néanmoins, c'est souvent une étape difficile à mettre en œuvre, coûteuse et dont les résultats doivent être interprétés et relativisés. Il faut aussi noter qu'en situation réelle, pour l'aide à la décision, une très grande majorité des résultats recherchés s'obtient uniquement par requêtes, grâce aux outils de visualisation ou par analyse multidimensionnelle.

Une approche traditionnelle pour découvrir ou expliquer un phénomène est de

- Regarder, explorer,
- Etablir un modèle ou une hypothèse,
- Essayer de le contredire ou le vérifier comme en 1; recommencer le point 2 jusqu'à obtenir une réponse de qualité satisfaisante.

La partie 1 est traditionnellement réalisée avec des outils de reporting ou d'analyse multidimensionnelle. La partie 2 est une hypothèse émise par l'utilisateur. On peut voir les outils de fouille de données comme des procédures qui permettent de faciliter ou encore d'automatiser ce processus.

La qualité du modèle obtenu se mesure selon les critères suivants :

- rapide à créer,
- rapide à utiliser,
- compréhensible pour l'utilisateur,
- les performances sont bonnes; Le modèle est fiable,
- les performances ne se dégradent pas dans le temps,
- Il évolue facilement.

Il va de soit qu'aucun modèle n'aura toutes ces qualités. Il n'existe pas de meilleure méthode de fouille. Il faudra faire des compromis selon les besoins dégagés et les caractéristiques connues des outils. Pour une utilisation optimale, une combinaison de méthodes est recommandée. On peut rapidement donner 3 catégories:

Classification, régression, prédiction

Il s'agit de trouver une classe ou une valeur selon un ensemble de descriptions. Ce sont des outils très utilisés. Les algorithmes reposent sur des arbres de décision, des réseaux de neurones, la règle de Bayes, les k plus proches voisins.

Association, Sequencing

Il s'agit de trouver des similarités ou des associations. Le sequencing est le terme anglais utilisé pour préciser que l'association se fera dans le temps. Par exemple, si j'achète un couffin aujourd'hui, j'ai trois fois plus de chance dans 3 mois d'acheter un

lit bébé (sequencing) ou encore si j'achète des pâtes et de la purée de tomates, j'ai deux fois plus de chance d'acheter aussi du parmesan (association).

Segmentation

La problématique est de trouver des groupes homogènes dans une population. On utilise souvent les algorithmes des k-moyennes ou de Kohonen. La difficulté essentielle dans ce type de construction est la validation qui nécessite l'intervention d'experts humains.

2.6. Validation

Les méthodes de validation vont déprendre de la nature de la tâche et du problème considéré. Nous distinguerons deux modes de validation : statistique et par expertise. Pour certains domaines d'application (le diagnostic médical, par exemple), il est essentiel que le modèle produit soit compréhensible. Il y a donc une première validation du modèle produit par l'expert, celle-ci peut être complétée par une validation statistique sur des bases de cas existantes.

Pour les problèmes d'apprentissage non supervisé (segmentation, association), la validation est essentiellement du ressort de l'expert. Pour la segmentation, le programme construit des groupes homogènes, un expert peut juger de la pertinence des groupes constitués. La encore, on peut combiner avec une validation statistique sur un problème précis utilisant cette segmentation. Pour la recherche des règles d'association, c'est l'expert du domaine qui jugera de la pertinence des règles. En effet, l'algorithme, s'il fournit des règles porteuses d'information, produit également des règles triviales et sans intérêt.

Pour la validation statistique, la première tâche à réaliser consiste à utiliser les méthodes de base de statistique descriptive. L'objectif est d'obtenir des informations qui permettront de juger le résultat obtenu, ou d'estimer la qualité ou les biais des données d'apprentissage.

- Calculer les moyennes et variances des attributs.
- Si possible, calculer la corrélation entre certains champs.
- Déterminer la classe majoritaire dans le cas de la classification.

Pour la classification supervisée, la deuxième tâche consiste à décomposer les données en plusieurs ensembles disjoints. L'objectif est de garder des données pour estimer les erreurs des modèles ou de les comparer. Il est souvent recommandé de constituer:

- Un ensemble d'apprentissage.
- Un ensemble de test.
- Un ensemble de validation.

Au moins deux ensembles sont nécessaires : l'ensemble d'apprentissage permet de générer le modèle, l'ensemble test permet d'évaluer l'erreur réelle du modèle sur un ensemble indépendant évitant ainsi un biais d'apprentissage. Lorsqu'il s'agit de tester plusieurs modèles et de les comparer, on peut sélectionner le meilleur modèle selon ses performances sur l'ensemble test et ensuite évaluer son erreur réelle sur l'ensemble de validation.

Lorsqu'on ne dispose pas de suffisamment d'exemples, on peut se permettre d'apprendre et d'estimer les erreurs avec un même ensemble par la technique de *validation croisée*. La validation croisée ne construit pas de modèle utilisable mais ne sert qu'à estimer l'erreur réelle d'un modèle selon l'algorithme suivant:

Validation croisée (*S*,*x*)
// *S* est un ensemble, *x* est un entier
Découper *S* en *x* parties égales $\{S_1,...,S_x\}$
Pour *i* de 1 à *x*
Construire un modèle *M* avec l'ensemble S-S_i
Evaluer l'erreur e_i de *M* avec S_i
Fin Pour
Retourner la moyenne des $e_i=\sum_{i=1}^{x} e_i/x$

3. Algorithmes de Fouille de Données

3.1. Introduction

On se situe dans un environnement d'aide à la décision à partir de données. On suppose que de grandes quantités de données sont disponibles. En règle générale, ces données sont structurées et correspondent aux enregistrements d'une table ou de plusieurs tables d'une base dédiée à la décision (entrepôt de données). Nous allons présenter, dans ce qui suit, les tâches à réaliser ou les problèmes à résoudre à l'aide de quelques-unes des méthodes disponibles pour.

Les tâches

On dispose de données structurées. Les objets sont représentés par des enregistrements (ou descriptions) qui sont constitués d'un ensemble de champs (ou attributs) prenant leurs valeurs dans un domaine. On peut mettre en évidence différentes problématiques. Les termes employés pouvant varier d'une discipline à l'autre (parfois même dans une même discipline selon le domaine d'application), nous définissons notre vocabulaire avec la description associée de la tâche.

La classification

Consiste à examiner les caractéristiques d'un objet et lui attribuer une classe, la classe est un champ particulier à valeurs discrètes. Des exemples de tâche de classification sont :

- Attribuer ou non un prêt à un client,
- Etablir un diagnostic,
- Accepter ou refuser un retrait dans un distributeur,
- Attribuer un sujet principal à un article de presse, ...

L'estimation

Consiste à estimer la valeur d'un champ à partir des caractéristiques d'un objet. Le champ à estimer est un champ à valeurs continues. L'estimation peut être utilisée dans un but de classification. Il suffit d'attribuer une classe particulière pour un intervalle de valeurs du champ estimé. Des exemples de tâche d'estimation sont:

- Noter un candidat à un prêt ; cette estimation peut être utilisée pour attribuer un prêt (classification), par exemple, en fixant un seuil d'attribution,
- Estimer les revenus d'un client.

La prédiction

Consiste à estimer une valeur future. En général, les valeurs connues sont historisées. On cherche à prédire la valeur future d'un champ. Cette tâche est proche des

précédentes. Les méthodes de classification et d'estimation peuvent être utilisées en prédiction. Des exemples de tâche de prédiction sont:

- Prédire les valeurs futures d'actions,
- Prédire au vu de leurs actions passées les départs de clients.

Les règles d'association (analyse du panier de la ménagère)

Consiste à déterminer les valeurs qui sont associées. L'exemple type est la détermination des articles (le poisson et le vin blanc ; la baguette et le camembert et le vin rouge, ...) qui se retrouvent ensemble sur un même ticket de supermarché. Cette tâche peut être effectuée pour identifier des opportunités de vente croisée et concevoir des groupements attractifs de produit. C'est une des tâches qui nécessite de très grands jeux de données pour être effective. Cette tâche a engendré l'exemple (l'anecdote) suivant présent dans de nombreux articles sur le Data Mining : dans les supermarchés américains, il a été possible de mettre en évidence des corrélations entre achat de bières et achat de couches-culottes avant le week-end, remarque justifiée par le comportement des jeunes pères américains qui préparent leurs week-ends en préparant leurs provisions de bière pour regarder la télévision et qui font les achats pour bébé au même moment. Gag ou réalité, je n'ai jamais lu de résultats sur les actions consistant à mettre, dans les supermarchés, les bières à côté des couches-culottes!!

La segmentation

Consiste à former des groupes (clusters) homogènes à l'intérieur d'une population. Pour cette tâche, il n'y a pas de classe à expliquer ou de valeur à prédire définie a priori, il s'agit de créer des groupes homogènes dans la population (l'ensemble des enregistrements). Il appartient ensuite à un expert du domaine de déterminer l'intérêt et la signification des groupes ainsi constitués. Cette tâche est souvent effectuée avant les précédentes pour construire des groupes sur lesquels on applique des tâches de classification ou d'estimation.

Les données

Ce sont les valeurs des champs des enregistrements des tables de l'entrepôt. Ces données possèdent un type qu'il est important de préciser. En effet, la plupart des méthodes sont sensibles aux données manipulées. Par exemple, certaines méthodes sont mises en défaut par les données continues alors que d'autres peuvent être sensibles à la présence de données discrètes.

Les données discrètes

- les données binaires ou logiques : 0 ou 1 ; oui ou non ; vrai ou faux. ce sont des données telles que le sexe, être bon client, ...
- les données énumératives : ce sont des données discrètes pour lesquelles il n'existe pas d'ordre défini a priori. Par exemple : la catégorie socioprofessionnelle, la couleur, ...
- les données énumératives ordonnées : les réponses à une enquête d'opinion (1: très satisfait ; 2 : satisfait ; ...), les données issues de la discrétisation de données continues (1 : solde moyen < 2000 ; 2 : 2000 □ solde moyen < 5000 ; ...)

Les données continues

Ce sont les données entières ou réelles : l'âge, le revenu moyen, ..., mais aussi les données pouvant prendre un grand nombre de valeurs ordonnées.

Les dates

Sont souvent problématiques car mémorisées selon des formats différents selon les systèmes et les logiciels. Pour les applications en fouille de données, il est fréquent de les transformer en données continues ou en données énumératives ordonnées. On transforme une date de naissance en âge entier ou en une variable énumérative ordonnée correspondant à des tranches d'âge.

Les données textuelles

Un texte peut, pour certaines applications, être résumé comme un n-uplet constitué du nombre d'occurrences dans le texte de mots clés d'un dictionnaire prédéfini.

Les méthodes

Les méthodes sont subdivisées en deux groupes: les méthodes supervisées et les méthodes non supervisées. La présentation ne prétend pas à l'exhaustivité. Les choix effectués sont subjectifs, quoique guidés par la disponibilité des méthodes dans les environnements de Data Mining standard. En tout état de cause, un fait important est:

Il n'existe pas de méthode supérieure à toutes les autres

Par conséquent, à tout jeu de données et tout problème correspond une ou plusieurs méthodes. Le choix se fera en fonction

- de la tâche à résoudre,
- de la nature et de la disponibilité des données,
- des connaissances et des compétences disponibles,
- de la finalité du modèle construit. Pour cela, les critères suivants sont importants: complexité de la construction du modèle, complexité de son utilisation, ses performances, sa pérennité et
- plus généralement, de l'environnement de l'entreprise.

Nous nous attachons dans la présentation des différentes méthodes à préciser les particularités de chacune des méthodes selon ces axes.

3.2. Quelques Méthodes

a. Réseau de Neurones

Les réseaux de neurones sont des outils très utilisés pour la classification, l'estimation, la prédiction et la segmentation. Ils sont issus de modèles biologiques, sont constitués d'unités élémentaires (les neurones) organisées selon une architecture. Nous nous limitons dans ce paragraphe aux réseaux de neurones dédiés aux tâches d'estimation et classification que sont les *Perceptrons multicouches (PMC)*. Ceux-ci obtiennent de bonnes performances, en particulier, pour la reconnaissance de formes et sont donc bien adaptés pour des problèmes comprenant des variables continues éventuellement bruitées. Le principal désavantage est qu'un réseau est défini par une architecture et un grand ensemble de paramètres réels (les coefficients synaptiques), le pouvoir explicatif est faible: on parle parfois de "***boîte noire"***.

Qu'est-ce qu'un réseau de neurones ?

Un neurone

Un neurone est une unité de calcul élémentaire dont le modèle est issu de certains principes de fonctionnement du neurone biologique. L'unité de calcul combine des entrées réelles $x_1,...,x_n$ en une sortie réelle o. Les entrées n'ont pas toutes la même importance et à chaque entrée x_i est associée un poids (ou coefficient synaptique) w_i. L'unité calcule d'abord l'activité d'entrée. En règle générale, pour le neurone formel, l'activité en entrée est mesurée par la somme pondérée des entrées $\Sigma_i w_i x_i$.

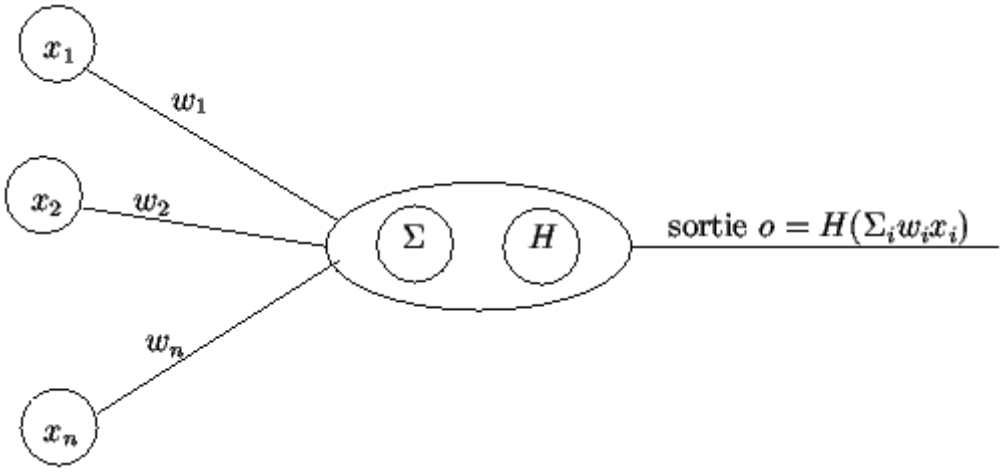


Figure 3.1: Un Neurone ou Perceptron

Un Perceptron Multicouches (PMC)

Le perceptron multicouches est un modèle de réseaux de neurones. Les neurones sont répartis en couches successives, les calculs sont effectués des entrées vers la ou les sorties, le neurone élémentaire est, en règle générale, celui considéré dans le paragraphe précédent.

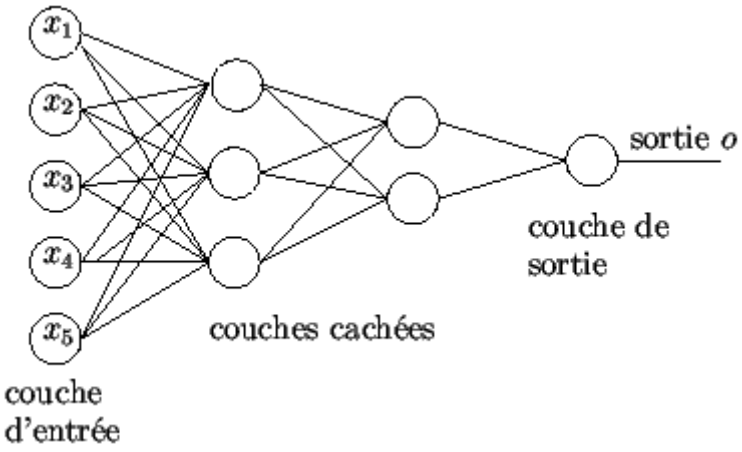


Figure 3.2: Un perceptron multicouches: une couche d'entrée de 5 cellules, 2 couches cachées possédant respectivement 3 et 2 neurones, 1 couche de sortie à 1 neurone

La première couche est la *couche d'entrée* : aucun calcul n'est effectué, une cellule d'entrée ne fait que copier son entrée vers sa sortie. Les entrées correspondent aux attributs (ou leurs codages) du problème considéré. Ensuite, viennent une ou

plusieurs *couches cachées* constituées de neurones élémentaires. Tous les neurones d'une couche sont les entrées de chaque neurone de la couche suivante et d'elle seulement : autrement dit, il n'y a pas de retour arrière et on passe d'une couche à la suivante. Enfin, la dernière couche est la *couche de sortie* qui contient un ou plusieurs neurones. Les sorties de ces neurones correspondent aux valeurs à estimer (ou leurs codages) pour le problème considéré.

Algorithme d'apprentissage

Dans la très jeune histoire de l'informatique, il faut savoir que les réseaux de neurones sont anciens (première définition du neurone en 43 par McCulloch et Pitts), que les recherches ont été nombreuses et prometteuses. Cependant, il a été vite constaté que, pour résoudre des problèmes complexes, le neurone élémentaire (éventuellement légèrement complexifié: perceptron de Rosenblatt 58) ne suffisait pas. Il fallait donc considérer des réseaux de neurones tels que le PMC défini auparavant. Le problème, qui a freiné les études sur les réseaux de neurones, est que l'on ne disposait pas d'algorithme d'apprentissage pour ces architectures. Il a fallu attendre le début des années 80 pour qu'un tel algorithme apparaisse : *l'algorithme de rétropropagation du gradient* défini par divers auteurs (Rumelhart, McClelland, Parker, Hinton, Le Cun).

Le principe de l'algorithme est le suivant :

- On initialise aléatoirement les poids de tous les neurones du réseau,
- de façon itérative, on présente un élément x de S en entrée du réseau, le réseau calcule une sortie o(x□), la sortie attendue est c(x□), on peut mesurer l'écart entre sortie calculée et sortie attendue, on procède alors à une rétropropagation du gradient, c'est-à-dire qu'à l'aide d'une méthode mathématique (méthode du gradient), on modifie les poids (les coefficients synaptiques) de la couche de sortie vers la couche d'entrée pour minimiser l'erreur sur cet exemple ;
- un critère d'arrêt doit être défini.

Nous supposons toujours que l'architecture du réseau est fixée. Nous allons présenter les différents paramètres à régler pour l'apprentissage d'un PMC (ils sont résumés dans la figure 3.1). Tout d'abord, on choisit une *fonction d'activation*. Un fichier d'apprentissage étant défini, on initialise aléatoirement les poids. Il faut noter que des initialisations différentes peuvent produire des résultats différents.

Paramètre	Objet du paramètre
fonction d'activation	fonction d'activation pour le neurone élémentaire, souvent la sigmoïde
nombre d'exemples	nombre d'exemples avant mise à jour des poids, par défaut 1
taux d'apprentissage	coefficient qui contrôle la vitesse de modification des poids
inertie	coefficient qui permet de lisser la variation des poids
tolérance	seuil d'erreur pour arrêt de l'apprentissage

Table 3.1: Paramètres d'apprentissage pour la rétropropagation

Le paramètre *nombre d'exemples* définit le nombre d'exemples qui sont passés dans le réseau avant qu'un calcul d'erreur soit effectué et que les poids soient mis à jour. Les valeurs extrêmes sont 1 et *m* : le cardinal de l'ensemble d'apprentissage. Pour la valeur 1, il y a mise à jour des poids après chaque exemple. Dans ce cas, la convergence est rapide, mais on peut tomber dans un minimum local (on trouve un minimum de l'erreur mais ce n'est pas l'erreur minimale que l'on peut espérer). Pour la valeur *m*, on passe l'échantillon complet, on calcule l'erreur $E(PMC)$ définie dans l'équation 3.4, puis on met à jour les poids. La convergence est plus lente.

Le paramètre *taux d'apprentissage* souvent nommé η ou ε est une valeur réelle de l'intervalle [0,1] qui influe sur la modification des poids. Une valeur grande signifie que l'on modifie fortement les poids à chaque mise à jour, une valeur petite entraîne des modifications minimes de ces poids. Il est fréquent de choisir une valeur initiale qui diminue avec le nombre d'itérations. La valeur initiale est choisie "assez grande" (0.2 ou 0.1 par défaut) pour une convergence rapide, puis diminue pour éviter les oscillations et converger vers un minimum.

Le paramètre *inertie* souvent nommé α est une valeur réelle de l'intervalle [0,1] qui lisse les modifications de poids. Plus précisément, si on modifie les poids après chaque exemple, il se peut que certains poids soient alternativement augmentés ou diminués. Pour atténuer ce phénomène, on modifie les poids en tenant compte des modifications faites aux étapes précédentes.

La *tolérance* définit l'erreur cible, c'est-à-dire l'erreur que l'on souhaite atteindre avant de s'arrêter. Elle est choisie en fonction du mode de calcul de l'erreur, l'erreur $E(PMC)$ étant parfois normalisée. Des critères d'arrêt sur le nombre d'exemples de l'échantillon d'apprentissage *S* qui sont bien classés (ou bien estimés : erreur inférieure à un certain seuil) sont parfois ajoutés au précédent.

Codage du problème

Les PMC prennent des entrées réelles. Cependant, pour un bon fonctionnement des algorithmes, il est souvent nécessaire de normaliser les entrées dans l'intervalle [0,1]. Cette normalisation peut être prise en charge par le générateur de réseaux de neurones ou est à la charge du programmeur. Les entrées binaires ne nécessitent pas de codage. Pour les données énumératives, plusieurs codages sont possibles. Prenons l'exemple d'un attribut A prenant ses valeurs dans l'ensemble {1,2,3,4,5}. Plusieurs codages sont envisageables:

- 5 entrées à valeurs binaires ; dans ce cas, la donnée A=3 est codée par la troisième entrée vaut 1, les autres valent 0.
- on peut coder 5 valeurs sur 3 bits. L'attribut A peut alors être codé sur 3 entrées ; par exemple, la donnée A=3 est codée par première entrée à 0, deuxième entrée à 1, troisième entrée à 1 si on choisit le codage binaire. D'autres codages sur 3 bits sont possibles si une notion d'ordre ou de proximité existe sur les valeurs de l'attribut A.
- 1 entrée réelle ; dans ce cas, on peut par exemple coder les valeurs 1, 2, 3, 4 et 5 par 0, 0.25, 0.5, 0.75 et 1.

Les PMC peuvent avoir une ou plusieurs sorties réelles. Pour un problème d'estimation d'une ou plusieurs variables, un neurone de sortie sert à estimer une des variables (normalisée). Pour un problème de classification à plusieurs classes,

on peut créer une sortie pour chacune des classes ou coder en binaire (pour quatre classes, deux sorties suffisent). Remarquer que le codage "une sortie par classe" est intéressant si un même exemple peut appartenir à plusieurs classes.

Choix de l'architecture

Le codage étant choisi, on connaît le nombre d'entrées et le nombre de sorties du réseau. Il reste à choisir l'architecture. Un exemple d'architectures possibles est présenté dans la Figure 3.9. Plus la structure du réseau est riche, plus le pouvoir d'expression du modèle est grand. Sur l'exemple présenté, l'architecture 4-1 est moins puissante que l'architecture 4-2-1, elle-même moins puissante que l'architecture 4-4-1.

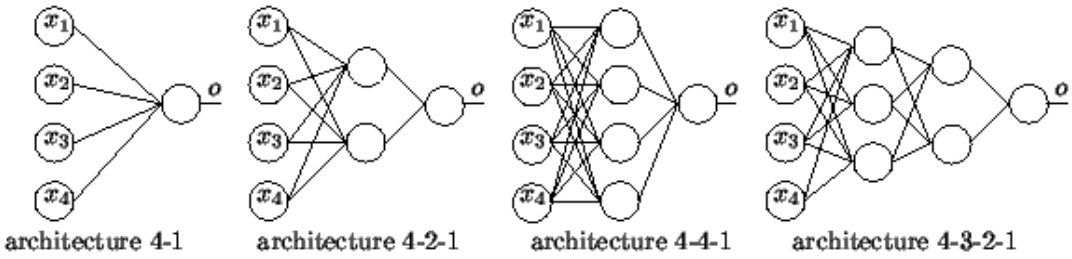


Figure 3.3 : différentes architectures possibles pour un problème à quatre entrées et une sortie

Si on souhaite avoir un réseau de neurones ayant un bon pouvoir de généralisation, il faut choisir une architecture assez riche, mais pas trop ! En effet, on rencontre ici la même difficulté que pour les arbres de décision, il faut éviter la sur-spécialisation et si l'architecture est trop riche, on prend le risque d'un apprentissage "par cœur", sans pouvoir de généralisation. Le choix de l'architecture est fait, soit par l'expérience, soit par des essais successifs.

Réglage des paramètres et choix de l'architecture

Nous venons de signaler l'importance du choix de l'architecture sur la qualité du réseau de neurones produit. Il faut également, une architecture étant choisie, régler les différents paramètres d'apprentissage pour générer un << bon >> réseau de neurones. Pour cela, si les données disponibles sont en nombre suffisant, on découpe l'échantillon en trois ensembles : un ensemble d'apprentissage A, un ensemble de test T et un ensemble de validation V. Pour un choix d'architecture et un réglage des paramètres, on apprend avec A, on estime la qualité du réseau produit avec T. Ensuite, on choisit l'architecture et le réglage ayant obtenu les meilleures performances sur T. On relance l'apprentissage, l'estimation de l'erreur réelle est obtenue en calculant l'erreur sur V.

Avantages et Inconvénients de la méthode

- **Lisibilité du résultat**: Le résultat de l'apprentissage est un réseau constitué de cellules organisées selon une architecture, définies par une fonction d'activation et un très grand nombre de poids à valeurs réelles. Ces poids sont difficilement interprétables. Pour un vecteur d'entrée, il est difficile d'expliquer le pourquoi de la sortie calculée.
- **Les données réelles:** Les réseaux traitent facilement les données réelles (préalablement normalisées) et les algorithmes sont robustes au bruit. Ce sont, par conséquent, des outils bien adaptés pour le traitement de données complexes

éventuellement bruitées comme la reconnaissance de formes (son, images sur une rétine, ...).

- **Classification efficace:** Les algorithmes de génération de réseaux de neurones sont disponibles dans tous les environnements de fouille de données.
- **Paramètres d'apprentissage:** Il n'est pas facile, sans expérience approfondie, de choisir l'architecture et de régler les paramètres d'apprentissage. Il est possible de procéder par différents essais mais le point suivant nous montre que ce n'est pas toujours possible.
- **Temps d'apprentissage:** L'échantillon nécessaire à l'apprentissage doit être suffisamment grand et représentatif des sorties attendues. Il faut passer un grand nombre de fois tous les exemples de l'échantillon d'apprentissage avant de converger et donc le temps d'apprentissage peut être long.
- **Évolutivité dans le temps:** Comme pour les arbres de décision, l'apprentissage n'est pas incrémental et, par conséquent, si les données évoluent avec le temps, il est nécessaire de relancer une phase d'apprentissage pour s'adapter à cette évolution.
- **En combinaison avec d'autres méthodes:** Pour des problèmes contenant un grand nombre d'attributs pour les entrées, il peut être très difficile de construire un réseau de neurones. On peut, dans ce cas, utiliser les arbres de décision pour sélectionner les variables pertinentes, puis générer un réseau de neurones en se restreignant à ces entrées.
- **Extensions:** Les extensions sont nombreuses pour les tâches de classification et d'estimation : autres fonctions d'activation, algorithmes d'apprentissage, apprentissage ***"on line"***, ... Rappelons également que des modèles de réseaux de neurones existent pour les tâches de prédiction (réseaux récurrents : la sortie d'un neurone peut influer sur les neurones des couches précédentes) et pour les tâches de segmentation (réseaux associatifs, cartes de Kohonen).

b. Recherche des Plus Proches Voisins

La méthode des plus proches voisins (PPV en bref, *nearest neighbor* en anglais ou kNN for short) est une méthode dédiée à la classification qui peut être étendue à des tâches d'estimation. La méthode PPV est une méthode de raisonnement à partir de cas. Elle part de l'idée de prendre des décisions en recherchant un ou des cas similaires déjà résolus en mémoire. Contrairement aux autres méthodes de classification qui seront étudiées dans les sections suivantes (arbres de décision, réseaux de neurones, algorithmes génétiques), il n'y a pas d'étape d'apprentissage consistant en la construction d'un modèle à partir d'un échantillon d'apprentissage. C'est l'échantillon d'apprentissage, associé à une fonction de distance et d'une fonction de choix de la classe en fonction des classes des voisins les plus proches, qui constitue le modèle. L'algorithme générique de classification d'un nouvel exemple par la méthode PPV est :

Algorithme de classification par k-PPV

Paramètre : le nombre k de voisins

Donnée : un échantillon de m enregistrements classés $(\vec{x},c(\vec{x}))$

Entrée : un enregistrement $\vec{y}$

1. déterminer les k plus proches enregistrements de $\vec{y}$

2. combiner les classes de ces *k* exemples en une classe *c*

Sortie : la classe de $y^{\rightarrow}$ est $c(y^{\rightarrow})=c$

Nous allons, par conséquent, présenter les différents choix possibles pour la définition de la distance et pour le mode de sélection de la classe du cas présenté.

Définition de la distance

Le choix de la distance est primordial au bon fonctionnement de la méthode. Quoique les distances les plus simples permettent d'obtenir des résultats satisfaisants (lorsque c'est possible), il faut être attentif à différents pièges. Tout d'abord, rappelons qu'une distance doit satisfaire les propriétés suivantes:

$$d(A,A)=0 d(A,B)= d(B,A) d(A,B) \leq d(A,C) + d(B,C)$$

Pour tous points A, B et C de l'espace. Dans notre cadre, les points sont des enregistrements d'une base de données. D'après la première de ces propriétés, le plus proche voisin d'un enregistrement est lui-même. Par la suite, nous désignons par plus proche voisin un point le plus proche de l'enregistrement autre que lui-même. Les deux propriétés suivantes rendent la notion de proximité stable. On peut cependant noter qu'un point A peut avoir un plus proche voisin B tandis que B possède de nombreux voisins plus proches que A (Figure 3.4).

Figure 3.4 : *A* à un plus proche voisin *B*, *B* a de nombreux voisins proches autres que *A*

Pour définir la fonction de distance, on définit d'abord une distance sur chacun des champs, puis on combine ces distances pour définir la distance globale entre enregistrements. Commençons par étudier les choix possibles selon le type du champ.

Champs numériques

La distance entre deux valeurs numériques x et y peut être choisie égale à :

$$d(x,y) = |x-y|$$

Champs discrets

La définition dépend alors des valeurs possibles.

- **Les données binaires**: 0 ou 1. On choisit d(0,0)=d(1,1)=0 et d(0,1)=d(1,0)=1.
- **Les données énumératives**: Distance = 0 si les valeurs sont égales et 1 sinon.
- **Les données énumératives ordonnées**: elles peuvent être considérées comme des valeurs énumératives mais on peut également définir une distance utilisant la relation d'ordre. Par exemple, si un champ prend les valeurs 1, 2, 3, 4 et 5, on peut définir la distance en considérant 5 points de l'intervalle [0,1] avec une distance de 0,2 entre deux points successifs, on a alors

  d(1,2) = 0,2 d(1,3) = 0,4 ... d(4,5) = 0,2.

Autres champs

Une distance peut être définie sur de nombreux types de champs comme des textes, des images, des informations géographiques.

Il reste à combiner les distances entre champs pour définir une distance entre enregistrements. Soit $x_i = (x_1,..., x_n)$ et $y_j = (y_1,..., y_n)$ deux enregistrements, d_1, ..., d_n sont les distances définies sur les différents champs, la distance entre deux enregistrements peut être définie par:

- la distance euclidienne : $d_e(x^{\rightarrow},y^{\rightarrow})= d_1(x_1,y_1)^2+...+d_n(x_n,y_n)^2$,
- la sommation : $d_s(x^{\rightarrow},y^{\rightarrow})= d_1(x_1,y_1) +...+ d_n(x_n,y_n)$.

Nous avons présenté différents choix possibles, selon le type des données, pour définir une distance sur un champ. Des variantes de ces définitions existent et, parfois, une distance spécifique peut être définie en fonction d'une expertise. De même, il existe de nombreuses variantes pour combiner les distances entre champs. Il est également possible de pondérer l'importance des différents champs.

Sélection de la classe

L'idée de la méthode est la recherche de cas similaires au cas à résoudre et d'utiliser les décisions des cas proches déjà résolus pour choisir une décision. La méthode la plus simple est de rechercher le cas le plus proche et de prendre la même décision. C'est la méthode 1-PPV (1-NN) du plus proche voisin. Si cette méthode peut fournir de bons résultats sur des problèmes simples pour lesquels les points (les enregistrements) sont bien répartis en groupes denses d'enregistrements de même classe, en règle générale, il faut considérer un nombre de voisins plus important pour obtenir de bons résultats.

Une première façon de combiner les k classes des k voisins les plus proches est le *vote majoritaire*. Elle consiste simplement à prendre la classe majoritaire. Dans le cas de deux classes, on choisit une valeur de k impaire.

Une seconde façon est le *vote majoritaire pondéré*. Chaque vote, c'est-à-dire chaque classe d'un des k voisins sélectionnés, est pondéré.

Dans les deux cas précédents, il est possible de définir une confiance dans la classe attribuée égale au rapport entre les votes gagnants et le total des votes.

Lorsque la technique est appliquée à une tâche d'estimation, donc à prédire la valeur d'un attribut continu, la notion de vote perd tout son sens. Une première solution pour combiner les réponses est l'interpolation, c'est-à-dire de calculer une moyenne pondérée des réponses. Un défaut de cette solution est de ***"lisser"*** les données. Une deuxième solution est de considérer les *k* enregistrements avec la valeur prédite correspondante et d'utiliser les techniques de régression linéaire pour estimer la valeur en $y^{\rightarrow}$.

Mise en place de la méthode

La méthode ne nécessite pas de phase d'apprentissage. Le modèle sera constitué de l'échantillon d'apprentissage, de la distance et de la méthode de combinaison des voisins.

Il faut choisir l'échantillon, c'est-à-dire les attributs pertinents pour la tâche de classification considérée et l'ensemble des enregistrements. Il faut veiller à disposer d'un nombre assez grand d'enregistrements par rapport au nombre d'attributs et à ce que chacune des classes soit bien représentée dans l'échantillon choisi.

Le choix de la distance par champ et du mode de combinaison des distances se fait en fonction du type des champs et des connaissances préalables du problème. Il est possible de choisir la distance en faisant varier cette distance et, pour chacun des choix, estimer l'erreur réelle. On choisit alors la distance donnant la meilleure erreur réelle estimée. L'estimation de l'erreur réelle se fait classiquement avec un ensemble test ou la validation croisée.

Le choix du nombre *k* de voisins peut, lui aussi, être déterminé par utilisation d'un ensemble test ou par validation croisée. Une heuristique fréquemment utilisée est de prendre *k* égal au nombre d'attributs plus 1. La méthode de vote ou d'estimation peut aussi être choisie par test ou validation croisée.

Avantages et Inconvénients de la Méthode

- **Pas d'apprentissage**

  C'est l'échantillon qui constitue le modèle. L'introduction de nouvelles données permet d'améliorer la qualité de la méthode sans nécessiter la reconstruction d'un modèle. C'est une différence majeure avec des méthodes telles que les arbres de décision et les réseaux de neurones.

- **Clarté des résultats**

  Bien que la méthode ne produit pas de règle explicite, la classe attribuée à un exemple peut être expliquée en exhibant les plus proches voisins qui ont amené à ce choix.

- **Tout type de données**

  La méthode peut s'appliquer dès qu'il est possible de définir une distance sur les champs. Or, il est possible de définir des distances sur des champs complexes tels que des informations géographiques, des textes, des images, du son. C'est parfois un critère de choix de la méthode PPV car les autres méthodes traitent difficilement les données complexes. On peut noter, également, que la méthode est robuste au bruit.

- **Nombre d'attributs**

  La méthode permet de traiter des problèmes avec un grand nombre d'attributs. Mais, plus le nombre d'attributs est important, plus le nombre d'exemples doit être grand. En effet, pour que la notion de proximité soit pertinente, il faut que les exemples couvrent bien l'espace et soient suffisamment proches les uns des autres. Si le nombre d'attributs pertinents est faible relativement au nombre total d'attributs, la méthode donnera de mauvais résultats car la proximité sur les attributs pertinents sera noyée par les distances sur les attributs non pertinents. Il est donc parfois utile de d'abord sélectionner les attributs pertinents.

- **Temps de classification**

  Si la méthode ne nécessite pas d'apprentissage, tous les calculs doivent être effectués lors de la classification. Ceci est la contrepartie à payer par rapport aux méthodes qui nécessite un apprentissage (éventuellement long) mais qui sont rapides en classification (le modèle est créé, il suffit de l'appliquer à l'exemple à classifier). Certaines méthodes permettent de diminuer la taille de l'échantillon en ne conservant que les exemples pertinents pour la méthode PPV, mais il faut, de toute façon, un nombre d'exemples suffisamment grand relativement au nombre d'attributs.

- **Stocker le modèle**

  Le modèle est l'échantillon, il faut donc un espace mémoire important pour le stocker ainsi que des méthodes d'accès rapides pour accélérer les calculs.

- **Distance et nombre de voisins**

  Les performances de la méthode dépendent du choix de la distance, du nombre de voisins et du mode de combinaison des réponses des voisins. En règle générale, les distances simples fonctionnent bien. Si les distances simples ne fonctionnent pour aucune valeur de k, il faut envisager le changement de distance, ou le changement de méthode !

c. Recherche d'Associations

Les règles d'association sont traditionnellement liées au secteur de la distribution car leur principale application est ***"l'analyse du panier de la ménagère"*** qui consiste en la recherche d'associations entre produits sur les tickets de caisse. Le but de la méthode est l'étude de ce que les clients achètent pour obtenir des informations sur qui sont les clients et pourquoi ils font certains achats. La méthode recherche *quels produits tendent à être achetés ensemble*. La méthode peut être appliquée à tout secteur d'activité pour lequel il est intéressant de rechercher des groupements potentiels de produits ou de services: services bancaires, services de télécommunications, par exemple. Elle peut être également utilisée dans le secteur médical pour la recherche de complications dues à des associations de médicaments ou à la recherche de fraudes en recherchant des associations inhabituelles.

Un attrait principal de la méthode est la clarté des résultats produits. En effet, le résultat de la méthode est un ensemble de *règles d'association*. Des exemples de règles d'association sont:

- Si un client achète des plantes alors il achète du terreau,
- Si un client achète du poisson et du citron alors il achète de la limonade,
- Si un client achète une télévision, il achètera un magnétoscope dans un an.

Ces règles sont intuitivement faciles à interpréter car elles montrent comment des produits ou des services se situent les uns par rapport aux autres. Ces règles sont particulièrement utiles en marketing. Les règles d'association produites par la méthode peuvent être facilement utilisées dans le système d'information de l'entreprise. Cependant, il faut noter que la méthode, si elle peut produire des règles intéressantes, peut aussi produire des règles triviales (déjà bien connues des intervenants du domaine) ou inutiles (provenant de particularités de l'ensemble d'apprentissage). La recherche de règles d'association est une méthode non supervisée car on ne dispose en entrée que de la description des achats.

Introduction de la Méthode

On suppose avoir prédéfini une classification des articles (nous reviendrons sur ce point par la suite). Les données d'entrée sont constituées d'une liste d'achats. Un achat est lui-même constitué d'une liste d'articles. On peut remarquer que, contrairement aux enregistrements d'une table, les achats peuvent être de longueur variable. Pour introduire la méthode, nous considérons l'exemple suivant:

	produit A	produit B	produit C	produit D	produit E

achat 1	X			X	
achat 2	X	X	X		
achat 3	X				X
achat 4	X			X	X
achat 5		X		X	

Figure 3.5: Liste des Achats

A partir de ces données, si on recherche des associations entre deux produits, on construit le tableau de co-occurrence qui montre combien de fois deux produits ont été achetés ensemble:

	produit A	produit B	produit C	produit D	produit E
produit A	4	1	1	2	1
produit B	1	2	1	1	0
produit C	1	1	1	0	0
produit D	2	1	0	3	1
produit E	1	0	0	1	2

Figure 3.6: Tableau de co-occurrence

Un tel tableau permet de déterminer avec quelle fréquence deux produits se rencontrent dans un même achat. Par exemple, le produit A apparaît dans 80% des achats, le produit C n'apparaît jamais en même temps que le produit E, les produits A et D apparaissent simultanément dans 40% des achats. Ces observations peuvent suggérer une règle de la forme : "si un client achète le produit A alors il achète le produit D".

Il nous faut préciser comment extraire les règles et, pour cela, il nous faut quantifier leur pertinence. Considérons les règles:

1. si A alors B,
2. si A alors D,
3. si D alors A.

A et B apparaissent dans 20% des achats, A et D apparaissent dans 40% des achats, la règle 1 a un support de 20% et les règles 2 et 3 ont un support de 40%. Considérons les règles 2 et 3, c'est-à-dire les produits A et D. D apparaît dans trois achats et, dans ces trois achats, A apparaît deux fois. Par contre, A apparaît quatre fois et, lorsqu'il apparaît, D n'apparaît que deux fois. On définit alors la confiance d'une règle, la règle 3 a une confiance de 67% et la règle 2 a une confiance de 50%. On préfère donc la règle 3 : si D alors A.

Ces idées se généralisent à toutes les combinaisons d'un nombre quelconque d'articles. Par exemple, pour trois articles, on cherche à générer des règles de la forme si X et Y alors Z. Nous allons maintenant formaliser ces remarques introductives.

Description de la méthode

On suppose avoir défini une liste d'articles. On dispose en entrée d'une liste d'achats.

Définitions:

Une *règle d'association* est une règle de la forme : Si **condition** alors **résultat**. Dans la pratique, on se limite, en général, à des règles où la *condition est une conjonction d'apparition d'articles* et le *résultat est constitué d'un seul article*. Par exemple, une règle à trois articles sera de la forme : Si X et Y alors Z ; règle dont la sémantique peu être énoncée : Si les articles X et Y apparaissent simultanément dans un achat alors l'article Z apparaît.
Pour choisir une règle d'association, il nous faut définir les quantités numériques qui vont servir à valider l'intérêt d'une telle règle. Le *support* d'une règle est la fréquence d'apparition simultanée des articles qui apparaissent dans la condition et dans le résultat dans la liste d'achats donnée en entrée, soit:

Support = Fréq (condition et résultat) = d / m

Où *d* est le nombre d'achats où les articles des parties condition et résultat apparaissent et *m* le nombre total d'achats.
La *confiance* est le rapport entre le nombre d'achats où tous les articles figurant dans la règle apparaissent et le nombre d'achats où les articles de la partie condition apparaissent, soit:

Confiance = Fréq (condition et résultat) / Fréq (condition) = d / c

Où *c* est le nombre d'achats où les articles de la partie condition apparaissent. La confiance ne dépend que des articles qui apparaissent dans la règle. Des règles, dont le support est suffisant, ayant été choisies, la règle de confiance maximale sera alors privilégiée. Cependant, nous allons montrer sur l'exemple suivant que le support et la confiance ne sont pas toujours suffisants. Considérons trois articles A, B et C et leurs fréquences d'apparition :

article(s)	A	B	C	A et B	A et C	B et C	A et B et C
fréquence	45%	42,5%	40%	25%	20%	15%	5%

Si on considère les règles à trois articles, elles ont le même niveau de support 5%. Le niveau de confiance des trois règles est :

Règle	Confiance
si A et B alors C	0.20
si A et C alors B	0.25
si B et C alors A	0.33

La règle "Si B et C alors A" possède la plus grande confiance. Une confiance de 0.33 signifie que si B et C apparaissent simultanément dans un achat alors A y apparaît aussi avec une probabilité estimée de 33%. Mais, si on regarde le tableau des fréquences d'apparition des articles, on constate que A apparaît dans 45% des achats. Il vaut donc mieux prédire A sans autre information que de prédire A lorsque B et C apparaissent. C'est pourquoi il est intéressant

d'introduire l'*amélioration* qui permet de comparer le résultat de la prédiction en utilisant la règle avec la prédiction sans la règle. Elle est définie par :

Amélioration = Confiance / Fréq (Résultat)

Une règle est intéressante lorsque l'amélioration est supérieure à 1. Pour les règles choisies, on trouve :

règle	confiance	f(résultat)	amélioration
si A et B alors C	0.20	40%	0.50
si A et C alors B	0.25	42.5%	0.59
si B et C alors A	0.33	45%	0.74

Par contre, la règle si A alors B possède un support de 25%, une confiance de 0.55 et une amélioration de 1.31, cette règle est donc la meilleure. En règle générale, la meilleure règle est celle qui contient le moins d'articles.

Recherche des règles:

Une liste de *n* articles étant définie, on considère une liste de *m* achats. On procède comme suit :

1. On calcule le nombre d'occurrences de chaque article,
2. On calcule le tableau des co-occurrences pour les paires d'articles,
3. On détermine les règles de niveau 2 en utilisant les valeurs de support, confiance et amélioration,
4. On calcule le tableau des co-occurrences pour les triplets d'articles,
5. On détermine les règles de niveau 3 en utilisant les valeurs de support, confiance et amélioration,
6. ...

La valeur du nombre *m* d'achats est, en règle générale, très grande. Il est fréquent d'avoir à traiter des problèmes où *m* est de l'ordre du million. Des parcours de cette liste sont nécessaires pour construire les tableaux de co-occurrences d'où la nécessité d'architectures qui permettent des accès rapides à de grands jeux de données. Pour le nombre *n* d'articles, cette valeur peut également être grande. Étudions les tailles des tableaux en fonction de *n* et du nombre d'articles apparaissant dans les règles:

	2	3	4
n	$n(n-1)/2$	$n(n-1)(n-2)/6$	$n(n-1)(n-2)(n-3)/24$
100	4950	161 700	3 921 225
10 000	$\approx 5 \times 10^7$	$\approx 1.7 \times 10^{11}$	$\approx 4.2 \times 10^{14}$

On constate que les calculs nécessaires à la construction des règles (tableaux, support, confiance, amélioration) dépassent très vite les capacités de calcul des machines (malgré l'évolution continuelle de ces capacités).

Une technique de réduction du nombre des articles et de leurs combinaisons pris en considération à chaque étape est appelée *élagage par support minimum*.

On ne considère, à une étape donnée (par exemple recherche des règles à trois articles), que les règles faisant apparaître des articles pour lequel le support est supérieur à un taux fixé à priori. Par exemple, pour une liste d'achats de m = 1000 000 d'articles, si le support minimum est fixé à 2%, à l'étape 2 on ne considère que les règles de la forme si X alors Y où X et Y apparaissent simultanément dans 20 000 achats. L'élagage par support minimum permet d'éliminer les articles qui sont trop peu fréquents pour générer des règles intéressantes. En faisant varier le support minimum selon l'étape, on peut trouver des combinaisons rares d'articles fréquents (s'il diminue) ou des combinaisons fréquentes d'articles rares (s'il augmente).

Pour certaines applications, il est possible de limiter le nombre de combinaisons en limitant la forme des règles recherchées. Par exemple, la conclusion de la règle est restreinte à un sous-ensemble de l'ensemble des articles comme les articles nouvellement vendus.

Une dernière façon de limiter les calculs est la création de groupes d'articles. Ceci nécessite de déterminer le bon niveau pour les articles, ce qui ne peut être fait qu'avec des experts du domaine demandeur de l'étude. Considérons des produits de supermarché. On peut aller d'une description d'un article qui va de "conserve" au code barre de l'article en passant par "conserve de légumes", "conserve de légumes de telle marque", "conserve de légumes de telle taille", "conserve de légumes de telle marque et telle taille". Le choix du bon niveau de description est difficile et dépend du problème à résoudre.

Avantages et Inconvénients de la méthode

- **Résultats clairs**: Les règles d'association sont faciles à interpréter. Elles sont faciles à utiliser pour des utilisations concrètes.
- **Apprentissage non supervisé**: La méthode ne nécessite pas d'autre information qu'une classification en articles et la donnée d'une liste d'articles pour extraire les règles.
- **Achats de taille variable**: La méthode est l'une des rares méthodes qui prend en entrée des achats qui sont des listes d'articles de taille variable. En effet, la plupart des autres méthodes travaillent sur la base d'enregistrements d'une table, ces enregistrements ayant une longueur fixe.

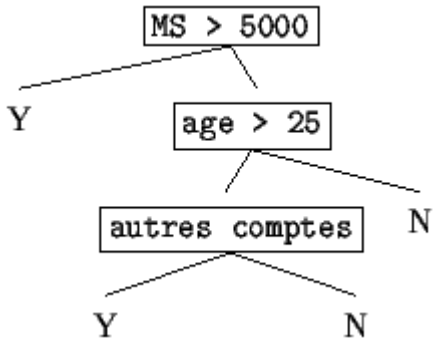
- **Introduction du temps:** Il est possible d'adapter la méthode pour traiter des séries temporelles pour générer des règles de la forme: "un client ayant acheté le produit A est susceptible d'acheter le produit B dans deux ans"; il est possible d'introduire des "articles virtuels" tels que le jour, la période ou la saison et limiter la forme des règles si on souhaite rechercher des comportements d'achat qui dépendent du temps.
- **Simplicité de la méthode:** La méthode et les calculs sont élémentaires (on ne calcule que des fréquences d'apparition). La méthode peut être programmée aisément sur tableur (pour des tailles de problème raisonnables) et est disponible dans la plupart des environnements de fouille de données.
- **Coût de la méthode:** Par contre, la méthode est coûteuse en temps de calcul. Le regroupement d'articles et la méthode du support minimum permettent de diminuer les calculs mais on peut alors éliminer malencontreusement des règles importantes.

- **Le choix des articles:** Il est difficile de déterminer le bon niveau d'articles. Les traitements préalables sur les achats peuvent être complexes : par exemple, il peut être difficile de retrouver l'article à partir de son code-barre enregistré sur le ticket de caisse.
- **Les articles rares:** La méthode est plus efficace pour les articles fréquents. Pour les articles rares, on peut restreindre la forme des règles choisies ou faire varier le support minimum.
- **La qualité des règles**: La méthode peut produire des règles triviales ou inutiles. Les règles triviales sont des règles évidentes (si camembert alors vin rouge) qui, par conséquent, n'apportent pas d'information. Les règles inutiles sont des règles difficiles à interpréter qui peuvent provenir de particularités propres à la liste des achats ayant servie à l'apprentissage.

d. Arbre de Décision

Nous étudions les algorithmes de génération d'arbres de décision à partir de données. Les deux algorithmes les plus connus et les plus utilisés (l'un ou l'autre ou les deux sont présents dans les environnements de fouille de données) sont CART (Classification And Regression Trees [*BFOS84*]) et C5 (version la plus récente après ID3 et C4.5 [*Qui93*]). Ces algorithmes sont très utilisés car performants et car ils génèrent des procédures de classification exprimables sous forme de règles.

Un arbre de décision est une représentation graphique d'une procédure de classification. Les nœuds internes de l'arbre sont des tests sur les champs ou attributs, les feuilles sont les classes. Lorsque les tests sont binaires, le fils gauche correspond à une réponse positive au test et le fils droit à une réponse négative. Ci-dessous L'exemple d'un arbre de décision où MS est la moyenne des soldes du compte courant, autres comptes est un champ binaire qui vaut oui si le client dispose d'autres comptes, la classe Y indique un a priori favorable pour l'attribution d'un prêt.



Pour classer un enregistrement, il suffit de descendre dans l'arbre selon les réponses aux différents tests pour l'enregistrement considéré. Soit l'enregistrement défini par:

Nom = Digra
Prénom = Omar
Age = 32
Csp = 2
MS = 2550
Prêts en cours = oui

Autres comptes = oui.

Cet enregistrement sera classé Y car:

MS <= 2550, âge > 25 et *autres comptes = oui*.

On peut déjà remarquer quelques propriétés importantes des arbres de décision:

- La procédure de classification associée est compréhensible par tout utilisateur,
- La classe associée à un enregistrement particulier peut être justifiée,
- Les attributs apparaissant dans l'arbre sont les attributs pertinents pour le problème de classification considéré.

On peut également remarquer qu'un arbre de décision est un système de règles. Il est donc possible de transformer l'arbre de l'exemple précédent en :

Si `MS > 5000`	alors Y
Si (`MS<=5000`) et (`age>25`) et (`autres comptes=oui`)	alors Y
Si (`MS<=5000`) et (`age>25`) et (`autres comptes=non`)	alors N
Si (`MS<=5000`) et (`age<=25`)	alors N

Les systèmes de règles construits sont particuliers. En effet, pour tout enregistrement une et une seule règle s'applique, c'est-à-dire que les règles sont exhaustives et mutuellement exclusives.

Apprentissage des arbres de décision

Avec en entrée, un échantillon de m enregistrements classés (x□,c(x□)), un algorithme d'apprentissage doit fournir en sortie un arbre de décision. La plupart des algorithmes procèdent de façon descendante, c'est-à-dire qu'ils choisissent la racine de l'arbre (en général un test) puis, récursivement, choisissent l'étiquette des fils. Pour simplifier la présentation, nous nous limitons, dans cette section à des problèmes où les attributs sont discrets et le nombre de classes égal à 2. L'algorithme générique peut s'écrire :

Algorithme d'apprentissage par arbres de décision

donnée : un échantillon S de m enregistrements classés $(\vec{x},c(\vec{x}))$
initialisation: arbre vide, nœud courant: racine, échantillon courant: S
répéter
décider si le nœud courant est terminal
si le nœud courant est terminal **alors**
étiqueter le nœud courant par une feuille
sinon
sélectionner un test et créer le sous-arbre
finsi
nœud courant : un nœud non encore étudié
échantillon courant : échantillon atteignant le nœud courant
jusque production d'un arbre de décision
élaguer l'arbre de décision obtenu
sortie : arbre de décision élagué

Nous allons préciser les différents points de cet algorithme en mettant en évidence les particularités des différents algorithmes.

Décider si le nœud courant est terminal : le nœud courant est terminal si:

- Il n'y a plus d'attributs disponibles, c'est-à-dire que sur le chemin menant de la racine au nœud courant tous les tests disponibles ont été utilisés ;
- Tous les exemples de l'échantillon courant sont dans une même classe.

Les critères précédents sont indiscutables, les autres critères sont spécifiques aux différents algorithmes et sont souvent paramétrables. Des exemples de critères sont :

- la proportion d'exemples d'une classe est supérieure à un seuil prédéfini. Par exemple, on décide de l'arrêt si une des classes contient plus de 95% des exemples.
- C5 utilise le critère suivant : s'il n'existe pas de test ayant au moins k éléments sur deux branches alors le nœud est terminal. L'objectif de ce critère est d'éviter une croissance trop grande de l'arbre par l'exploration de branches comprenant trop peu d'exemples. La valeur de k est, par défaut, égale à 2, elle peut être modifiée par l'utilisateur.

Étiqueter le nœud courant par une feuille

On étiquette le nœud courant par la classe majoritaire. Par exemple, si le nœud courant est terminal et s'il y a 5 exemples de classe 0 et 20 exemples de classe 1, on étiquette par 1. Cependant, pour certains problèmes, il se peut que les erreurs de classification d'une classe vers l'autre aient des conséquences différentes. C'est le cas, par exemple, pour un diagnostic médical pour lequel classer un individu malade comme sain ou classer un individu sain comme malade n'ont pas les mêmes conséquences. Dans ce cas, il est possible de définir des coûts de mauvaise classification et la classe choisie le sera en fonction des coûts attribués.

Sélectionner un test

On suppose que le nœud courant n'est pas terminal. Soit S l'échantillon associé au nœud courant. Pour introduire les possibles critères de sélection du test, considérons l'exemple suivant : S contient 100 exemples, 60 de classe 0 et 40 de classe 1. Le nœud courant sera étiqueté par le couple (60,40). Supposons que deux tests soient disponibles, et que ces deux tests déterminent les répartitions suivantes:

(60,40) $\rightarrow$ *A* (30,10) (30,5) (0,25)

(60,40) $\rightarrow$ *B* (40,20) (20,20)

Pour choisir le test, on utilise des fonctions qui mesurent le "degré de mélange" des différentes classes. Pour les problèmes à deux classes, on peut utiliser une des fonctions suivantes :

- La fonction de Gini : Gini(x) = 4x (1-x)
- La fonction entropie : Entropie(x) = -x log x - (1-x) log (1-x)

Où x désigne la proportion d'éléments dans l'une des deux classes. Ces deux fonctions sont à valeurs dans l'intervalle réel [0,1], prennent leur minimum pour x = 0 ou x = 1 (tous les exemples sont dans une même classe) et leur maximum lorsque x = 1/2 (les exemples sont également répartis entre les deux classes). Choisissons, par exemple, la fonction de Gini. Pour le nœud courant, x = 60/100 et Gini(x) = 4 × 60/100 × 40/100 = 0.96. Si on choisit le test A, pour le premier fils (le plus à gauche), x = 3/4 et Gini(x) = 0.75, pour le second fils x=6/7 et Gini(x)=0.49, pour le troisième fils, Gini(x) = 0. Pour comparer les trois tests, on estime le "degré de mélange espéré" en pondérant les degrés de mélange des fils par la proportion des exemples allant sur ce fils, on obtient:

- Pour A : 40/100 × 0.75 + 35/100 × 0.49 + 25/100 × 0 = 0.47
- Pour B : 60/100 × 0.89 + 40/100 × 1 = 0.93

On choisit alors le test qui fournit de degré de mélange espéré minimum. Souvent, on introduit le Gain qui est égal au degré de mélange du noeud courant diminué du degré de mélange espéré par l'introduction du test, on choisit alors le test qui apporte le gain maximal.

Élaguer l'arbre de décision obtenu

Il est possible de poursuivre la croissance de l'arbre jusqu'à obtention d'un arbre d'erreur nulle (si c'est possible : si il n'existe pas d'exemples ayant la même description mais des classes différentes) ou d'un arbre d'erreur mesurée sur l'ensemble d'apprentissage la plus petite possible. Cependant, l'objectif d'une procédure de classification est de bien classer des exemples non encore rencontrés, on parle de pouvoir de généralisation. Si l'algorithme fournit en sortie un arbre très grand qui classe bien l'échantillon d'apprentissage, on se trouve confronté au problème de sur-spécialisation : on a appris "par cœur" l'ensemble d'apprentissage, mais on n'est pas capable de généraliser. L'objectif de la phase d'élagage est d'obtenir un arbre plus petit (on élague des branches, c'est-à-dire que l'on détruit des sous-arbres) dans le but d'obtenir un arbre ayant un meilleur pouvoir de généralisation (même si on fait augmenter l'erreur sur l'ensemble d'apprentissage).

Les algorithmes d'apprentissage

CART

CART a été défini dans les années 80 ([*BFOS84*]). Depuis, il a été intégré à de nombreux environnements de fouille de données sous de nombreuses variantes. Nous donnons ici ses principales particularités. A l'origine, l'algorithme ne considérait que des tests binaires. La fonction qui mesure le degré de mélange (et donc le gain) est la fonction de Gini (les versions diffusées proposent d'autres choix). Pour l'élagage, on effectue un parcours ascendant de l'arbre construit. Pour décider si un sous arbre peut être élagué, on compare l'erreur réelle estimée de l'arbre courant avec l'arbre élagué.

L'estimation de l'erreur réelle est mesurée sur un ensemble test ou par validation croisée.

C5

C5 est la version la plus récente d'un algorithme ID3 développé par R. Quinlan [*Qui93*]. L'algorithme peut prendre en compte des attributs d'arité quelconque. La fonction qui mesure le degré de mélange (et donc le gain) est la fonction entropie. Cette fonction a tendance à privilégier les attributs possédant un grand nombre de valeurs. Pour éviter ce biais, une fonction gain d'information est également disponible. L'élagage est effectué avec l'ensemble d'apprentissage par une évaluation pessimiste de l'erreur. Bien que cette technique puisse sembler inadaptée, elle donne de bons résultats en pratique.

Extensions

Bien que les algorithmes précédents semblent spécialisés pour les attributs discrets, ils ont été adaptés pour considérer également les attributs continus. Par exemple, dans C5, soit A un attribut continu, pour sélectionner un test, l'algorithme fait participer à la compétition tous les tests de la forme $A>a$ où a est une valeur prise par l'attribut A dans l'ensemble d'apprentissage.

Les algorithmes peuvent traiter les valeurs manquantes (descriptions contenant des champs non renseignés) pour l'apprentissage, mais aussi pour la classification.
Pour les attributs discrets possédant plusieurs valeurs, il est possible de demander à l'algorithme de grouper les valeurs pour le choix des tests. Par exemple, si un attribut A prend ses valeurs dans l'ensemble {1,2,3,4,5}, l'arbre engendré pourra contenir des tests de la forme $A \in \{1,3,5\}$.

C5 propose également de générer un système de règles à partir de l'arbre de décision. Le système obtenu n'est pas une simple réécriture de l'arbre car des transformations et simplifications sont effectuées.

Avantages et Inconvénient de la Méthode

- **Lisibilité du résultat:** Un arbre de décision est facile à interpréter et est la représentation graphique d'un ensemble de règles. Si la taille de l'arbre est importante, il est difficile d'appréhender l'arbre dans sa globalité. Cependant, les outils actuels permettent une navigation aisée dans l'arbre (parcourir une branche, développer un nœud, élaguer une branche) et, le plus important, est certainement de pouvoir expliquer comment est classé un exemple par l'arbre,

ce qui peut être fait en montrant le chemin de la racine à la feuille pour l'exemple courant.

- **Tout type de données:** L'algorithme peut prendre en compte tous les types d'attributs et les valeurs manquantes. Il est robuste au bruit.
- **Sélection des variables:** L'arbre contient les attributs utiles pour la classification. L'algorithme peut donc être utilisé comme pré-traitement qui permet de sélectionner l'ensemble des variables pertinentes pour ensuite appliquer une autre méthode.
- **Classification efficace:** L'attribution d'une classe à un exemple à l'aide d'un arbre de décision est un processus très efficace (parcours d'un chemin dans un arbre).
- **Outil disponible:** Les algorithmes de génération d'arbres de décision sont disponibles dans tous les environnements de fouille de données.
- **Extensions et modifications:** La méthode peut être adaptée pour résoudre des tâches d'estimation et de prédiction. Des améliorations des performances des algorithmes de base sont possibles grâce aux techniques de bagging et de boosting : on génère un ensemble d'arbres qui votent pour attribuer la classe.
- **Sensible au nombre de classes:** Les performances tendent à se dégrader lorsque le nombre de classes devient trop important.
- **Évolutivité dans le temps:** L'algorithme n'est pas incrémental, c'est-à-dire, que si les données évoluent avec le temps, il est nécessaire de relancer une phase d'apprentissage sur l'échantillon complet (anciens exemples et nouveaux exemples).

e. Algorithme Génétique

C'est en 1860 que Charles Darwin publie son livre intitulé "L'origine des espèces au moyen de la sélection naturelle ou la lutte pour l'existence dans la nature. Dans ce livre, Darwin rejette l'existence «de systèmes naturels figés», déjà adaptés pour toujours à toutes les conditions extérieures, et expose sa théorie de l'évolution des espèces : sous l'influence des contraintes extérieurs, les êtres vivants se sont graduellement adaptés à leur milieu naturel au travers de processus de reproductions.

Les algorithmes génétiques (AGs) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Leur fonctionnement est extrêmement simple. On part avec une population de solutions potentielles (chromosomes) initiales arbitrairement choisies. On évalue leur performance (fitness) relative. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. On recommence ce cycle jusqu'à ce que l'on trouve une solution satisfaisante.

C'est à partir du 20ème siècle que la mutation génétique a été mise en évidence. Les problèmes de traitement de l'information sont résolus de manières figés : lors de sa phase de conception, le système reçoit toutes les caractéristiques nécessaires pour les conditions d'exploitation connues au moment de sa conception, ce qui empêche une adaptation à des conditions d'environnement inconnues, variables ou évolutives. Les chercheurs en informatique étudient donc des méthodes pour

permettre aux systèmes d'évoluer spontanément en fonction de nouvelles conditions: c'est l'émergence de la programmation évolutionnaire.

Leur fonctionnement est extrêmement simple. On part avec une population de solutions potentielles (chromosomes) initiales arbitrairement choisies. On évalue leur performance (fitness) relative. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. On recommence ce cycle jusqu'à ce que l'on trouve une solution satisfaisante.

Pour résumer, Lerman et Ngouenet (1995) distinguent 4 principaux points qui font la différence fondamentale entre ces algorithmes et les autres méthodes:

1. Les algorithmes génétiques utilisent un codage des paramètres, et non les paramètres eux mêmes.
2. Les algorithmes génétiques travaillent sur une population de points, au lieu d'un point unique.
3. Les algorithmes génétiques n'utilisent que les valeurs de la fonction étudiée, pas sa dérivée, ou une autre connaissance auxiliaire.
4. Les algorithmes génétiques utilisent des règles de transition probabilistes, et non déterministes.

Présentation des algorithmes génétiques (AGs)

Selon Lerman et Ngouenet (1995) un algorithme génétique est défini par:

- Individu/chromosome/séquence: : une solution potentielle du problème,
- Population: un ensemble de chromosomes ou de points de l'espace de recherche,
- Environnement: l'espace de recherche,
- Fonction de fitness: la fonction - positive - que nous cherchons à maximiser.

Définition 1: (Séquence/Chromosome/Individu (Codage binaire)).

Un chromosome est donc une suite de bits en codage binaire, appelé aussi chaîne binaire. Dans le cas d'un codage non binaire, tel que le codage réel, la suite A ne contient qu'un point.

Définition 2 (Fitness d'une séquence).

Nous appelons *fitness* d'une séquence toute valeur positive notée f(A), où f est typiquement appelée fonction de *fitness*.

La *fitness* (efficacité) est donc donnée par une fonction à valeurs positives réelles. Dans le cas d'un codage binaire, nous utiliserons souvent une fonction de décodage d qui permettra de passer d'une chaîne binaire à un chiffre à valeur réelle. La fonction de *fitness* est alors choisie telle qu'elle transforme cette valeur en valeur positive.

Le but d'un algorithme génétique est alors simplement de trouver la chaîne qui maximise cette fonction f. Bien évidemment, chaque problème particulier nécessitera ses propres fonctions d et f.

Les AGs sont alors basés sur les phases suivantes :

1. Initialisation: Une population initiale de N chromosomes est tirée aléatoirement.
2. Évaluation. Chaque chromosome est décodé, puis évalué.

3. Sélection. Création d'une nouvelle population de N chromosomes par l'utilisation d'une méthode de sélection appropriée.
4. Reproduction. Possibilité de croisement et mutation au sein de la nouvelle population.
5. Retour à la phase d'évaluation jusqu'à l'arrêt de l'algorithme.

Exemple de Génération de Règles d'Association:

1. A C D
2. A B C E
3. B C E
4. B E
5. A B C E
6. B C E

	A	B	C	D	E
1	1	0	1	1	0
2	1	1	1	0	1
3	0	1	1	0	1
4	0	1	0	0	1
5	1	1	1	0	1
6	0	1	1	0	1

Il existe 32 itemsets

0-itemset : {} (trivial)

1-itemsets : {A}, {B}, {C}, {D}, {E}

2-itemsets : {AB}, {AC}, {AD}, {AE}, {BC}, {BD}, {BE}, {CD}, {CE}, {DE}

3-itemsets : {ABC}, {ABD}, {ABE}, {ACD}, {ACE}, {ADE}, {BCD}, {BCE}, {BDE}, {CDE}

4-itemsets : {ABCD}, {ABCE}, {ABDE}, {ACDE}, {BCDE}

5-itemsets : {ABCDE}

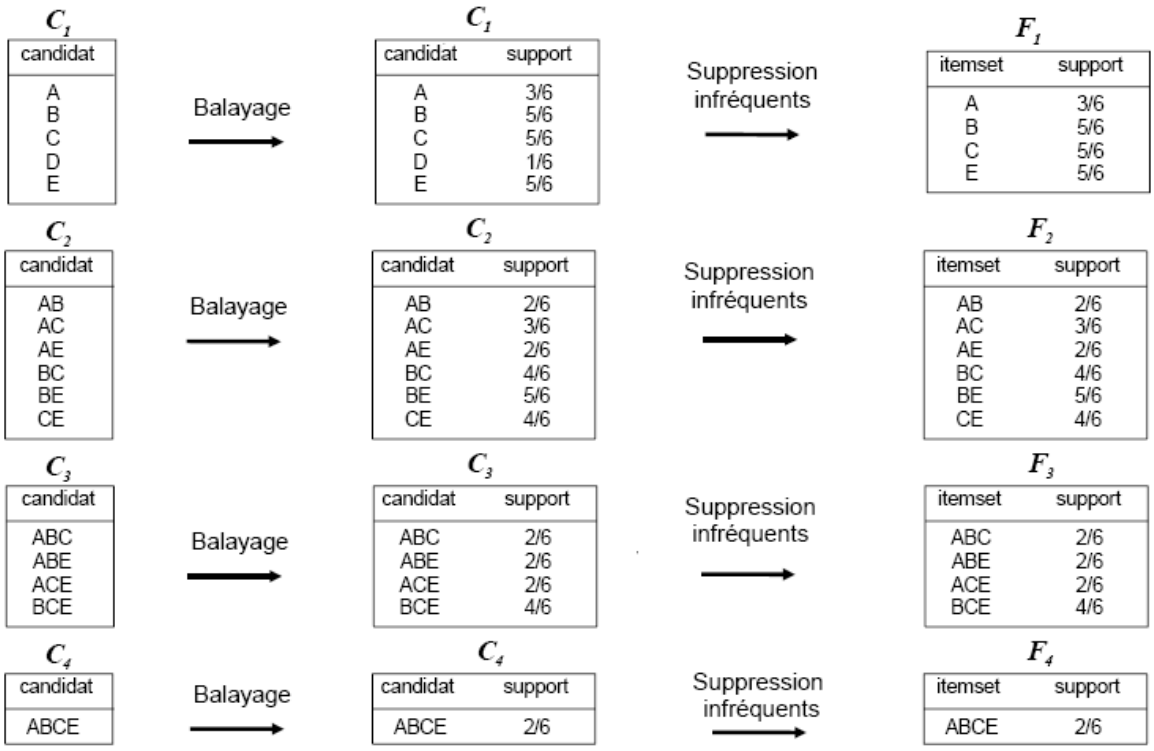
- Support d'un itemset
  - Proportion d'objets contenant l'itemset
  - Ex : support({BC}) = |{2,3,5,6}| / |**O**| = 4/6 /* **O étant la base de transactions */**
    support({BE}) = |{2,3,4,5,6}| / |**O**| = 5/6
- Support d'une règle d'association
  - Support de l'union de l'antécédent et la conséquence
  - Ex: r = BC → E

support(r) = support({BCE}) = |{2,3,5,6}| / |**O**| = 4/6

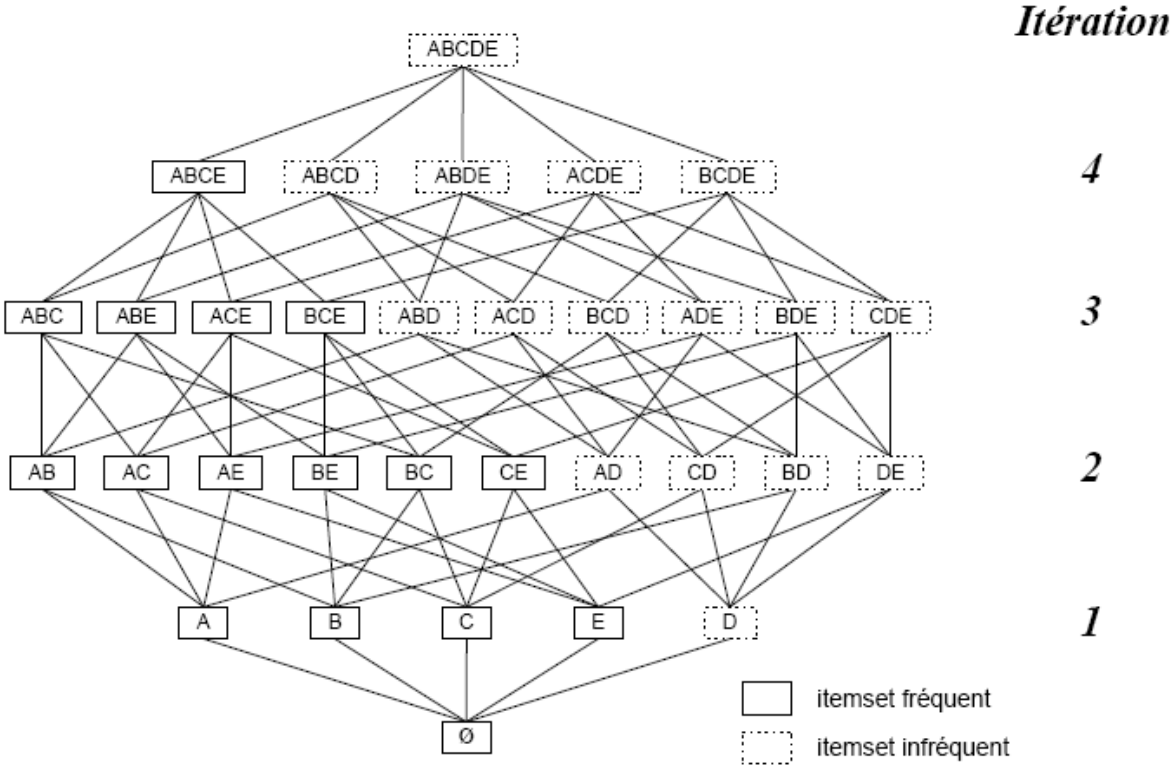
- Confiance d'une règle d'association
  - Proportion d'objets vérifiant l'implication
  - Exemple : confiance(r) = support({BCE}) / support({BC}) = 1

- Règles d'association valides
  - support ≥ *minsup*
  - Règles concernant une population suffisamment importante
  - confiance ≥ *minconf*
  - Statistiquement significatives
- Seuils *minsup* et *minconf* définis selon
  - La nature des données analysées
  - L'objectif de l'analyse
- Règles d'association générées à partir des itemsets fréquents (support ≥ *minsup*)
  - Exemple :
    - r = {BC → E, support(r), confiance(r)}
    - Valide si support(r) = support(BCE) ≥ minsup = 2/6
    - Valide si confiance(r) = support(BCE)/support(BC) ≥ minconf = 2/3
    - Générée à partir de {BC} et {BCE}
  - Approche initiale : extraction des itemsets fréquents
    2. Extraction des itemsets fréquents avec support
    3. Génération des règles valides avec support et confiance
    4. Itemsets fréquents : {ABC, ABD, ACD, ACE, BCD}
    5. Jointure des itemsets fréquents de même préfixe
       - ABC et ABD : ABCD
       - ACD et ACE : ACDE
  - Élagage des candidats inutiles
       - ABCD conservé : ABC, ABD, ACD fréquents
       - ACDE supprimé : ADE infréquent
  - 4-itemsets candidats : {ABCD}

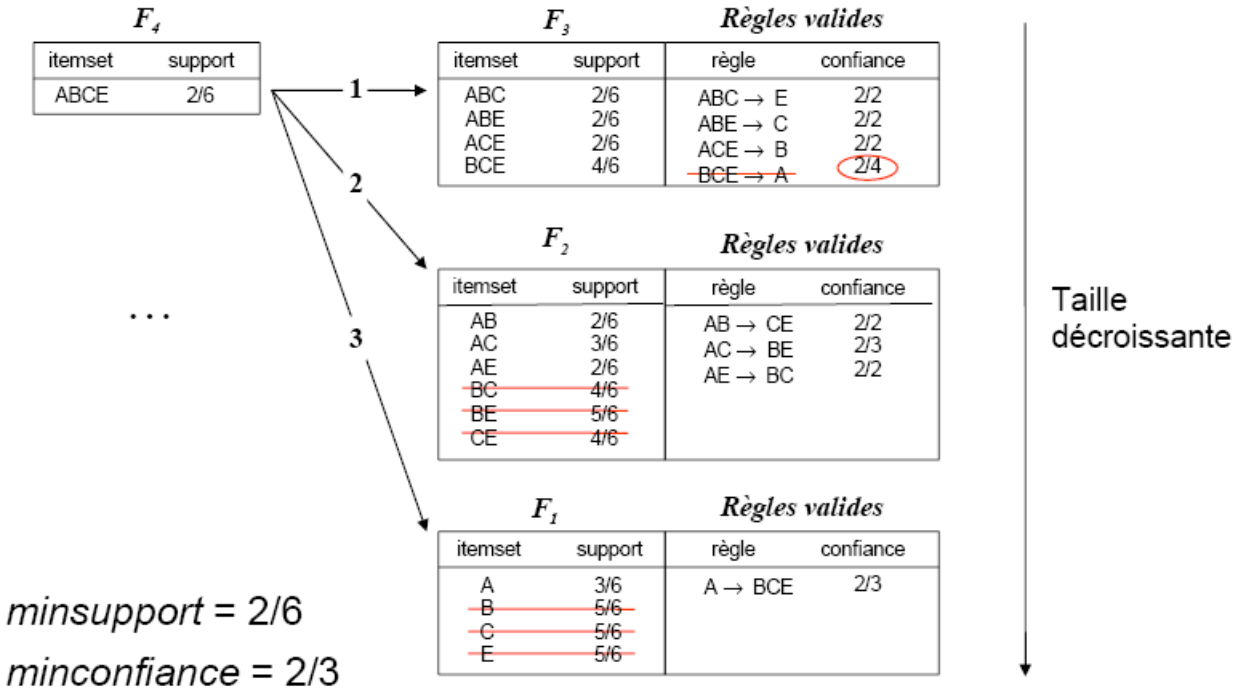
Exemple : extraction des itemsets fréquents



Extraction des itemsets fréquents



Exemple : génération des règles



F3			F2			
Itemset	Support		Itemset	Support	Règles	Confiance
ABC	2/6		AB	2/6	AB →C	1
ABE	2/6		AC	3/6	AB →E	2/3
ACE	2/6		AE	2/6	AC →B	2/3
BCE	4/6		BC	4/6	AC →E	1
			BE	5/6	AE →B	1
			CE	4/6	~~AE →C~~	~~2/5~~
					~~BC →A~~	~~2/4~~
					BC →E	1
					~~BE →A~~	~~2/5~~
					BE →C	4/5
					~~CE →A~~	~~2/4~~
					CE →B	1
			F1			
			Itemset	Support	Règles	Confiance
			A	3/6	A→BC	2/3
			B	5/6	A→BE	2/3
			C	5/6	A→CE	2/3
			E	5/6	~~B→AC~~	~~2/5~~
					~~B→AE~~	~~2/5~~
					B→CE	4/5
					~~C→AB~~	~~3/5~~
					~~C→AE~~	~~2/5~~
					C→BE	4/5
					~~E→AB~~	~~2/5~~
					~~E→AC~~	~~2/5~~
					E→BC	4/5

F2			F1			
Itemset	Support		Itemset	Support	Règles	Confiance
AB	2/6		A	3/6	A→B	1
AC	3/6		B	5/6	A→C	1
AE	2/6		C	5/6	A→E	2/3
BC	4/6		E	5/6	~~B→A~~	~~2/5~~
BE	5/6				B→C	4/5
CE	4/6				B→E	1
					~~C→A~~	~~3/5~~
					C→B	1
					C→E	4/5
					~~E→A~~	~~2/5~~
					E→B	1
					E→C	4/5

Extraction des règles d'associations :

1. Définitions :

Les articles traitants les règles d'association utilisent les termes *item* pour désigner un élément ou un article, et *itemset* pour désigner un ensemble d'éléments ou d'articles.

Notations :

- B la base de données, $|B|$ le nombre de transactions total de la base.
- $I = \{A, B, \ldots\}$ l'ensemble des items, et m son cardinal (Nombre d'éléments).

Définition 1 :

Un k-Itemset est un Itemset contenant k items.

Définition 2 :

Une règle d'association est de la forme $X \rightarrow Y$, où $X \subseteq I$, $Y \subseteq I$, et $X \cap Y = \varnothing$, où X et Y sont des itemsets. Elle est évaluée par un support (aussi appelée "fréquence") et une confiance.

Définition 3 : support d'un Itemset.

Soit X un itemset, son support est le nombre de transactions de la base B contenant X divisé par le nombre total de transactions.
Support (X) = $\left| \{t \in B / X \subseteq t\} \right| / \left| B \right|$
Pour simplifier, nous pouvons dire que le support de l'itemset X est le nombre de transactions de la base B contenant X.

Définition 4 : support d'une règle d'association

Le support d'une règle $X \rightarrow Y$ est le rapport entre le nombre de transactions de B contenant $X \cup Y$, et le nombre total de transactions.
Support ($X \rightarrow Y$) = $\left| \{t \in B / X \cup Y \subseteq t\} \right| / \left| B \right|$

Définition 5 : confiance d'une règle d'association

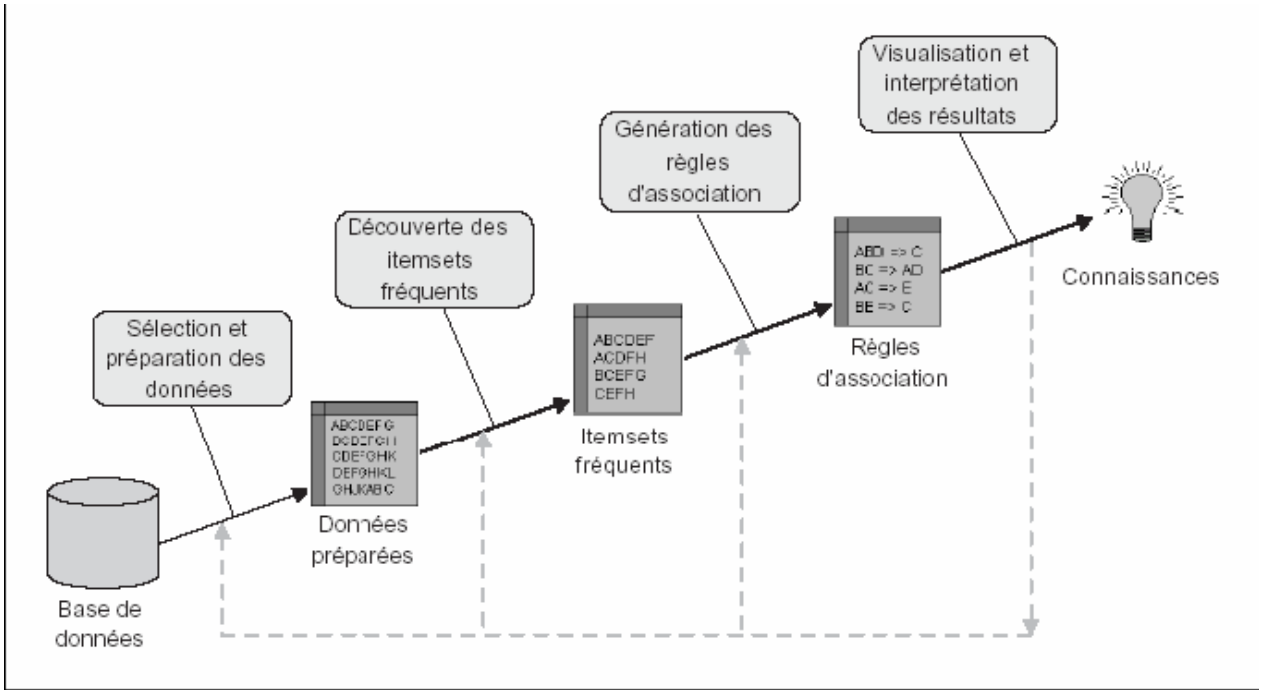
La confiance d'une règle est le rapport entre le nombre de transactions de B contenant $X \cup Y$, et le nombre de transactions de B contenant X.
Confiance ($X \rightarrow Y$) = $\left| \{t \in B / X \cup Y \subseteq t\} \right| / \left| \{t \in B / X \subseteq t\} \right|$
On voit immédiatement que la confiance se définit aussi par un rapport de support:
Confiance ($X \rightarrow Y$) = Support ($X \cup Y$) / Support (X)

Définition 6 : itemset fréquent

Soit X un itemset, on dit que X est fréquent si support (X) ≥ *minsup*.

2. Etapes de l'extraction des règles d'associations :

L'extraction des règles d'association peut être décomposée en quatre étapes. La figure suivante résume ce processus:



2.1. Sélection et préparation des données :

Cette étape permet de préparer les données afin de leur appliquer les algorithmes d'extraction des règles d'association. Elle est constituée de deux phases :

- La sélection des données de la base qui permettront d'extraire les informations intéressant l'utilisateur. Ainsi la taille des données traitées est réduite ce qui assure une meilleure efficacité de l'extraction.
- La transformation de ces données en un contexte d'extraction (il s'agit d'un triplet constitué d'un ensemble d'objets, d'un ensemble d'itemsets et d'une relation binaire entre les deux). La transformation des données sélectionnées en données binaires améliore l'efficacité de l'extraction et la pertinence des règles d'association extraites.

2.2. Extraction des itemsets fréquents :

C'est l'étape la plus coûteuse en terme de temps d'exécution car le nombre d'itemsets fréquents dépend exponentiellement du nombre d'items manipulés (pour n items, on a 2 puissance n itemsets potentiellement fréquents).

2.3. Génération des règles d'association :

A partir de l'ensemble des itemsets fréquents pour un seuil minimal de support minsup, la génération des règles d'association pour un seuil de confiance minconf est un problème qui dépend exponentiellement de la taille de l'ensemble des itemsets fréquents.

2.4. Visualisation et interprétation des règles d'association :

Elle met entre les mains de l'utilisateur un ensemble de déductions fiables qui peuvent l'aider à prendre des décisions. Il faut que l'outil de visualisation prenne en compte la priorité des règles les unes par rapport aux autres, ainsi que les critères définis par l'utilisateur. De plus, il doit présenter les règles sous une forme claire et compréhensible.

3. Exemple :

Supposant qu'une base de données d'un supermarché contient les transactions suivantes :

Transaction 1: Lait Fromage Biscuits

Transaction 2: Pain Fromage Chocolat

Transaction 3: Lait Pain Fromage Chocolat

Transaction 4: Pain Chocolat

Transaction 5:Lait Pain Fromage Chocolat

Transaction 6: Pain Fromage Chocolat

Pour faciliter le traitement des données, les items seront remplacés par des entiers. On ordonnera les items de chaque transaction pour qu'on puisse leur appliquer l'algorithme de recherche des itemsets fréquents APRIORI. On obtient le tableau suivant pour notre exemple:

Transaction 1: 1 3 4
Transaction 1: 2 3 5
Transaction 1: 1 2 3 5 Où: 1: Lait 2: Pain 3: Fromage 4: Biscuit 5: Chocolat
Transaction 1: 2 5
Transaction 1: 1 2 3 5
Transaction 1: 2 3 5

Ainsi, la première étape est terminée. On utilisera l'algorithme APRIORI pour déterminer les itemsets fréquents.

On choisit *minsup* = 4. On obtient un seul itemset fréquent : (2 3 5) de support égal à 4.

Parmi les règles qu'on peut générer :

2, 3 → 5 avec une confiance = 4/ 4, c'est-à-dire: celui qui achète du pain et du fromage achèterait à 100% du chocolat,

3 → 2, 5 avec une confiance = 4/5, c'est à dire celui qui achète du fromage achèterait à 80% du pain et chocolat.

4. Algorithme APRIORI :

4.1. Propriétés et propositions :

L'algorithme APRIORI s'explique par les propriétés et propositions suivantes:

Propriété 1 :

Les itemsets construits à partir d'itemsets non fréquents sont non fréquents.

Propriété 2 :

Un sous itemset d'un itemset fréquent est fréquent.

Proposition 1 :

Un k-itemset Z candidat est l'union de deux (k-1)-itemsets fréquents X et Y.

Proposition 2 :

Si les itemsets sont triés et si Z est un k-itemset candidat, alors il existe deux (k-1)- itemsets X et Y partageant leurs (k-2) premiers items tel que Z = XUY.

4.2. Génération des itemsets fréquents : Algorithme APRIORI

La stratégie de cet algorithme est simple. Il cherche les itemsets potentiellement fréquents puis il ne garde que ceux qui le sont vraiment. Il détermine les itemsets fréquents dans un ordre croissant de taille, en se basant sur les propriétés et propositions citées précédemment.

NOTATIONS :

B : Ensemble des transactions.
L_k : Ensemble des itemsets fréquents de taille k.
C_k : Ensemble des itemsets candidats de taille k.
k-itemsets : les itemsets de taille k.

Algorithme APRIORI
Début
L_1 = {1-itemsets fréquents)
k=2 ;
Tant que L_{k-1} non vide **faire**
C_k=Apriori-Gen (L_k) ;
Pour chaque t de B **faire**
C_t = Subset (C_k, t) ; {les candidats contenus dans C_k}
Pour chaque c de C_t **faire**
c.count++;
Fin pour
Fin pour
L_k = {c de C_t / c.count _ *minsup*};
k++ ;
Fin du tant que
Return UL_k ;
Fin

A chaque itemset, on associe un compteur noté «*count* », initialisé à 0, pour stocker le support. Les 1-itemsets sont déterminés lors de la première passe sur les données en comptant les nombres d'apparitions de chaque item. La $k^{ième}$ passe est composée de trois phases :

a) Les (k-1)- itemsets fréquents trouvés lors de la $(k-1)^{ième}$ passe sont utilisés pour générer les itemsets candidats de taille k (algorithme APRIORI-GEN).

b) La base de données est parcourue et la fréquence des candidats est comptée, en utilisant la fonction Subset qui fournit l'ensemble des itemsets C_k contenus dans une transaction.

c) On sélectionne les itemsets qui ont une fréquence supérieure ou égale au seuil *minsup*.

4.3. Génération des itemsets candidats : Algorithme APRIORI-GEN

Cet algorithme construit à partir de l'ensemble des (k-1)-itemsets fréquents L_{k-1}, l'ensemble des k- itemsets candidats C_k. Il se déroule en deux étapes :

a) Il fusionne deux (k-1)-itemsets P et Q qui partagent leur (k-2)-premiers items.

b) Il supprime de C_k tout itemset X pour lequel au moins un sous-ensemble de longueur k-1 de X n'appartient pas à Lk-1.

Algorithme APRIORI-GEN
$C_k = \{ \}$
Pour chaque X de L_{k-1} **faire**
  Pour chaque Y de L_{k-1}, avec X < Y, X et Y partagent leur k-2 premiers items **faire**
    Pour chaque $Z \subset XUZ$ tel que : Z = k-1 **faire**
      Si $Z \notin L_{k-1}$ alors continuer avec Y suivant ;
    Fin pour
    Ajouter (XUZ) à C_k ;
  Fin pour
Fin pour
Return C_k

5. Génération de règles d'associations :

Les règles d'association que l'on considère ici ne se limitent pas aux règles dont les conséquences sont constituées d'un seul item. Pour générer les règles on prend en compte pour chaque itemset fréquent *f* tous les sous-ensembles non vides de *f*. Pour chacun de ces sous-ensembles h, on renvoie une règle de la forme *h => (f-h)* si le rapport *support (f) / support (h)* vaut au moins un seuil minimal de confiance *minconf*. Nous considérons tous les sous-ensembles de f pour générer les règles dont les conséquences sont des itemsets de taille supérieure à un.

5.1. Algorithme Gen-Règles :

Propriété :

Etant donné un itemset f, le support d'un sous-ensemble k de f est supérieur ou égal au support de f. Cette propriété permet de diminuer le nombre de règles d'association testées par l'algorithme.

Exemple:

Soit (1 2 5 8) un itemset fréquent et supposons que la règle d'association :
(1 5) → (2 8) n'est pas valide.

Selon la propriété, tout itemset fréquent inclus dans (1 5) aura un confiance inférieure ou égale à la confiance de (1 5), on peut alors déduire directement que les règles : 1 → (2 5 8) et 5 → (1 2 8) ne sont pas valides non plus. Ce qui nous évite de calculer les confiances de ces deux dernières règles.
Réciproquement, si la règle (2 8) → (1 5) est valide, alors:
Les règles (1 2 8) → 5 et (2 5 8) → 1 sont également valides.

L'algorithme Gen_Règles choisit successivement chaque itemset fréquent f_k de taille supérieure à un. Il génère à partir de cette itemset un ensemble H1 constitué des itemsets de taille 1 et qui sont des sous-ensembles de f_k. Pour chacun des itemset h1 de H, on calcule la confiance de la règle : (f_k-h1) → h1. Si sa confiance est supérieure ou égale à *minconf* (seuil minimal de confiance), on ajoute la règle à ER (l'ensemble de règles d 'association), sinon on supprime l'itemset h1 de H1. H1 contient alors la liste des 1-itemsets qui sont les conséquences des règles valides générées à partir de f_k. Après avoir construit l'ensemble H1, on fait appel à Gen_Rules.

Algorithme Gen-Rules

Entrée : k-itemsets fréquents f_k ; ensemble H_m de m-itemsets conséquences de règles valides générées à partir de f_k ; seuil minimal de confiance *minconf* ;

Sortie : ensemble ER de règles d'association valides augmenté des règles valides générées à partir de f_k dont la conséquence est un (m+1)-itemset ;