

Support de Cours INF 2333



UFR Sciences Et Technologies (SET)

Département Informatique

Classe : Licence 2 Informatique

Présentation générale

- **Unité d'enseignement : INFORMATIQUE INF 233**
 - **Elément constitutif :**
 - sigle : INF 2333**
 - intitulé : Langage et Modélisation de bases de données**
 - **Autres éléments constitutifs de l'UE :**
 - INF 2331 Structures de données
 - INF 2332 Analyse et Conception de systèmes d'information
 - **Crédits de l'UE : 12**
 - **Coefficient : 4**

Présentation générale

- Charge horaire VHT = 80 h
- Clé de répartition du volume horaire total :
CM : 20h
TP/TP : 20h
TPE : 40h
- Chargé de CM : Prof. Papa DIOP
- Travaux dirigés et pratiques : Prof. Papa DIOP

Objectif du cours

À l'issue du cours, l'apprenant doit être capable :

- d'**analyser des systèmes d'information de gestion** ;
- de **représenter de manière intelligible une situation de gestion réelle (MCD, MLD, MPD)** ;
- de **créer une base de données relationnelles** ;
- d'**interroger une base de données relationnelles par l'algèbre relationnelle et le langage SQL**.

Plan du cours

Chapitre 0 : Etude des Système d'information.

Chapitre 1 : Généralités sur les BdDR

Chapitre 2 : Modélisation de BdDR

Chapitre 3 : Langage d'interrogation de BdDR

Chapitre 0

Etude des Systèmes d'Information de Gestion (SIG)

- Définition et fonctions des systèmes d'information
- Place de l'informatique dans les systèmes d'information
- Composantes et activités d'un Système d'information
- Typologies des systèmes d'information d'entreprise

1- Définition et fonctions des SI

Un système d'information (SI) est un ensemble de moyens humains, matériels, immatériels et financiers destinés :

- à la collecte/génération de l'information
- à la mémorisation/stockage de l'information
- au traitement de l'information
- à la diffusion/communication de l'information

1- Définition et fonctions des SI

L'analyse systémique permet de faire émerger la notion de système d'information en tant que représentation de l'activité du système opérant et/ou du système de pilotage, et de ses échanges avec l'environnement, conçue à l'initiative du système de pilotage en fonction des objectifs à atteindre et de l'organisation choisie.

1- Définition et fonctions des SI

Rôle d'un SI?



2- Place de l'informatique dans les SI

Système d'information et Système Informatique



Un Système d'information existe dès qu'une organisation se crée. Cette notion existait avant l'invention de l'Informatique.

Il n'est pas forcément souhaitable d'informatiser à 100% un système d'information.

Un système d'information peut être vu (appréhendé) selon quatre angles :

Vue externe : permet de répondre à la question POURQUOI.

Vue fonctionnelle : permet de répondre aux questions QUI, OU et QUAND?

Vue Informationnelle : permet de répondre à la question QUOI.

Vue applicative : donne la carte des flux inter-applicatifs et l'architecture technique. Permet de répondre à la question COMMENT?

Les trois premières vues fournissent la compréhension « métier », la dernière l'implantation de la solution informatique.

2- Place de l'informatique dans les SI

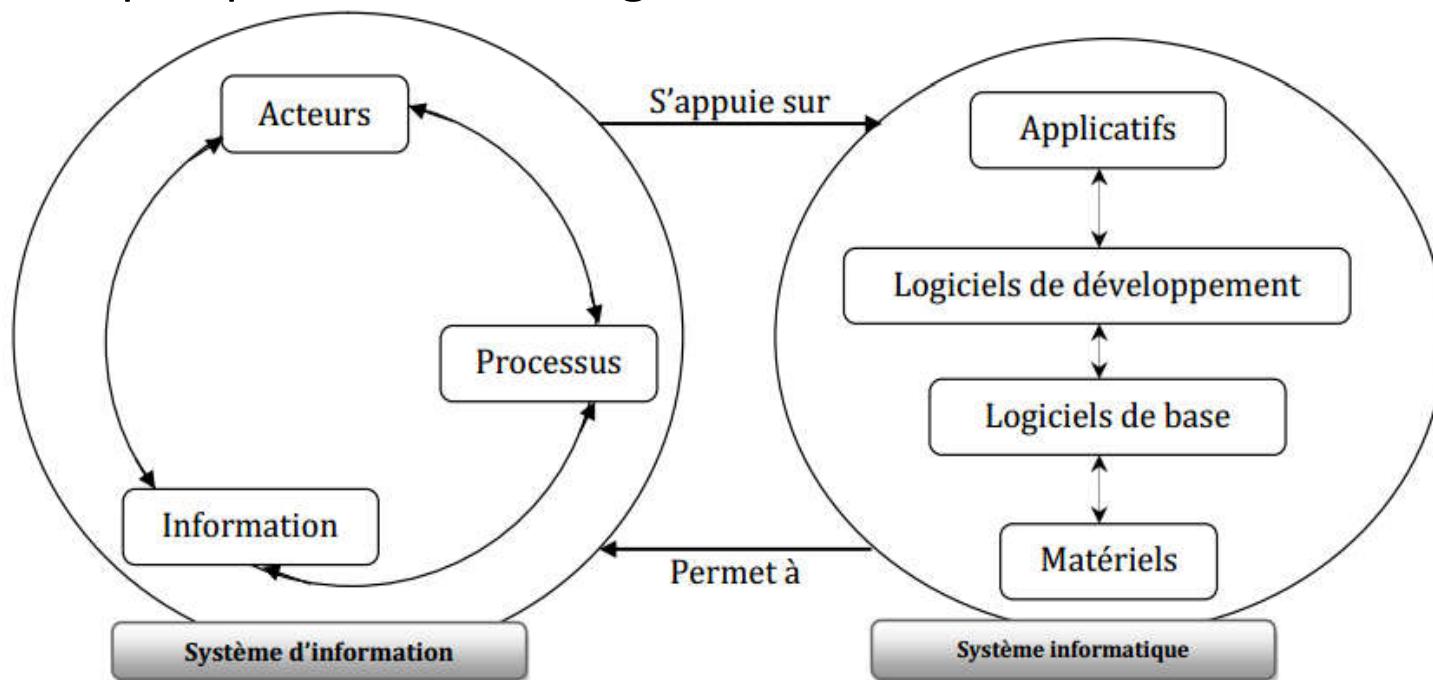
Aux trois rôles traditionnels joués par l'information dans les organisations : **support pour l'action, mémoire des activités, aide à la prise de décision**, les technologies de l'information et de la communication ont ajouté des fonctions considérables qui élargissent l'étendue des systèmes d'information et en modifient profondément la structure. Ce phénomène est particulièrement visible à travers trois terrains d'innovation :

- *la dématérialisation croissante des objets de gestion,*
- *l'exigence systématique de qualité,*
- *l'émergence de structures informationnelles virtuelles liées aux structures réelles.*

L'extension de la place de l'information et de ses technologies au sein des organisations déterminent l'évolution de la notion même de système d'information.

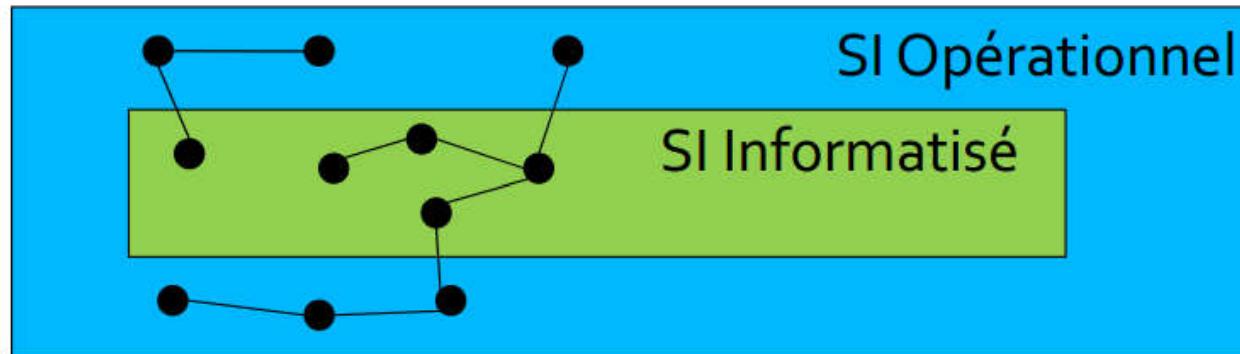
2- Place de l'informatique dans les SI

La confusion ou l'intégration entre le système d'information et le système informatique tient autant au phénomène de numérisation croissante de la réalité informationnelle qu'à l'évolution déterminante du rôle de l'informatique dans l'organisation. Le tableau ci-dessous, adapté de l'analyse de **Gérard Karsenti**, souligne, à travers quelques critères, les grands traits de cette évolution.



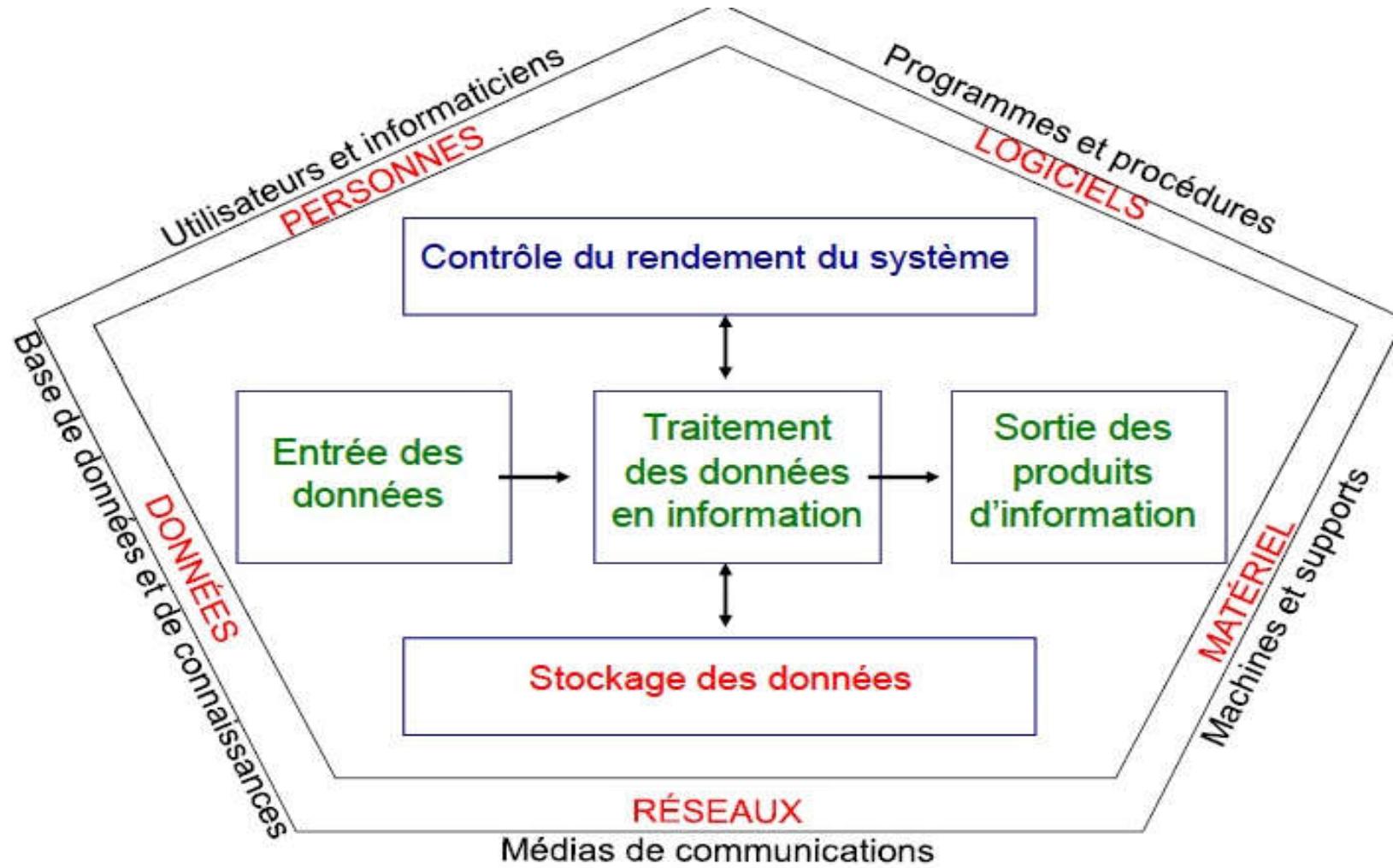
2- Place de l'informatique dans les SI

L'informatique facilite la gestion d'un SI mais ne le couvre pas dans son ensemble.



- SIO – Système d'Information Opérationnel = toute l'activité autour du SI
- SII – Système d'Information Informatisé = uniquement le contenu informatisé (fichiers, bases, logiciels, ...)

3- Composantes et activités d'un SI



4- Typologie des systèmes d'information d'entreprise

- Connaître les systèmes d'information en usage dans les entreprises
 - **L'évolution des SI dans les entreprises**
 - **Les grands systèmes fonctionnels**
 - Systèmes d'information de Gestion des Ressources Humaines, Comptabilité /CG /finance, Marketing et ventes, Gestion des Opérations, Gestion de Production
 - **Les systèmes intégrés (transversaux)**
 - CRM, ERP, EAI
- Comprendre le rôle de ces systèmes d'information dans la gestion des données et l'organisation des processus

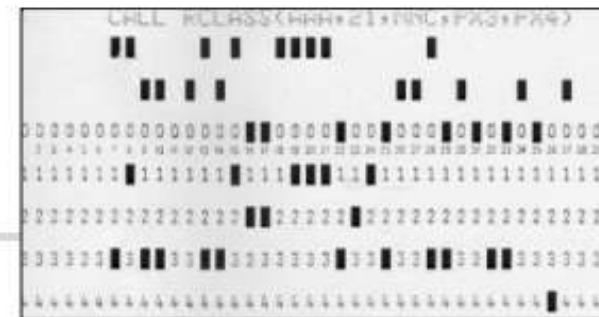
4- Typologie des systèmes d'information d'entreprise

L'évolution des SI dans les entreprises

Name	Era	Scope	Perspective	Example	Technology Symbols
Calculation systems	1950–1980 (Your grandfather)	Single purpose	Eliminate tedious human calculations. “Just make it work!”	Payroll General ledger Inventory	Mainframe Punch card
Functional systems	1975–20?? (Your mother)	Business function	Use computer to improve operation and management of individual departments.	Human resources Financial reporting Order entry Manufacturing (MRP and MRP II)	Mainframe Stand-alone PCs Networks and LANs
Integrated systems (also cross-functional or process-based systems)	2000 ... (You)	Business process	Develop IS to integrate separate departments into organization-wide business processes.	Customer relationship management (CRM) Enterprise resource planning (ERP)	Networked PCs Client-servers The Internet Intranets

4- Typologie des systèmes d'information d'entreprise

Les systèmes de calcul (1^{ère} génération)



- Premiers systèmes d'information apparus dans les entreprises
 - L'objectif était de prendre en charge des calculs répétitifs et fastidieux (automatisation)
 - Premières applications : calcul de la paie, comptabilité et suivi des quantités en stock
- Ces systèmes traitaient des données mais produisaient peu d'informations

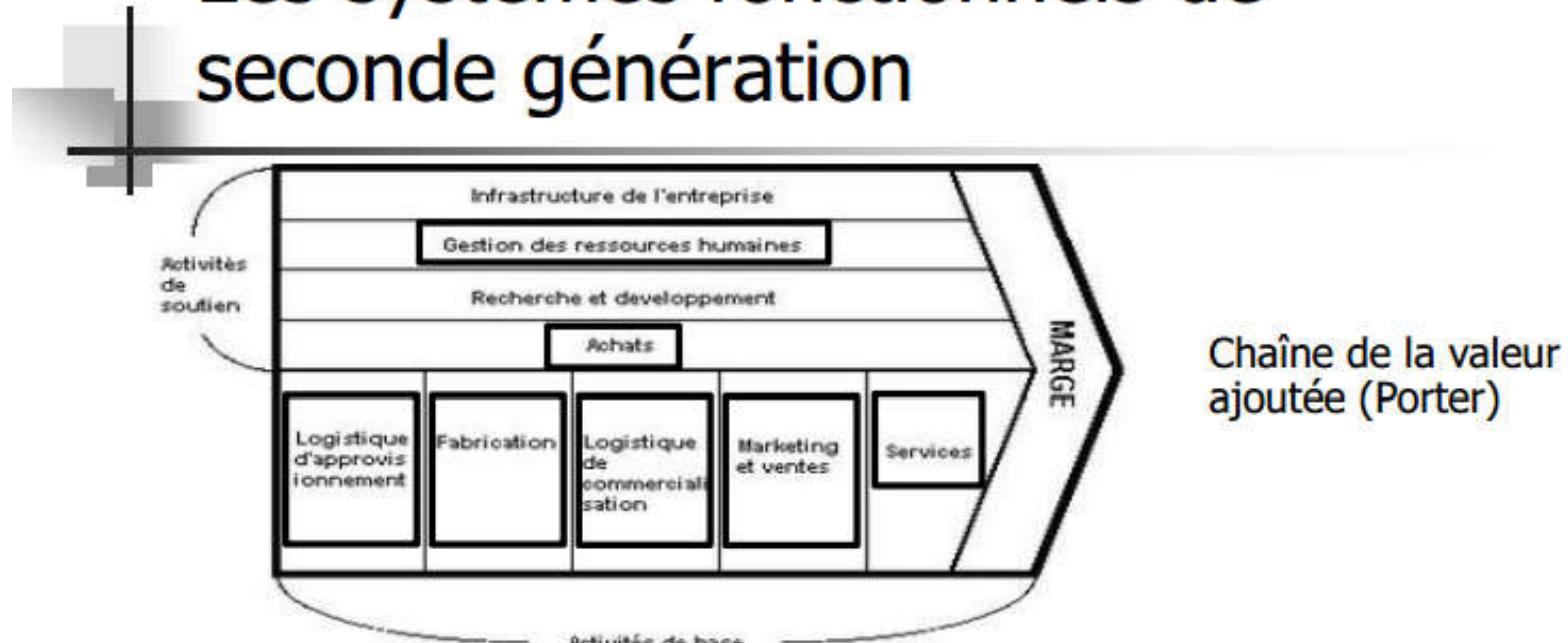
4- Typologie des systèmes d'information d'entreprise

Les systèmes fonctionnels (2^{ème} génération)

- Ils se sont développés à partir des systèmes de 1^{ère} génération en élargissant le périmètre traité au sein de l'activité
 - Calcul de la paie -> gestion ressources humaines
 - Comptabilité -> reporting financier
 - Gestion des stocks -> gestion de production
- Ils sont plus conviviaux et mieux tournés vers les besoins des utilisateurs

4- Typologie des systèmes d'information d'entreprise

Les systèmes fonctionnels de seconde génération



- Ils répondent aux besoins d'une fonction précise de l'entreprise
- Mais leur problème est qu'ils sont isolés et communiquent très mal entre eux : on parle d'« îlots d'automatisation »

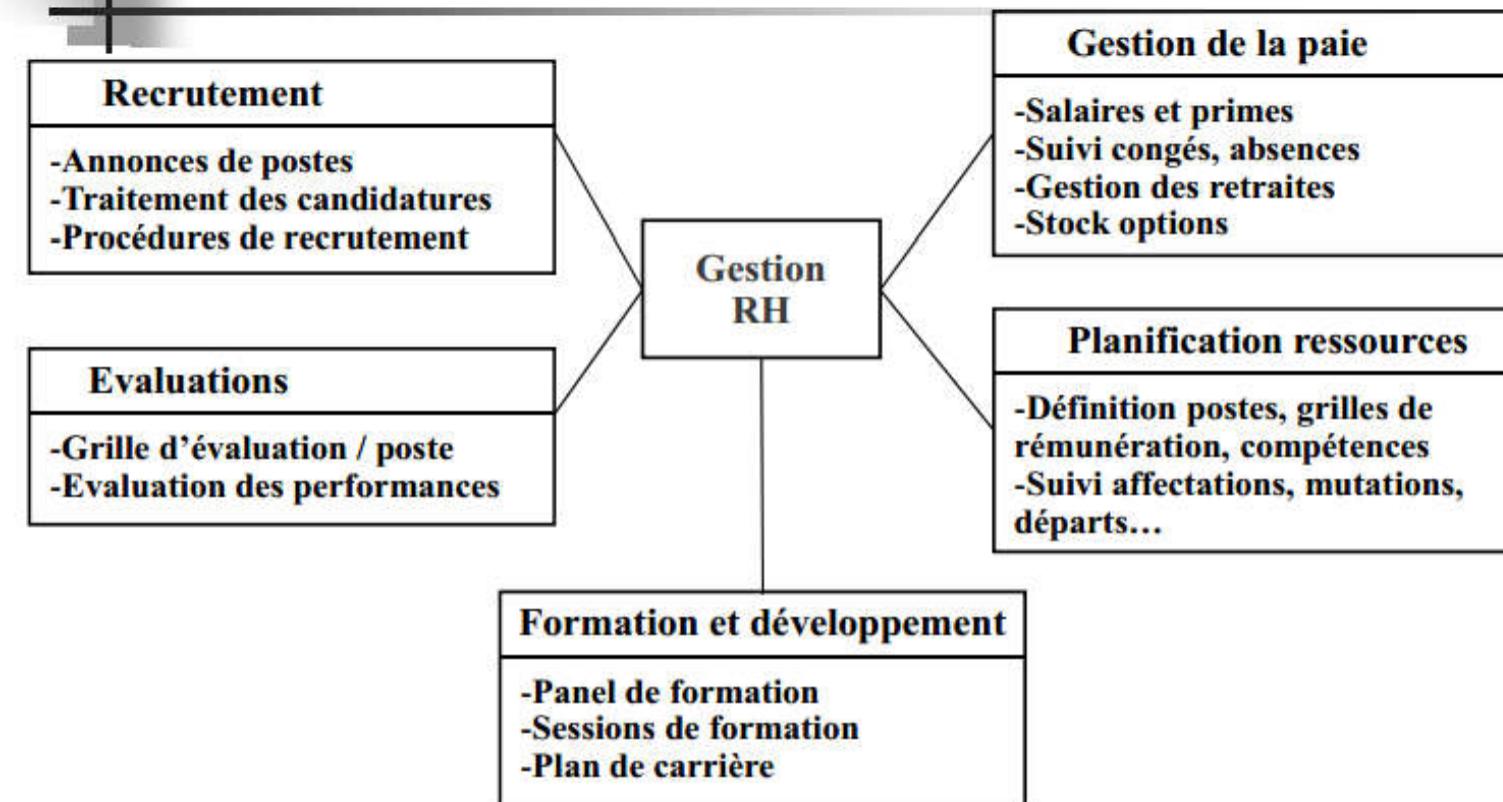
4- Typologie des systèmes d'information d'entreprise

Systèmes d'information fonctionnels classiques dans les entreprises

Function	Example Information Systems
Human resources	Recruiting Compensation Assessment Development and training Human resources planning
Accounting and finance	General ledger Financial reporting Cost accounting Budgeting Accounts receivable Accounts payable Cash management Treasury management
Sales & marketing	Lead tracking Sales forecasting Customer management Product management
Operations	Order entry Order management Finished-goods inventory management Customer service
Manufacturing	Inventory Planning Scheduling Manufacturing operations

4- Typologie des systèmes d'information d'entreprise

Systèmes de gestion des ressources humaines



4- Typologie des systèmes d'information d'entreprise



Valeur ajoutée des SI RH

- Optimiser l'efficacité du département RH en automatisant les processus
- Donner facilement accès aux managers aux profils de leurs employés
- Améliorer le processus d'évaluation et de recrutement des candidats
- Mieux gérer les carrières et les évolutions

4- Typologie des systèmes d'information d'entreprise

Zoom sur l'évolution du recrutement avec Internet

- Le recrutement utilise de + en + le web
 - Publication offres,
 - Recherche cv,
 - Alertes,
 - Abonnement à des sites spécialisés
- Modèle économique type LinkedIn: gratuit pour le particulier, les entreprises payent pour les services et l'accès à l'information

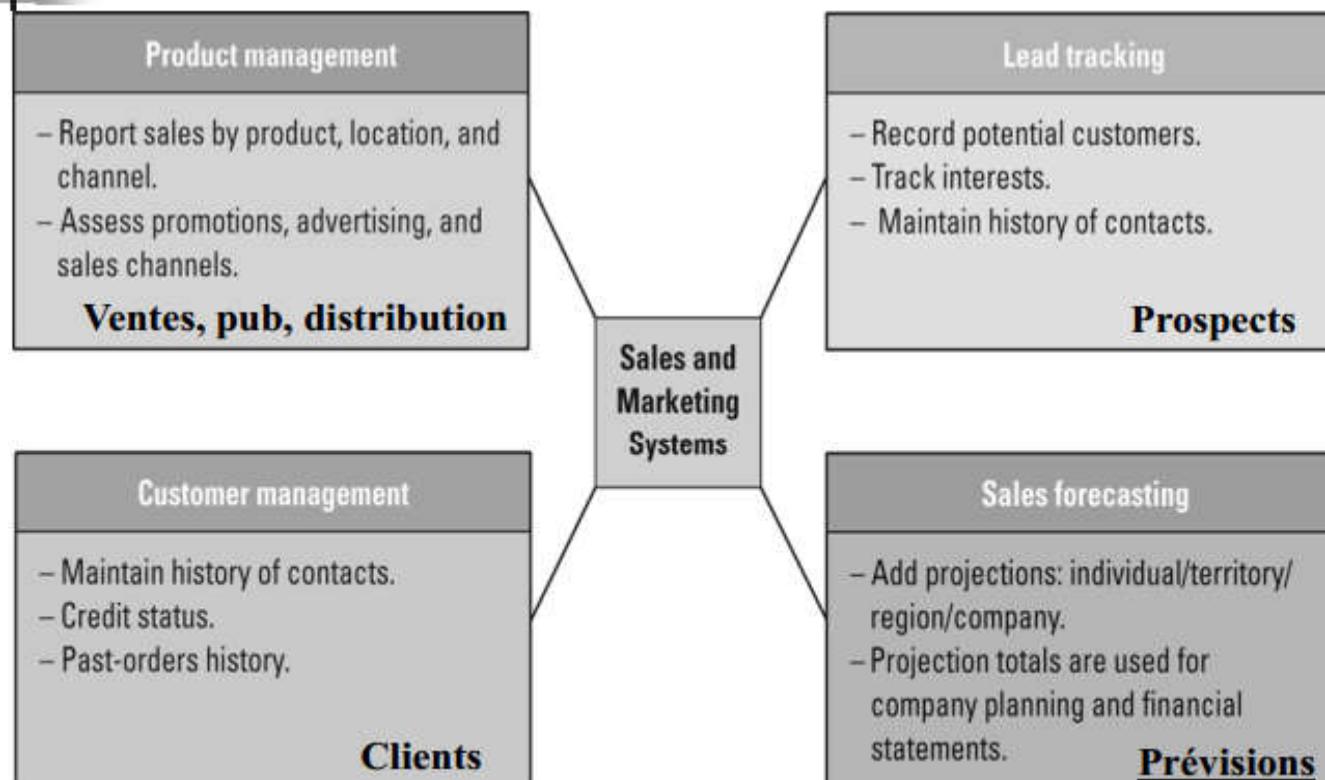
4- Typologie des systèmes d'information d'entreprise

Systèmes de gestion comptable et financière

- Les applications de comptabilité des tiers effectuent :
 - L'enregistrement des factures
 - Le suivi des paiements / règlements, positions des clients/fournisseurs, impayés
- La gestion de la trésorerie suit l'équilibre des flux financiers à très court terme de l'entreprise
 - Prévenir le risque de cessation de paiement, optimiser l'utilisation des excédents

4- Typologie des systèmes d'information d'entreprise

Systèmes de marketing et de suivi des ventes

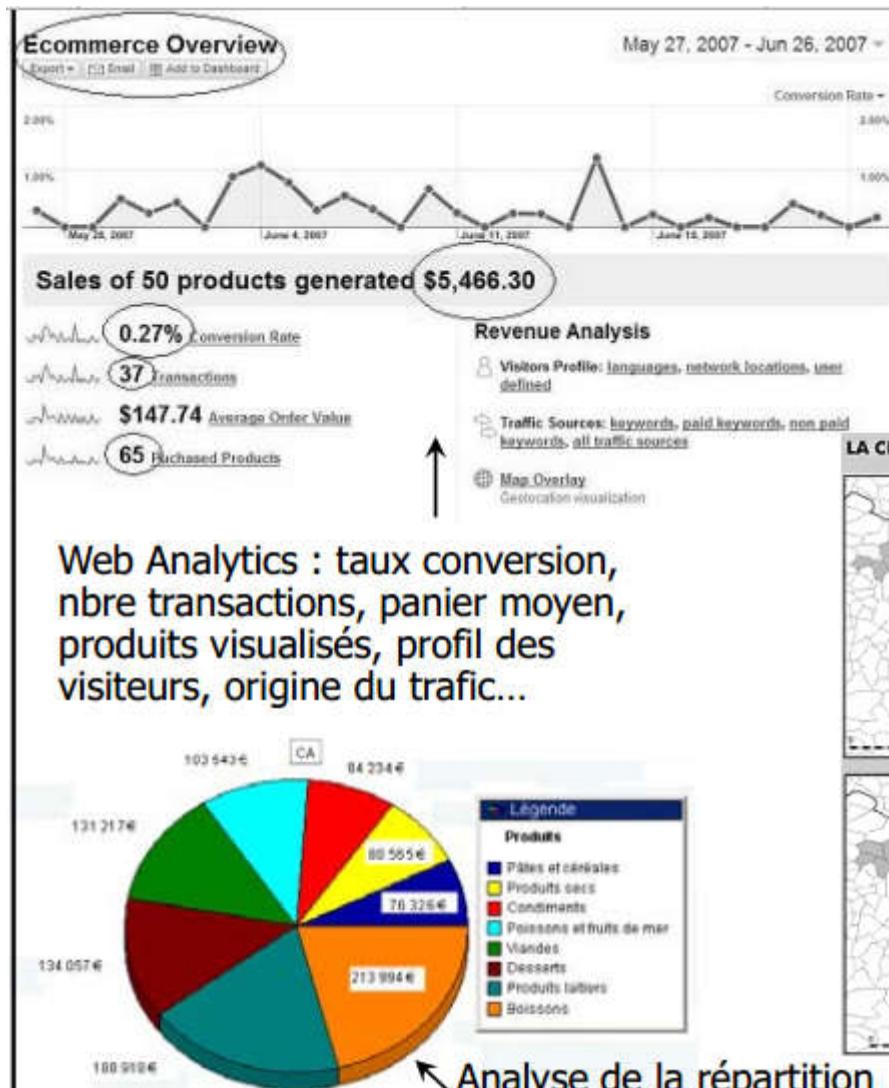


4- Typologie des systèmes d'information d'entreprise

Systèmes de marketing et de suivi des ventes

- Ils enregistrent les données sur les prospects, leurs attentes et leurs intentions d'achat et l'historique de leurs contacts avec la force de vente.
- Le suivi des ventes utilise des systèmes de prévision pour anticiper et influencer le CA futur.
- Les systèmes de gestion des clients tracent l'historique des contacts clients, des commandes et des paiements.

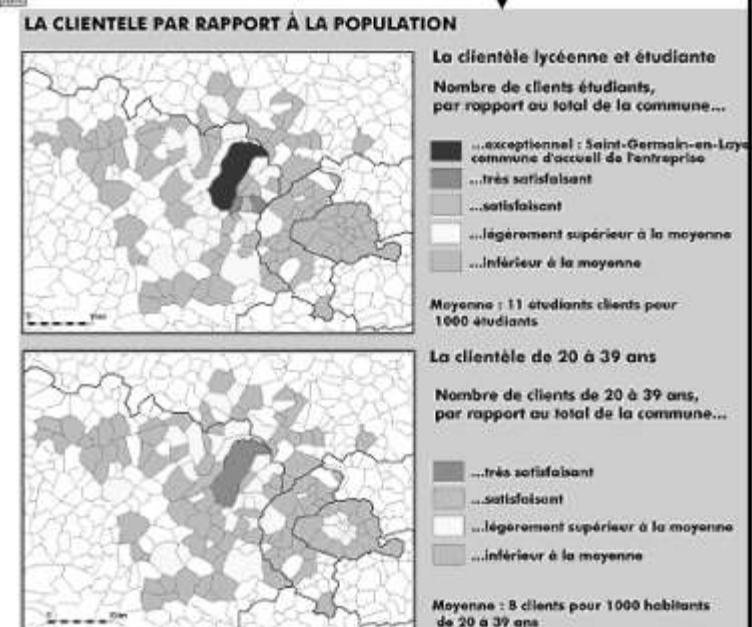
4- Typologie des systèmes d'information d'entreprise



Systèmes de marketing et de suivi des ventes

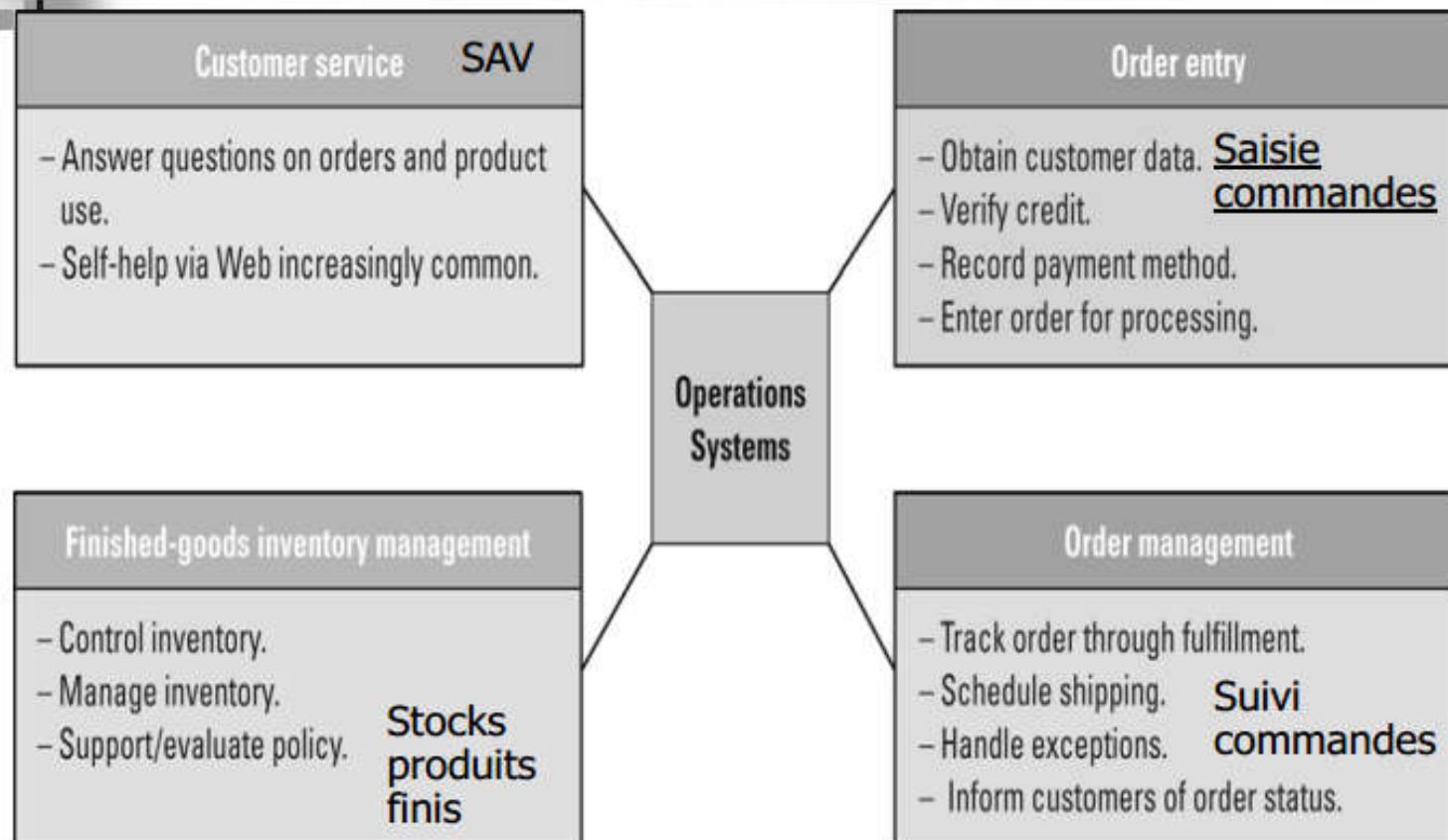
Systèmes de Pricing

Etude d'implantation



4- Typologie des systèmes d'information d'entreprise

Systèmes de gestion des opérations



4- Typologie des systèmes d'information d'entreprise

Systèmes de gestion des opérations

- Les systèmes de gestion des opérations intègrent la gestion des stocks de produits finis, la saisie et suivi des commandes et la gestion du SAV.
- Ils sont vitaux pour les entreprises n'effectuant pas de production (distributeurs, grossistes, détaillants).

**Formulaire de saisie
des commandes**

The screenshot shows a Windows application window titled 'Commandes'. The interface is divided into several sections:

- Customer Information:** Facturé à: Bon app' (Address: 12, rue des Bouchers, Marseille, France), Envoyé à: Bon app' (Address: 12, rue des Bouchers, Marseille, France).
- Representative:** Peacock, Margaret.
- Order Details:** N°comm: 11076, Date comm: 06-05-1998, Livrer avant: 03-06-1998, Date envoi: [empty].
Messenger options: Speedy (unchecked), United (checked), Federal (unchecked).
- Product Table:** A grid showing items, unit price, quantity, discount, and total price.

Produit	Prix unitaire	Quantité	Remise (%)	Prix total
Teatime Chocolate Biscuits	7,00 €	10	25%	52,50 €
Tofu	19,00 €	20	25%	380,00 €
Grandma's Boysenberry Spread	20,00 €	20	25%	400,00 €
- Total Summary:** Sous total: 637,50 €, Port: 191,40 €, Total: 828,90 €.
- Buttons:** Afficher les produits du mois, Imprimer facture.

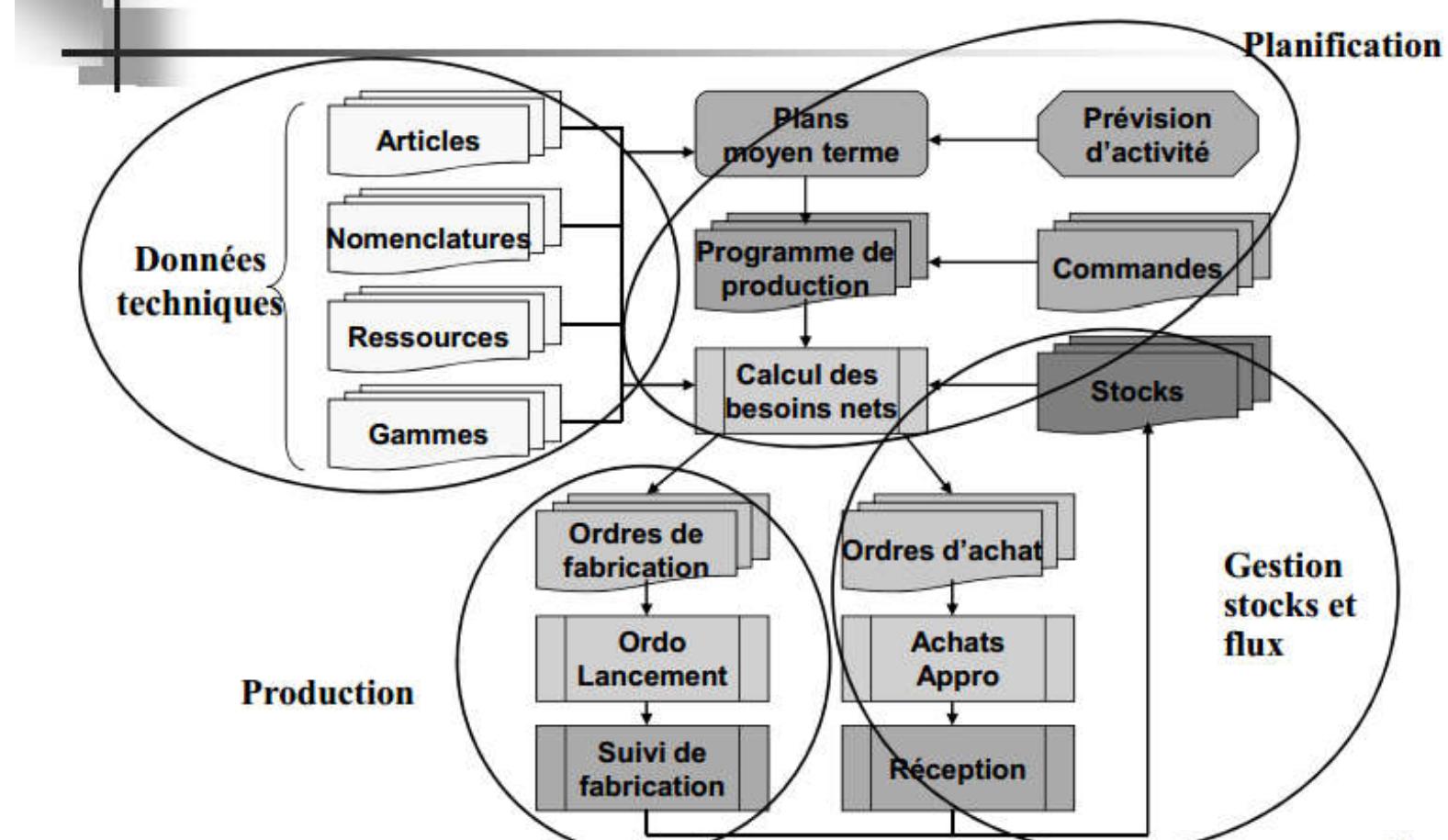
4- Typologie des systèmes d'information d'entreprise

Systèmes de gestion des opérations

- La gestion des commandes
 - suit les commandes,
 - organise la livraison,
 - gère les problèmes divers (ruptures de stock)
 - informe les clients quant au statut de leur commande.
- Le service après vente permet aux clients
 - de contacter l'entreprise
 - de poser des questions sur les produits, l'état de leur commande, les problèmes suite à la livraison
 - de faire des réclamations.

4- Typologie des systèmes d'information d'entreprise

Systèmes de gestion de production



4- Typologie des systèmes d'information d'entreprise

Systèmes de gestion de production

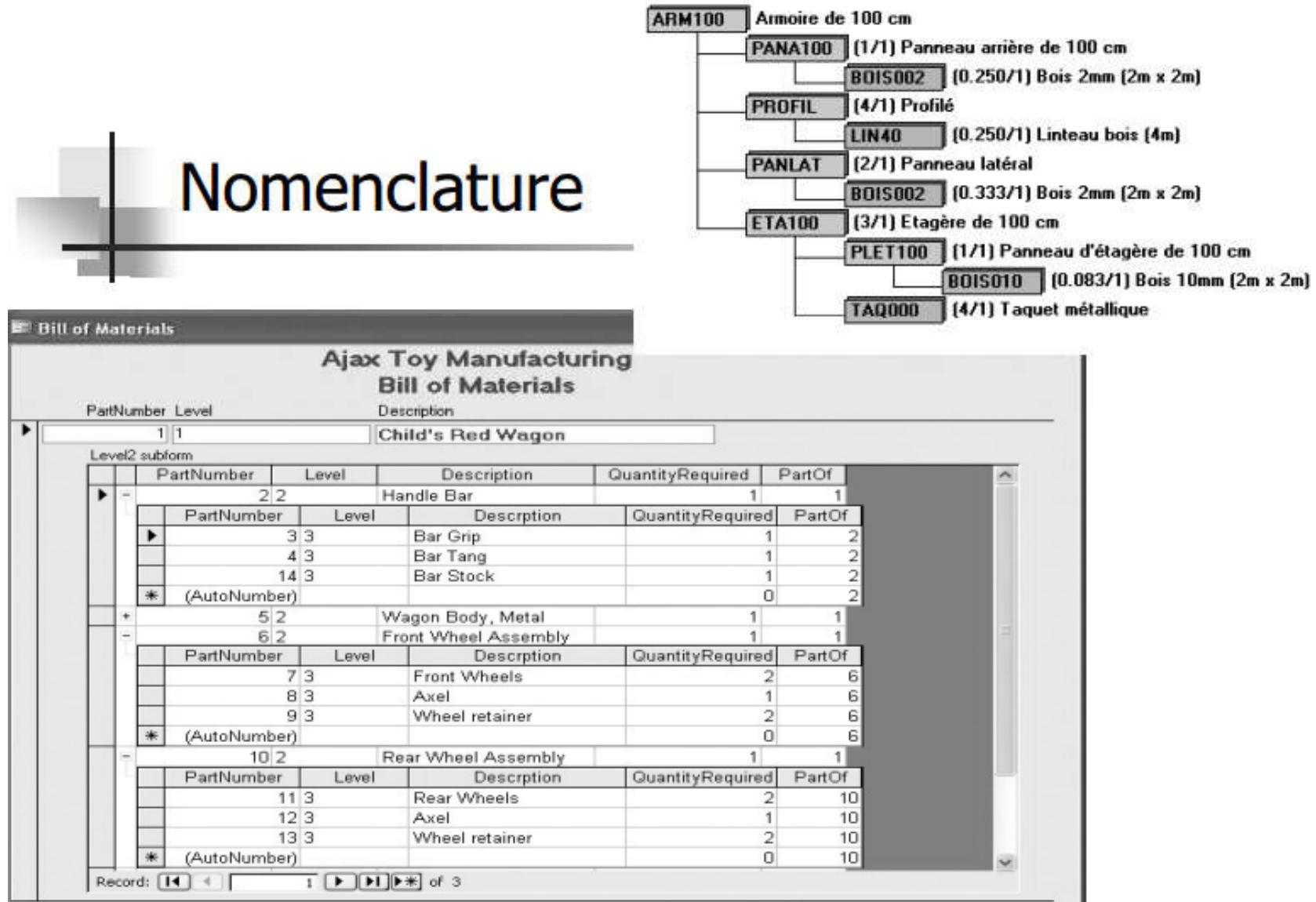
- Ils facilitent les opérations de production
- Ils incluent
 - le suivi des opérations de production
 - la gestion des données techniques (nomenclatures)
 - la planification de la production
 - la gestion des stocks et des flux (matières premières, produits semi finis et finis)

4- Typologie des systèmes d'information d'entreprise

La gestion des données techniques

- Pour gérer la production et les achats, il faut d'abord établir la liste des matières premières et des composants entrant dans les produits finis
- La nomenclature (bill of material) décrit les composantes
 - des produits finis,
 - de leurs sous-ensembles
 - en couvrant l'intégralité du cycle de production

4- Typologie des systèmes d'information d'entreprise



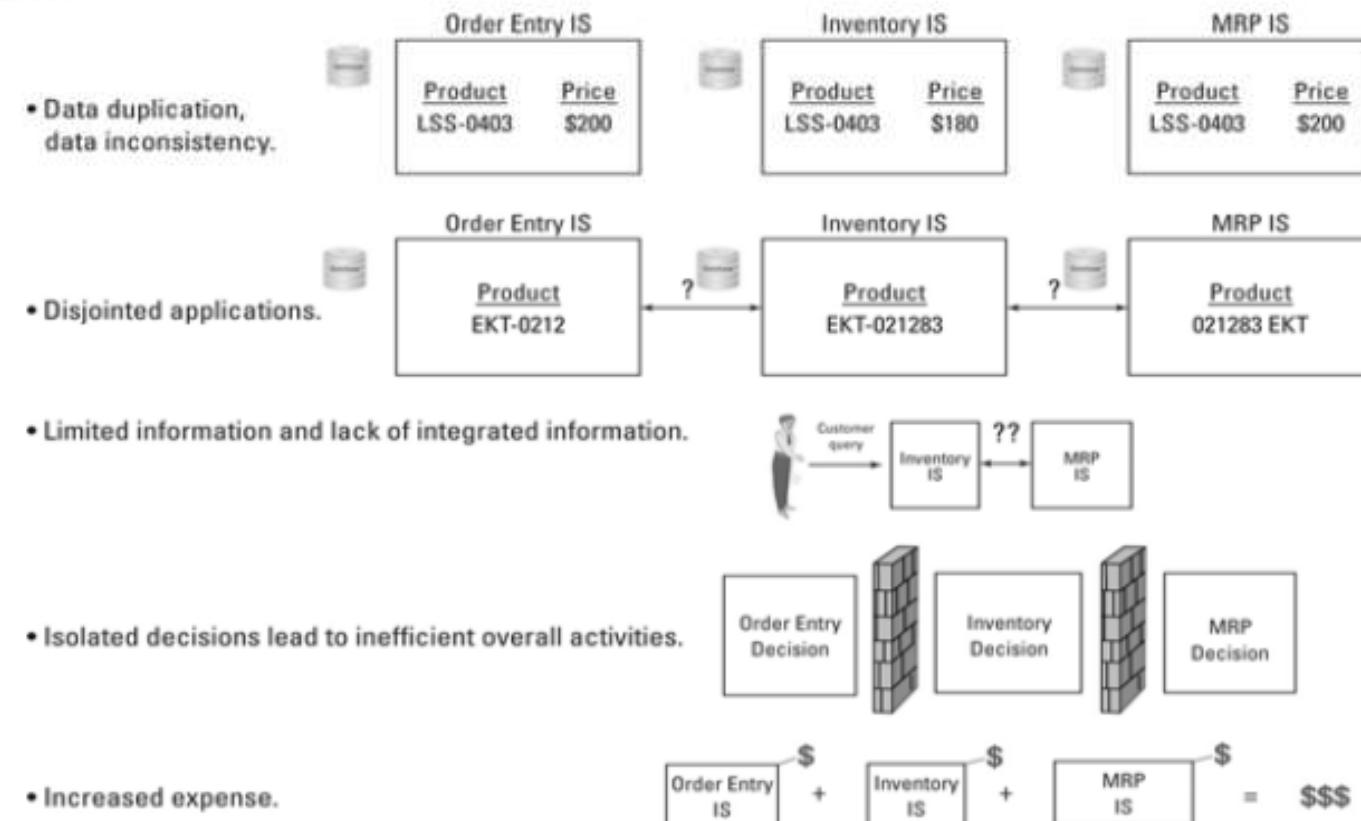
4- Typologie des systèmes d'information d'entreprise

Les problèmes des systèmes fonctionnels de seconde génération

- Ils rendent de considérables services aux départements qui les utilisent ; mais leur fonctionnement en silos isolés limite leur champ d'action.
- Avec des systèmes en silos
 - Les données sont dupliquées dans les bases de données de chaque système (risques d'incohérence)
 - Les processus métiers sont séparés
 - Il est très difficile d'intégrer l'information
 - Au global, l'entreprise encourt des coûts supplémentaires

4- Typologie des systèmes d'information d'entreprise

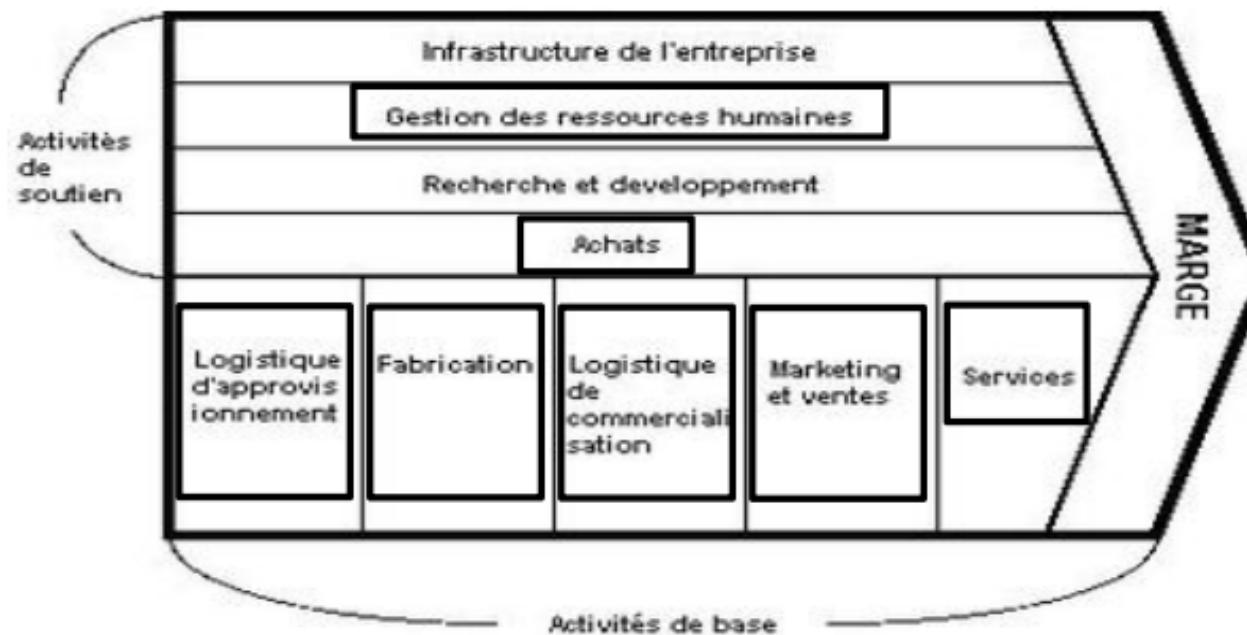
Les problèmes des systèmes fonctionnels de seconde génération



4- Typologie des systèmes d'information d'entreprise

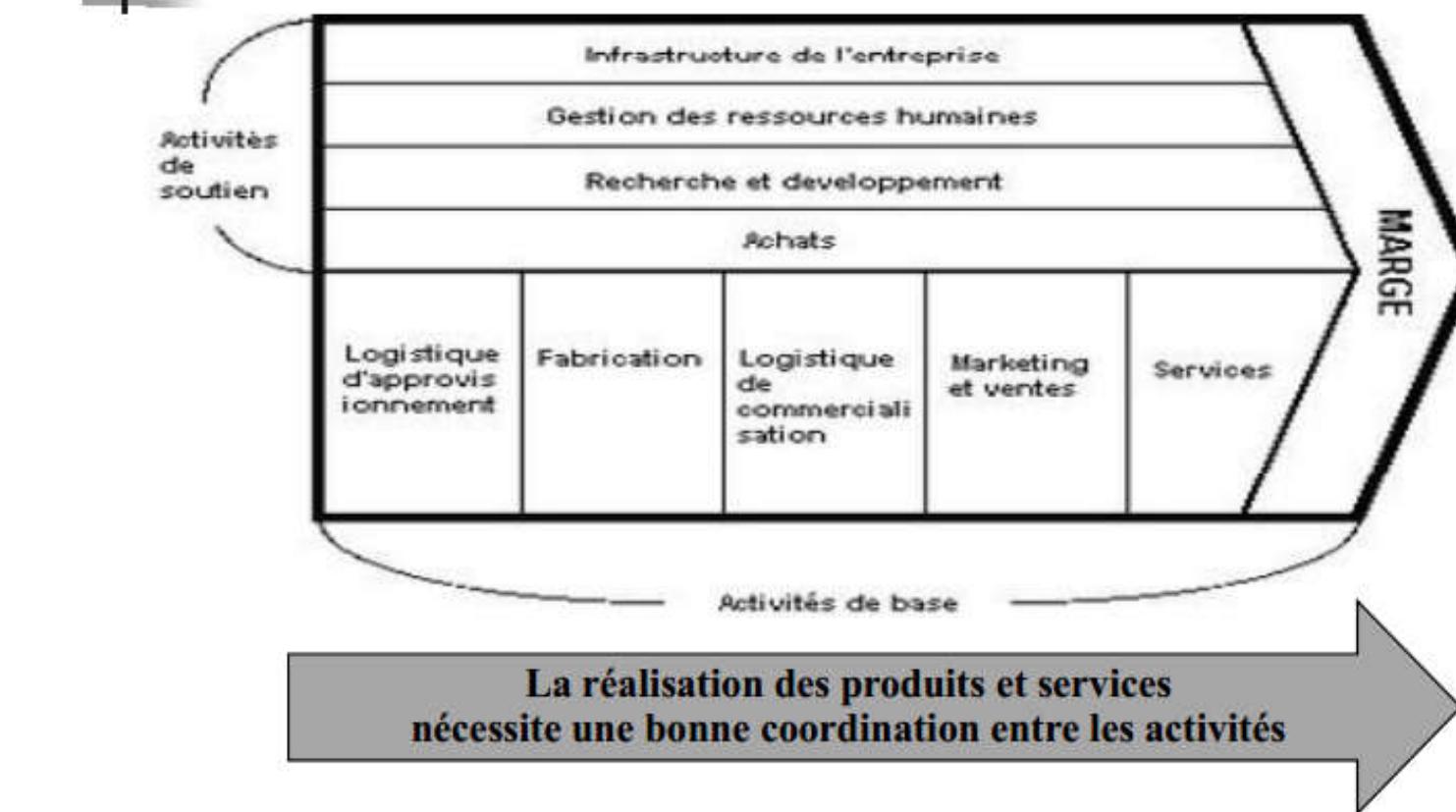
Les systèmes intégrés de troisième génération

- Objectif : éviter les silos des systèmes de 2^{ème} génération



4- Typologie des systèmes d'information d'entreprise

Objectif des systèmes intégrés :
coordonner les différentes activités



4- Typologie des systèmes d'information d'entreprise

Les systèmes d'information de 3^{ème} génération (intégrés)

- Le passage des systèmes fonctionnels aux systèmes intégrés est difficile.
- Les systèmes intégrés imposent aux départements de ne plus fonctionner en silo mais de coordonner leurs activités (données, processus, hommes).
- La plupart des entreprises utilisent aujourd'hui un mélange de systèmes fonctionnels et de systèmes intégrés.
- Pour faire face à la concurrence internationale, les entreprises doivent atteindre l'efficacité que procure l'usage de systèmes intégrés et facilitant les processus transversaux.

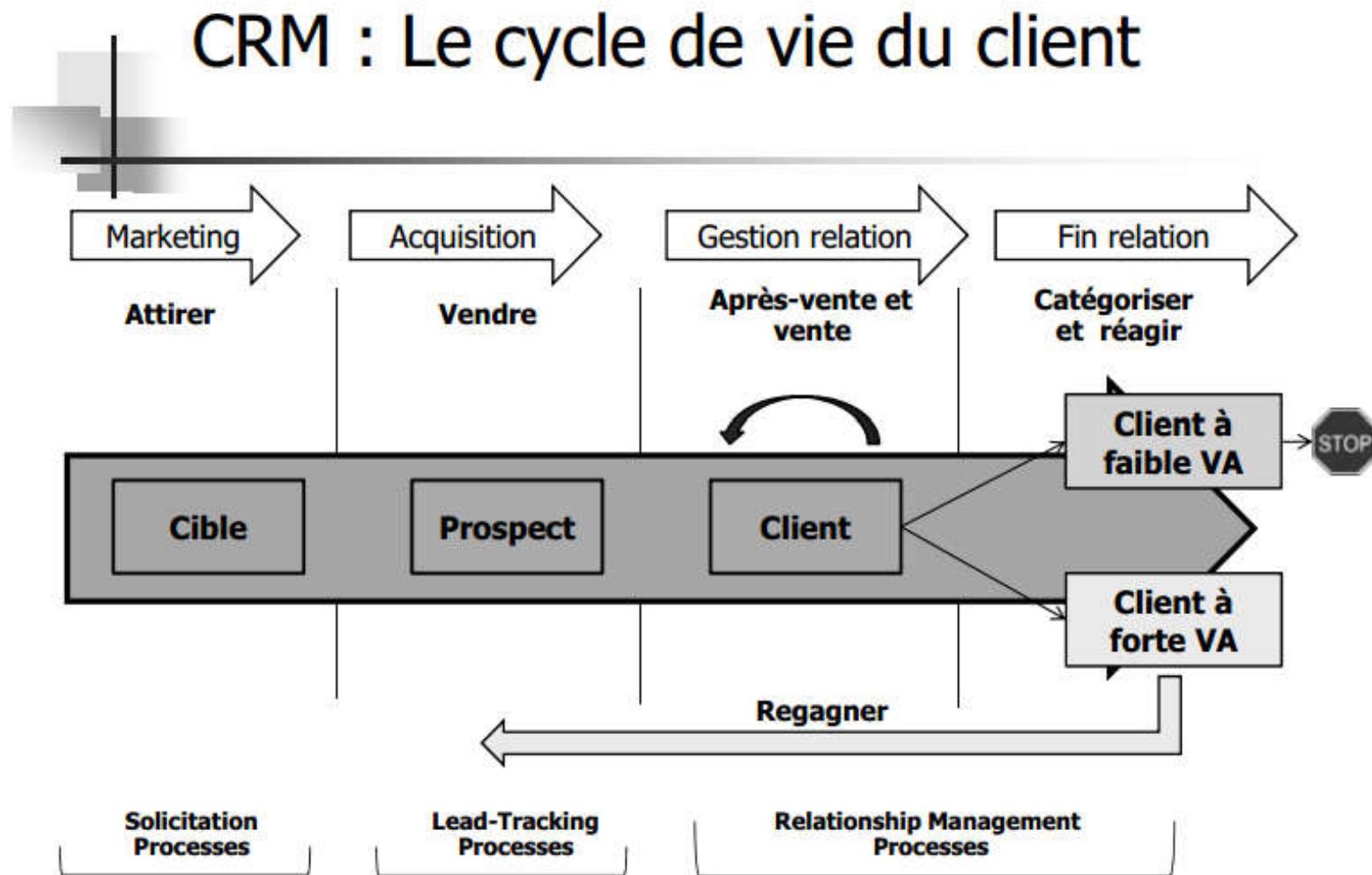
4- Typologie des systèmes d'information d'entreprise



Les systèmes intégrés

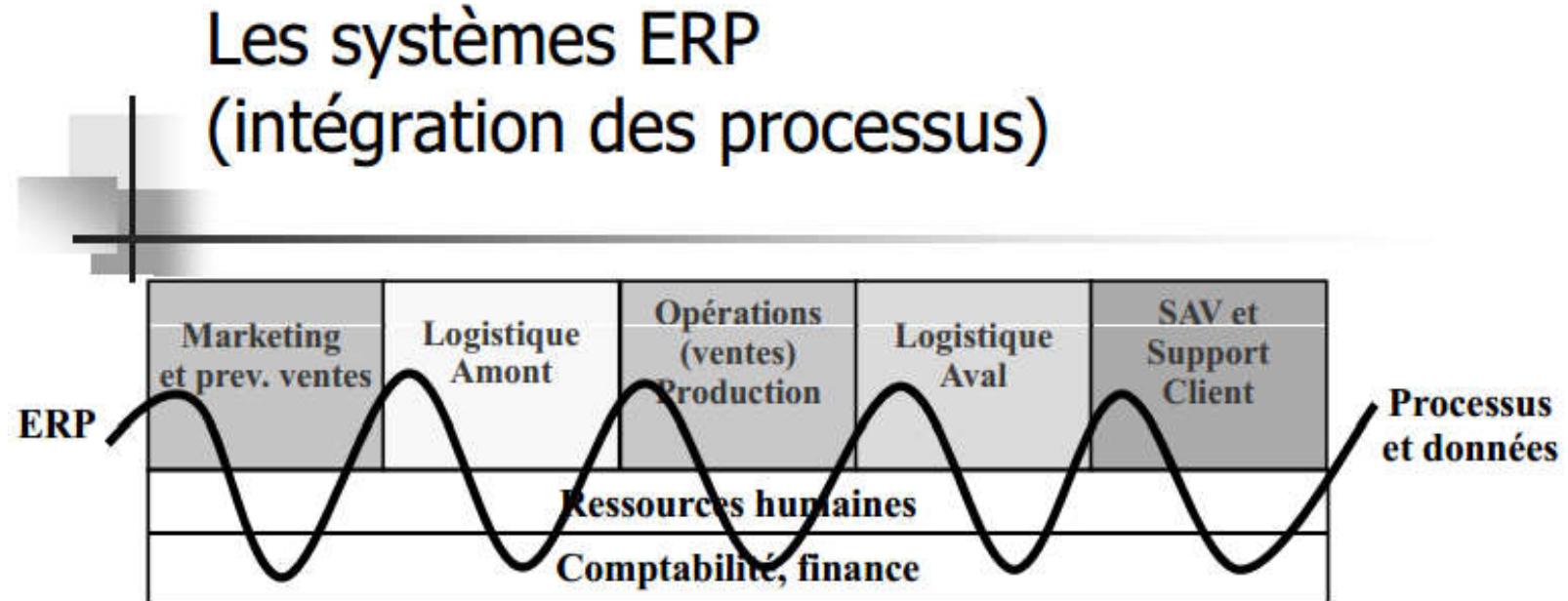
- Objectif : intégrer les processus dans une vision transversale : 3 axes
 - CRM (gestion relation client) : vision unique du client quel que soit le service concerné
 - ERP (Enterprise Resource Planning) – en français PGI (progiciels de gestion intégrés) : intégration des processus opérationnels, RH et comptabilité
 - EAI (Enterprise Application Integration) : ajout d'une couche logicielle pour échanger les données des différents systèmes îlots

4- Typologie des systèmes d'information d'entreprise



Source: Douglas MacLachlan, University of Washington.

4- Typologie des systèmes d'information d'entreprise



- Principe de l'ERP : intégration des processus et partage des données dans une vision transversale de la chaîne de la valeur.

4- Typologie des systèmes d'information d'entreprise

Pour être intégré, un progiciel de gestion (package) doit :

- émaner d'un concepteur unique ;
- garantir à l'utilisateur l'unicité d'information assurée par la disponibilité de l'intégralité de la structure de la base de données à partir de chacun des modèles, même pris individuellement ;
- reposer sur une mise à jour en temps réel des informations modifiées dans tous les modules affectés ;
- fournir des pistes d'audit basées la garantie totale d'une traçabilité des opérations de gestion ;
- couvrir un ensemble de fonctions de gestion de l'entreprise.



4- Typologie des systèmes d'information d'entreprise

SI intégré de type SAP

- Les ERP (Enterprise Resource Planning, PGI – progiciels de gestion intégrés) intègrent tous les processus principaux de l'entreprise.
- Ils viennent des systèmes de production MRP II et se sont d'abord développés dans les entreprises industrielles.
- Le leader du marché des ERP est SAP (SAP AG Corp., siège social en Allemagne).

4- Typologie des systèmes d'information d'entreprise

Les avantages des logiciels proposant des processus pré-définis

- Quand une organisation achète une application SAP ou Oracle, elle a accès à des processus pré-définis et souvent conçus pour répondre aux besoins de son secteur d'activité.
- Dans la plupart des cas, les entreprises modifient leurs propres processus pour ce conformer aux solutions standard embarquées dans le logiciel (- pourquoi?)

4- Typologie des systèmes d'information d'entreprise

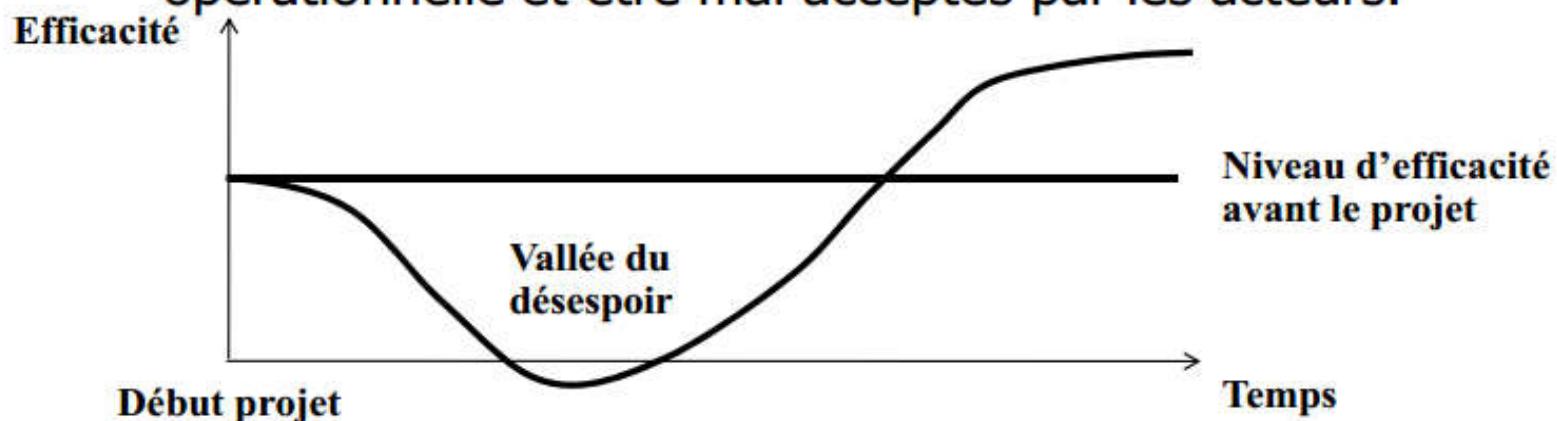
Les avantages des logiciels proposant des processus pré-définis

- Dans certains cas, le principal intérêt pour une entreprise dans le choix d'un logiciel intégré est l'existence de processus transversaux pré-définis (plus que logiciel lui-même).
- Acheter un système intégré évite à l'entreprise une perte de temps, d'argent, ainsi que la gestion de projets complexes d'élaboration de nouveaux processus.
- Cela lui permet également de bénéficier de processus testés et éprouvés, correspondant aux règles de l'art en usage dans son secteur d'activité.

4- Typologie des systèmes d'information d'entreprise

Les inconvénients des processus pré-définis

- Les processus définis dans le logiciel peuvent être très différents des processus existant dans l'organisation et demander à l'entreprise de gros efforts d'adaptation.
- Ces efforts d'adaptation peuvent perturber l'activité opérationnelle et être mal acceptés par les acteurs.



4- Typologie des systèmes d'information d'entreprise

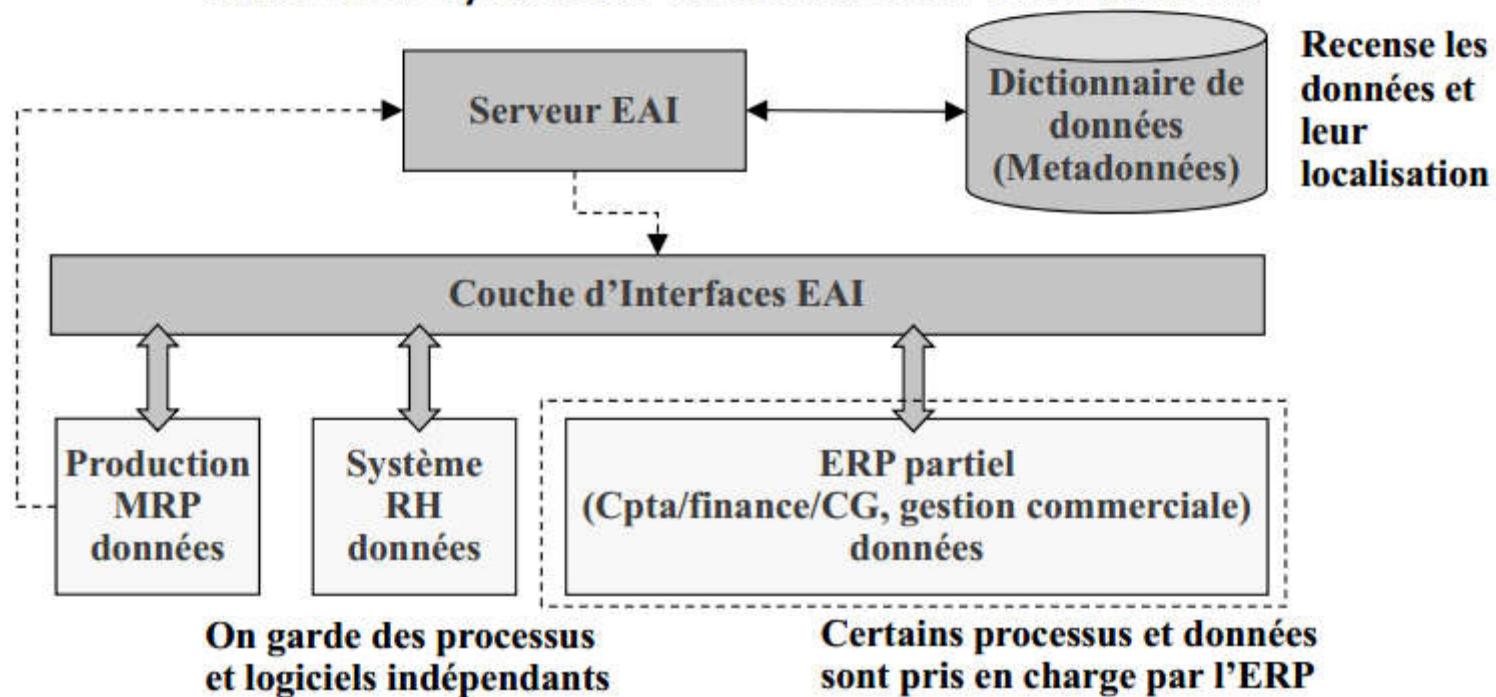
Les systèmes EAI (Enterprise Application Integration)

- Dans certains cas, l'ERP n'est globalement pas adapté (les produits existants prennent mal en charge le secteur d'activité)
- Ou certains modules de l'ERP ne sont pas adaptés
- Mais l'entreprise veut résoudre le problème des systèmes fonctionnels isolés
- L'EAI permet d'intégrer des systèmes isolés en ajoutant une couche logicielle qui connecte entre elles ces applications et leur permet de partager des données.

4- Typologie des systèmes d'information d'entreprise

Les solutions EAI (enterprise application integration)

- Objectif : rationaliser le système d'interfaces entre les différents systèmes d'information fonctionnels



4- Typologie des systèmes d'information d'entreprise

La refonte des processus métiers

- Le concept de chaîne de la valeur ajoutée a donné lieu au développement des pratiques de refonte des processus métiers (BPR).
- L'idée centrale est que les organisations ne doivent pas se focaliser sur l'automatisation ou l'amélioration des systèmes en silos de 2^{ème} génération.
- Elles doivent plutôt élaborer de nouveaux processus, plus efficents, et qui intègrent les activités de tous les départements contribuant à la chaîne de la valeur.

4- Typologie des systèmes d'information d'entreprise

Les difficultés de la refonte des processus

- Les projets de réorganisation de processus sont longs, difficiles et chers.
- Ils sont menés par des spécialistes (internes ou consultants) qui interviewent les personnes clés de nombreux départements, documentent le système existant et proposent des alternatives.
- Les Managers étudient les recommandations des spécialistes, les discutent et tentent de mettre en œuvre les nouveaux processus.
- De nouveaux systèmes d'information sont développés pour accompagner la refonte de processus.

4- Typologie des systèmes d'information d'entreprise

Les difficultés de la refonte des processus

- Il est parfois nécessaire de mettre en œuvre les nouveaux processus avant même que le nouveau système soit complet.
- La résistance au changement constitue l'une des principales difficultés (cf. cas Alcatel).
- Tant que les nouveaux processus ne sont pas opérationnels, il reste une incertitude forte sur les résultats.
- Certains projets de réorganisation ont bien fonctionné; mais de nombreuses entreprises ont vu leurs projets échouer.

Chapitre 1

Généralités sur les bases de données

- Définitions des concepts clés
- Principes fondamentaux de SGBDR
- Niveaux de représentation d'une base de données

1- Définitions des concepts clés (1/3)

- **Une base de données** est un ensemble structuré de données accessible via un ordinateur (*smart et autres*) pour satisfaire continuellement plusieurs utilisateurs en même temps (simultanément).

Exemple : nom, tél., mél., adresse, ...

- Une BD est conçue, construite et remplie avec des données dans un but précis
- **Les bases de données se constituent fortement en vivier majeur des domaines de :** **Cloud Computing** (infonuagique ou informatique dématérialisée), **Business Intelligence** (informatique décisionnelle), **Intelligence Artificielle**, ...

1- Définitions des concepts clés (2/3)

- La structure de la BD dépend du modèle choisi
 - Hiérarchique ou réseaux (≈ 1960)
 - Relationnel ($\approx 1970/1980$)
 - Objet (≈ 1990)
 - XML (arborescent) (≈ 2000)
 - Graphe (en particulier RDF)
- Une BD peut avoir n'importe quelle taille (agenda personnel ≈ 100 entrées ; Facebook $\approx 600+$ millions d'utilisateurs)

1- Définitions des concepts clés (3/3)

- **Un Système de Gestion de Bases de Données (SGBD) est un ensemble d'outils logiciels permettant la manipulation de BD :**
 - Facilite la manipulation des données pour des non informaticiens ;
 - Fournit des fonctionnalités d'administration de la base.

Exemple de SGBD : MS Access, MySQL, Postgres, Oracle, ...

Exercice d'application

- Un projet C est-il un système d'information ?
- Donnez un exemple de système d'information et dire qui a (ont) besoin de l'information ?

Exemple de SI de la SNCF

- Plusieurs utilisateurs :



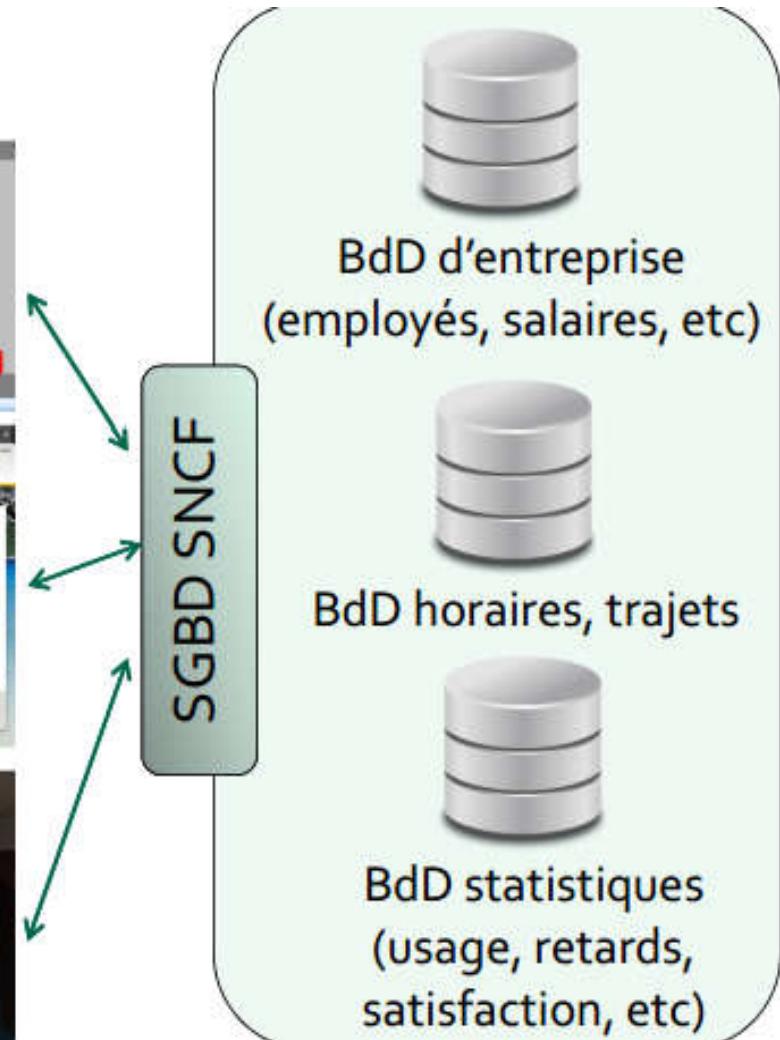
Guichetier



Internaute



Usager



2- Principes fondamentaux de SGBD (1/3)

- Fidélité
 - ◆ image fidèle de la réalité qu'elle modélise
- Unicité
 - ◆ pas de redondance d'information dans la BdD
- Indépendance
 - ◆ indépendant du modèle de stockage
- Concurrence
 - ◆ Gestion d'accès simultanés à une même donnée.
- Performance
 - ◆ temps d'exécution raisonnable

2- Principes fondamentaux de SGBD (2/3)

- Confidentialité
 - ◆ Accessibilité des données dépendant de l'utilisateur
- Intégrité
 - ◆ garanties de fiabilité et de cohérence.
- Robustesse
 - ◆ tolérant aux problèmes matériels, logiciels ou humains

3- Niveaux de représentation d'une BD (3/3)

- niveau externe (sous-schéma conceptuels)
 - définition des « interfaces » d'accès aux données
 - géré par le concepteur de la BdD et/ou les utilisateurs
- niveau conceptuel (e.g. modèle Entité-Association)
 - identification des concepts concrets et abstraits de la réalité
 - géré par le concepteur de la BdD
- niveau logique (e.g. modèle relationnel)
 - formalisation de la structure des données
 - géré par le concepteur de la BdD
- niveau physique (e.g. système de fichiers, index)
 - stockage physique des données
 - géré par le SGBD

Dans ce cours,
On ne s'occupe
Que de ça

Chapitre 2

Modélisation des bases de données

- Proposition de démarche
- Modèle Entité/Association
- Modèle relationnel
- Algèbre relationnelle

1- Proposition de démarche (1/1)

- **Définir** le nom et la finalité du système.
- **Identifier** les acteurs pour ainsi déceler les fonctionnalités attendues de la base de données.
- **Regrouper** tous les objets identifiables du monde réel (entités) qui sont porteurs d'informations pertinentes.
N.B : on appelle dictionnaire données le tableau récapitulatif de la désignation, de la description, du typage et des observations sur l'ensemble des informations à stocker dans la BD.
- **Faire sortir** les liens (associations) éventuels entre ces catégoriser.
- **Evaluer** leur degré de liaison (cardinalité) établie à travers les règles de gestion du système.
- **En déduire** la structure de la base de données (tables).

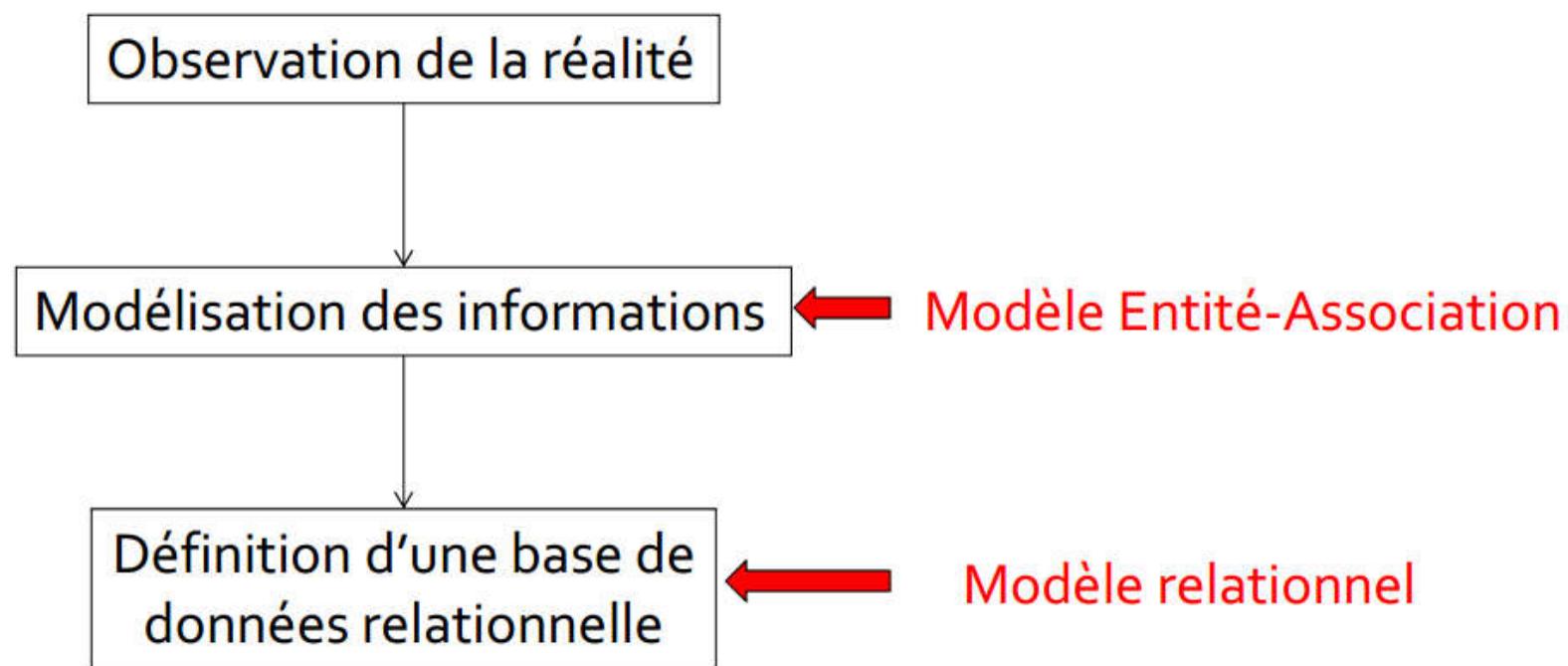
2- Modèle Entité/Association (1/12)

Origine du modèle Entité-Association

- Proposé par Chen en 1976
- Modèle sémantique pour comprendre et visualiser l'organisation des données
- Également appelé modèle EAR (Entité-Attribut-Relation)
- Objectif : concevoir un Modèle Conceptuel de Données (MCD)

2- Modèle Entité/Association (2/12)

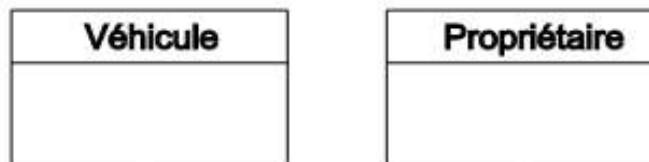
Objectifs & démarche



2- Modèle Entité/Association (3/12)

Concepts de base : Entité & Occurrence

(Type d')entité(s) : type d'objet abstrait ou concret provenant de l'observation du monde réel et pour lequel nous souhaitons enregistrer et connaître des informations



Une **entité** est une instance d'un type d'entités. Souvent, le terme « *entité* » est utilisé à la place de « *type d'entités* », auquel cas on parle d'**occurrence** pour les instances

Ex: la Peugeot 206 immatriculée « 1234 WW 42 » est une occurrence de l'entité Véhicule, et la personne prénommée « Paul Martin » née le 4 fevrier 1980 une occurrence de Propriétaire

2- Modèle Entité/Association (4/12)

Concepts de base : attributs

Attribut : caractéristique d'une entité (ou d'une association) que le concepteur juge nécessaire de répertorier

Véhicule
<i>immatriculation: String</i>
<i>marque: String</i>

Propriétaire
<i>nom: String</i>
<i>prénom: String</i>
<i>naissance: Date</i>

Remarques :

- Les attributs sont typés
- Une entité définit les attributs par leur type
- Une occurrence affecte une valeur à chaque attribut

2- Modèle Entité/Association (5/12)

Concepts de base : clé

Clé : attribut ou un ensemble d'attribut qui permet d'identifier de manière unique une occurrence d'une entité parmi toutes ses occurrences

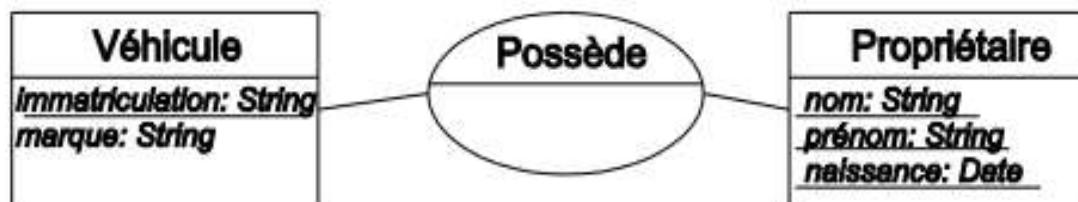
Véhicule	Propriétaire
<i>immatriculation: String</i> <i>marque: String</i>	<i>nom: String</i> <i>prénom: String</i> <i>naissance: Date</i>

- Plus d'une occurrence de Véhicule avec une même immatriculation ne peut pas exister
- Plus d'une occurrence de Propriétaire avec un même nom, prénom et date de naissance ne peut pas exister

2- Modèle Entité/Association (6/12)

Concepts de base :
association

Association : lien entre des entités présentant un intérêt pour la conception que l'on souhaite réaliser

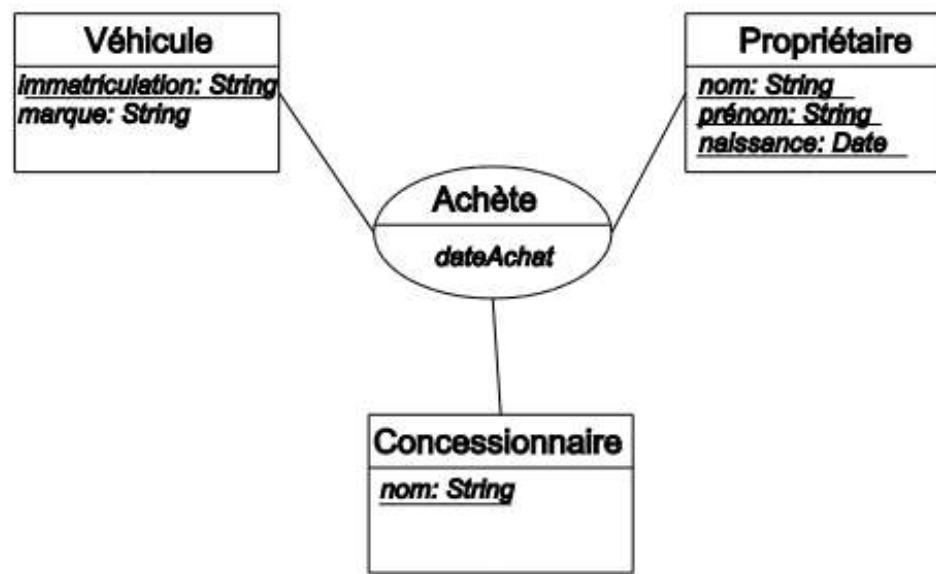


2- Modèle Entité/Association (7/12)

Concepts de base : association (2)

Une association peut :

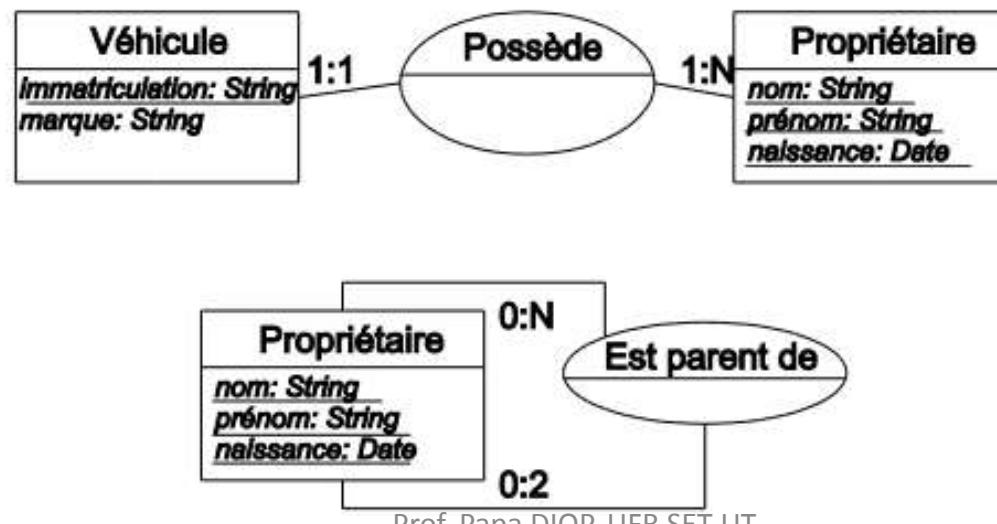
- relier plus de 2 entités
- bénéficier d'attributs



2- Modèle Entité/Association (8/12)

Cardinalité d'une association (min:max)

La cardinalité d'une association indique le nombre d'occurrences de chaque entité qui peuvent être impliquées dans une même association



2- Modèle Entité/Association (9/12)

Exemple de modélisation Entité-Association

On souhaite concevoir le SI correspondant à la gestion de la scolarité de l'EMSE. La scolarité fonctionne de la manière suivante :

- Un élève appartient à une promotion (1A, 2A ou 3A).
- Les élèves d'une promotion suivent plusieurs groupes pédagogiques (GP). Chaque groupe pédagogique est constitué d'unités pédagogiques (UP).
- A chaque GP et UP est affecté un enseignant responsable.
- Un élève obtient une note par UP et par GP qu'il suit.

2- Modèle Entité/Association (10/12)

Exemple : les entités

On représente les entités suivantes :

- Élève
- Enseignant
- Promotion
- Groupe Pédagogique
- Unité Pédagogique

2- Modèle Entité/Association (11/12)

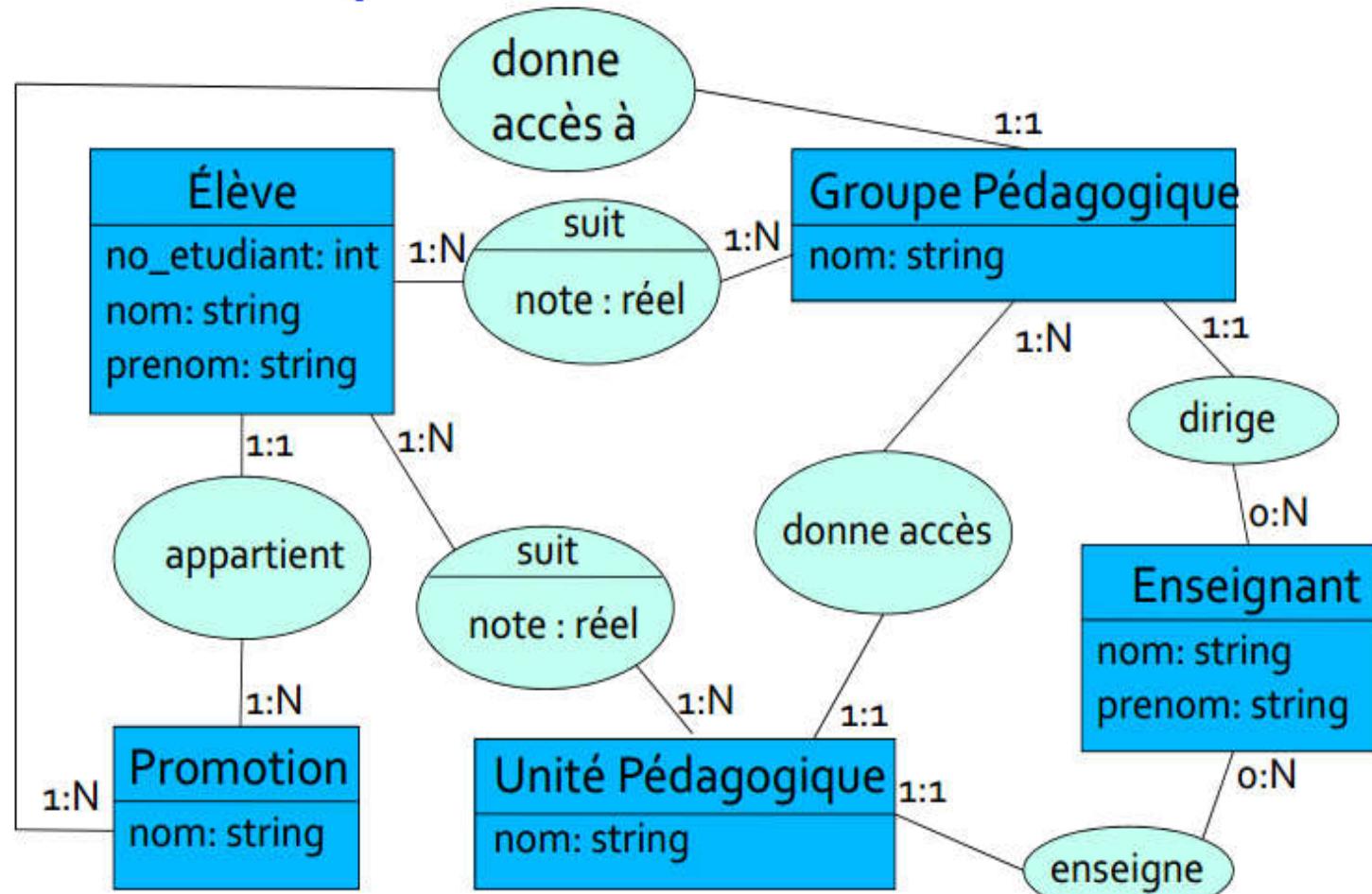
Exemples : les associations

On représente les associations suivantes :

- Un élève **appartient** à une promotion
- Une promotion **donne accès** à certains groupes pédagogiques
- Un élève **suit** plusieurs groupes pédagogiques et y obtient une note
- Un élève **suit** plusieurs unités pédagogiques et y obtient une note
- Les GP et UP sont **enseignés** par un enseignant

2- Modèle Entité/Association (12/12)

Exemple : le modèle E-A



2- Modèle relationnel (1/26)

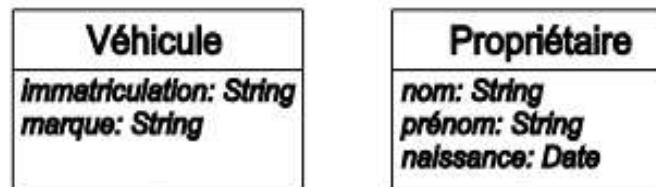
Du modèle Entité-Association vers le modèle relationnel

- Un modèle Entité-Association peut se traduire simplement en un modèle relationnel
 - ❖ Schéma de données plus formalisé
 - ❖ Possibilité d'utiliser l'algèbre relationnel pour exprimer des requêtes
- 3 règles simples de transformation

2- Modèle relationnel (2/26)

Règle 1

Toute entité est traduite par une relation
contenant les mêmes attributs et clés que l'entité



Se traduit par :

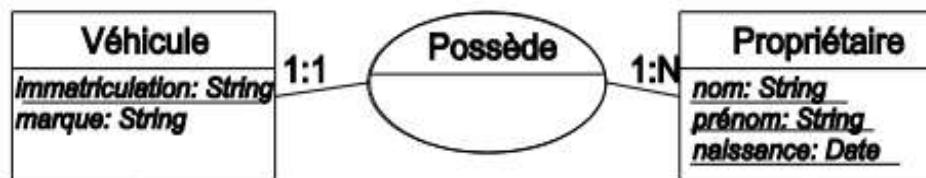
Vehicule(immatriculation: string, marque: string)

Proprietaire(nom: string, prenom: string, naissance: date)

2- Modèle relationnel (3/26)

Règle 2

Toute association depuis une entité R vers une entité R' ayant une cardinalité $0:1$ ou $1:1$ se traduit par l'ajout, dans la relation résultat de la traduction de R , de la clé de R' avec le statut d'attributs



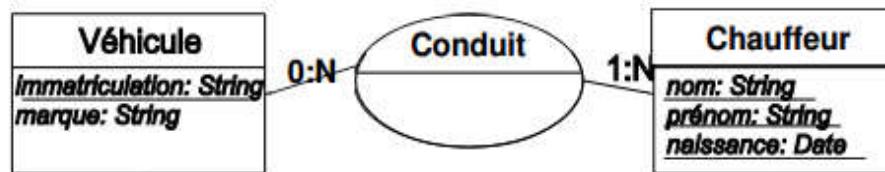
Se traduit par :

Vehicule(immatriculation: string, marque: string, nom_prop: string, prenom_prop: string, naissance_prop: date)

2- Modèle relationnel (4/26)

Règle 3

Toute association entre entités ayant des cardinalités **0:N** ou **1:N** à chaque extrémité se traduit par la création d'une relation contenant comme attributs les clés des entités associées ainsi que d'éventuels attributs de l'association. La clé de la relation créée est l'ensemble des attributs représentant les clés des entités associées.



Se traduit par :

Conduit (immatriculation: string, nom_cond: string, prenom_cond: string, naiss_cond: date)

2- Modèle relationnel (5/26)

Introduction de clés numériques

- Quand la clé d'une entité est composée de plusieurs attributs, il peut être efficace d'introduire une clé artificielle numérique
- Cette clé ne correspond à aucun attribut réel mais facilite les jointures et sélection

Ex: *Proprietaire(numero: int, nom: string, prenom: string, naissance: date)*

2- Modèle relationnel (6/26)

Cohérence des données

- Les relations définissent la structure des données mais pas la cohérence de contenu.
- Exemple de contenu incohérent : une même personne enregistrée avec deux dates de naissance différentes
- La gestion de la cohérence des données consiste à rendre impossible la saisie de données incohérente
- 2 concepts clés
 - ◆ Dépendances fonctionnelles
 - ◆ Formes normales

2- Modèle relationnel (7/26)

Dépendances fonctionnelles (1)

- Une dépendance fonctionnelle (DF) représente le fait qu'à la valeur d'un ou plusieurs attributs, on associe une valeur pour un autre attribut

Soit $R(A_1, A_2, \dots, A_n)$ et une DF de A_i vers A_j se note :

$$A_i \rightarrow A_j$$

- Par exemple, il existe une DF entre le numéro et le nom d'un propriétaire

$$\text{numero} \rightarrow \text{nom}$$

2- Modèle relationnel (8/26)

Dépendances fonctionnelles (2)

- Une DF $A \rightarrow B$ est dite élémentaire si un sous-ensemble de la clé n'est pas source de la DF

$$\forall a \in A, A - \{a\} \not\rightarrow B$$

- Une DF $A \rightarrow B$ est dite directe s'il n'existe pas d'ensemble d'attributs C tel que

$$A \rightarrow C \text{ et } C \rightarrow B$$

2- Modèle relationnel (9/26)

Le langage de définition de données (DDL) SQL

- Le LDD permet de spécifier le schéma d'une BD relationnelle.
- Ce langage correspond à une partie de la norme SQL
 - ▶ L'autre partie étant relative à la *manipulation des données* (LMD)
- La définition d'un schéma logique comprend trois parties:
 - ▶ Description des *tables* et de leur *contenu*.
 - ★ Attributs, types, etc.
 - ▶ Description des *contraintes* qui portent sur les données de la base.
 - ★ Contrôles sur l'intégrité des données qui s'imposent à toutes les applications accédant à cette base.
 - ▶ Description de la représentation physique.
 - ★ Les index, ...

2- Modèle relationnel (10/26)

Le langage de définition de données (DDL) SQL

• Types SQL.

- ▶ La norme SQL ANSI propose un ensemble de types.
- ▶ Tailles données à titre indicatif (peuvent varier selon les systèmes)

Type	Description	Taille
INTEGER	Type des entiers relatifs	4 octets
SMALLINT	Idem.	2 octets
BIGINT	Idem.	8 octets
FLOAT	Flottants simple précision	4 octets
DOUBLE PRECISION	Flottants double précision	8 octets
REAL	Synonyme de FLOAT	4 octets
NUMERIC (M, D)	Numérique avec précision fixe.	M octets
DECIMAL (M, D)	Idem.	M octets
CHAR(M)	Chaînes de longueur fixe	M octets
VARCHAR(M)	Chaînes de longueur variable	$L+1$ avec $L \leq M$
BIT VARYING	Chaînes d'octets	Longueur de la chaîne.
DATE	Date (jour, mois, an)	env. 4 octets
TIME	Horaire (heure, minutes, secondes)	env. 4 octets
DATETIME	Date et heure	8 octets
YEAR	Année	2 octets

- ▶ Chaîne très longue (txt): BIT VARYING. Variantes: BLOB ou LONG.

2- Modèle relationnel (11/26)

Le langage de définition de données (**DDL**) SQL

- **Création des tables.**

- ▶ La commande principale est : CREATE TABLE

```
CREATE TABLE Internaute (email VARCHAR (50) NOT NULL,  
                         nom VARCHAR (20) NOT NULL,  
                         prenom VARCHAR (20),  
                         motDePasse VARCHAR (60) NOT NULL,  
                         anneeNaiss DECIMAL (4));
```

- ▶ NOT NULL: L'attribut correspondant doit **toujours** avoir une valeur.
 - ★ On ne peut pas faire d'opération incluant un NULL. Ni une comparaison.
 - ★ Le SGBD rejettéra toute tentative d'insertion non conforme.
- ▶ Autre manière de forcer: valeur par défaut (option DEFAULT)
 - ★ Valeur par défaut: constante ou variable système (ex: SYSDATE)

```
CREATE TABLE Cinéma (nom VARCHAR (50) NOT NULL,  
                     adresse VARCHAR (50) DEFAULT 'Inconnue');
```

2- Modèle relationnel (12/26)

Le langage de définition de données (DDL) SQL

● **Contraintes d'intégrité. 1/12**

Voici les règles que l'on peut demander au système de garantir :

- ▶ Un attribut doit toujours avoir une valeur.
 - ★ Contrainte NOT NULL vue précédemment.
- ▶ Un attribut (ou un ensemble d'attributs) constitue la clé de la relation.
 - ★ Option PRIMARY KEY
- ▶ Un attribut dans une table est liée à la clé primaire d'une autre table.
 - ★ Clé étrangère (*intégrité référentielle*). FOREIGN KEY ... REFERENCES
- ▶ La valeur d'un attribut doit être unique au sein de la relation.
 - ★ *Clé secondaire*. Option UNIQUE
- ▶ Toute règle s'appliquant à la valeur d'un attribut (valeur *min* par ex).
 - ★ Contrainte de domaine. Option CHECK

2- Modèle relationnel (13/26)

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 2/12**

Clés d'une table

Une clé est un attribut (ou un ensemble d'attributs) qui identifie de manière unique un tuple. Il peut y avoir plusieurs clés mais l'une d'entre elles doit être choisie comme clé primaire (option PRIMARY KEY).

```
CREATE TABLE Internaute  (email VARCHAR (50) NOT NULL,  
                         nom VARCHAR (20) NOT NULL,  
                         prenom VARCHAR (20),  
                         motDePasse VARCHAR (60) NOT NULL,  
                         anneeNaiss DECIMAL (4),  
                         PRIMARY KEY (email));
```

- ▶ Il devrait toujours y avoir une PRIMARY KEY dans une table pour ne pas risquer d'insérer involontairement deux lignes strictement identiques.

2- Modèle relationnel (14/26)

Le langage de définition de données (DDL) SQL

- **Contraintes d'intégrité. 3/12**

Clés d'une table

Une clé peut être constituée de plusieurs attributs :

```
CREATE TABLE Notation (idFilm INTEGER NOT NULL,  
                      email VARCHAR (30) NOT NULL,  
                      note INTEGER DEFAULT 0,  
                      PRIMARY KEY (titre, email));
```

- ▶ Tous les attributs figurant dans une clé doivent être déclarés NOT NULL.

2- Modèle relationnel (15/26)

Le langage de définition de données (DDL) SQL

- **Contraintes d'intégrité. 4/12**

Clés d'une table

On peut également spécifier que la valeur d'un attribut est unique pour l'ensemble de la colonne.

- ▶ Cela permet d'indiquer des *clés secondaires*
- ▶ Ex: deux artistes ne peuvent pas avoir les mêmes nom et prénom

```
CREATE TABLE Artiste  (id INTEGER NOT NULL,  
                      nom VARCHAR (30) NOT NULL,  
                      prenom VARCHAR (30) NOT NULL,  
                      anneeNaiss INTEGER,  
                      PRIMARY KEY (id),  
                      UNIQUE (nom, prenom));
```

2- Modèle relationnel (16/26)

Le langage de définition de données (DDL) SQL

- **Contraintes d'intégrité. 5/12**

Clés d'une table

On peut également spécifier que la valeur d'un attribut est unique pour l'ensemble de la colonne.

- ▶ Un autre exemple d'utilisation d'une clé secondaire
 - ★ On ne peut pas trouver deux cinémas à la même adresse.

```
CREATE TABLE Cinéma (nomCinema VARCHAR (30) NOT NULL,  
                     adresse VARCHAR (50) UNIQUE,  
                     PRIMARY KEY (nomCinema));
```

- ▶ N'est possible que quand cette contrainte ne concerne qu'un attribut.
 - ★ $\text{UNIQUE}(x, y) \neq \text{UNIQUE}(x) \wedge \text{UNIQUE}(y)$.
- ▶ La clause UNIQUE ne s'applique pas aux valeurs NULL
 - ★ Il peut y avoir des cinémas d'adresse inconnue.

2- Modèle relationnel (17/26)

Le langage de définition de données (DDL) SQL

- **Contraintes d'intégrité. 6/12**

Clés étrangères

La norme SQL ANSI permet d'indiquer quelles sont les *clés étrangères* dans une table, autrement dit, quels sont les attributs qui font référence à une ligne dans une autre table.

- ▶ Option FOREIGN KEY ... REFERENCES.

```
CREATE TABLE Film    (idFilm INTEGER NOT NULL,  
                      titre VARCHAR (50) NOT NULL,  
                      annee INTEGER NOT NULL,  
                      idMES INTEGER,  
                      codePays INTEGER,  
                      PRIMARY KEY (idFilm),  
                      FOREIGN KEY (idMES) REFERENCES Artiste,  
                      FOREIGN KEY (codePays) REFERENCES Pays );
```

2- Modèle relationnel (18/26)

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 7/12**

Clés étrangères

La commande

```
FOREIGN KEY (idMES) REFERENCES Artiste
```

indique que idMES référence la clé primaire de la table *Artiste*.

- ★ Le SGBD vérifiera, pour toute modification pouvant affecter le lien entre les deux tables, que la valeur de idMES correspond bien à une ligne de *Artiste*. Ces modifications sont:
 1. L'insertion dans *Film* d'une valeur inconnue pour idMES ;
 2. La destruction d'un artiste ;
 3. La modification de id dans *Artiste* ou de idMES dans *Film*.
- ▶ Quand un attribut est à NULL (ex: idMES de *Film*), la contrainte d'intégrité référentielle ne s'applique pas.

2- Modèle relationnel (19/26)

Le langage de définition de données (DDL) SQL

● **Contraintes d'intégrité. 8/12**

Clés étrangères

- ▶ Que se passe-t-il quand la violation d'une contrainte d'intégrité est détectée par le système ?
 - ★ Par défaut: mise à jour rejetée.
 - ★ Il est aussi possible de demander la **répercussion** de cette mise à jour de manière à ce que la contrainte soit respectée.
- ▶ Les événements que l'on peut répercuter sont:
 - ★ La modification, désignée par ON UPDATE
 - ★ La destruction, désignée par ON DELETE
- ▶ La répercussion elle-même consiste
 - ★ Soit à mettre la clé étrangère à NULL (option SET NULL),
 - ★ Soit à appliquer la même opération aux lignes de l'entité composante (option CASCADE).

2- Modèle relationnel (20/26)

Le langage de définition de données (DDL) SQL

● **Contraintes d'intégrité. 9/12**

Clés étrangères

- ▶ Que se passe-t-il quand la violation d'une contrainte d'intégrité est détectée par le système ?
 - ★ **Exemple 1:** Comment indiquer que la destruction d'un metteur en scène déclenche la mise à NULL de la clé étrangère idMES pour tous les films qu'il a réalisés.

```
CREATE TABLE Film  (idFilm INTEGER NOT NULL,  
                    titre VARCHAR (50) NOT NULL,  
                    annee INTEGER NOT NULL,  
                    idMES INTEGER,  
                    codePays INTEGER,  
                    PRIMARY KEY (idFilm),  
                    FOREIGN KEY (idMES) REFERENCES Artiste  
                                ON DELETE SET NULL,  
                    FOREIGN KEY (codePays) REFERENCES Pays);
```

2- Modèle relationnel (21/26)

Le langage de définition de données (DDL) SQL

- **Contraintes d'intégrité. 10/12**

Clés étrangères

- ▶ Que se passe-t-il quand la violation d'une contrainte d'intégrité est détectée par le système ?
 - ★ Cas d'une entité faible: on décide en général de détruire le composant quand on détruit le composé.
 - ★ **Exemple 2:** Destruction d'un cinéma ⇒ Destruction de ses salles ; Modification clé d'un cinéma ⇒ Répercussion des modif. sur ses salles.

```
CREATE TABLE Salle  (nomCinema VARCHAR (30) NOT NULL,  
                     no INTEGER NOT NULL,  
                     capacite INTEGER,  
                     PRIMARY KEY (nomCinema, no),  
                     FOREIGN KEY (nomCinema) REFERENCES Cinema  
                         ON DELETE CASCADE,  
                         ON UPDATE CASCADE);
```

- ★ Aurions-nous pu faire **ON DELETE SET NULL** pour nomCinema ?

2- Modèle relationnel (22/26)

Le langage de définition de données (DDL) SQL

- **Contraintes d'intégrité. 11/12**

Contraintes de domaine: option CHECK

- ▶ Ex: Restriction des valeurs possibles pour année et genre.

```
CREATE TABLE Film    (idFilm INTEGER NOT NULL,  
                      titre VARCHAR (50) NOT NULL,  
                      annee INTEGER  
                        CHECK (annee BETWEEN 1890 AND 2000) NOT NULL,  
                      genre VARCHAR (10)  
                        CHECK (genre IN ('Histoire', 'Western')),  
                      idMES INTEGER,  
                      codePays INTEGER,  
                      PRIMARY KEY (idFilm),  
                      FOREIGN KEY (idMES) REFERENCES Artiste,  
                      FOREIGN KEY (codePays) REFERENCES Pays);
```

2- Modèle relationnel (23/26)

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 12/12**

- La spécification des actions (contraintes)

- ▶ ON DELETE
 - ▶ ON UPDATE
 - ▶ CHECK
 - ▶ FOREIGN KEY

simplifie considérablement la gestion de la base par la suite

- ▶ On n'a plus, par ex, à se soucier de faire des mises à jour en cascade.

- Si on n'opte pas pour la vérification automatique:

- ▶ Il faudra faire la vérification dans l'application cliente.
 - ▶ Cela est plus lourd à gérer.

2- Modèle relationnel (24/26)

Le langage de définition de données (DDL) SQL

- **Modification du schéma. 1/2**

La forme générale de la commande est :

—

ALTER TABLE *nomTable* ACTION *description*

—

où ACTION peut être principalement

- ▶ ADD,
- ▶ MODIFY,
- ▶ DROP,
- ▶ RENAME

et *description* est la commande de modification associée à ACTION

- La modification d'une table peut poser des problèmes. Par exemple, passer un attribut à NOT NULL implique que cet attribut a déjà des valeurs pour toutes les lignes de la table.

2- Modèle relationnel (25/26)

Le langage de définition de données (DDL) SQL

- **Modification du schéma. 2/2**

Modification des attributs

- ?

```
ALTER TABLE Internaute ADD region VARCHAR(10);
```

- ?

```
ALTER TABLE Internaute MODIFY region VARCHAR(30);
```

- ?

```
ALTER TABLE Internaute DROP region;
```

2- Modèle relationnel (26/26)

Le langage de définition de données (**DDL**) SQL

- Création d'index

▶ *S'il nous reste du temps ...*

3- Algèbre relationnelle (1/11)

Quelques définitions et opérateurs usuels

- L'**algèbre relationnelle** consacre un ensemble de symboles conventionnels permettant de (manipuler) lancer des opérations sur les tables d'une base de données. **Le résultat est toujours une relation (ensemble)**
- Une **expression** étant une opération ou encore une requête sollicitant du contenu de la base de donnée.
- Une **table** dans une base de données correspond à un ensemble d'occurrences (lignes ou enregistrements).

Comparaison : < ; <= ; > ; >= ; # ; =

Booléan : AND (\wedge) ; OR (\vee)

3- Algèbre relationnelle (2/11)

Types d'opérations et symboles associés

- **Unaires** : $\text{UnOp}(\text{Table}_1) \rightarrow \text{Table}_2$
sélection (σ : choix de lignes) Projection (π : choix de colonnes).

- **Binaires** : $(\text{Table}_1) \text{BinOp} (\text{Table}_2) \rightarrow \text{Table}_3$

Sur tables de la même structure :

... Union (\cup), Intersection (\cap), Différence (-)

Sur tables de structures différentes :

... Produit cartésien (\times), Jointure (\bowtie)

3- Algèbre relationnelle (3/11)

Sélection

- Filtrer (rechercher) des lignes dans une table selon une condition booléenne.

$\sigma_{\text{condition}} (\text{Table})$

Person

Name	Age	Address	Birthplace
James B.	47	London, UK	London, UK
Clark K.	35	Metropolis, USA	Krypton
Lois L.	28	Metropolis, USA	Metropolis, USA
Tarzan	25	London, UK	Forest, Benin

- Personnes dont l'adresse est London, UK

$\sigma_{\text{Address}=\text{"London, UK"}}(\text{Person})$

Name	Age	Address	Birthplace
James B.	47	London, UK	London, UK
Tarzan	25	London, UK	Forest, Benin

Question : Les personnes vivant à leurs lieux de naissance???

3- Algèbre relationnelle (4/11)

Projection

- Filtrer (choisir) des colonnes dans une table.

$\pi_{\text{column(s)}} (\text{Table})$

Person

Name	Age	Address	Birthplace
James B.	47	London, UK	London, UK
Clark K.	35	Metropolis, USA	Krypton
Lois L.	28	Metropolis, USA	Metropolis, USA
Tarzan	25	London, UK	Forest, Benin

- Nom et âge des personnes

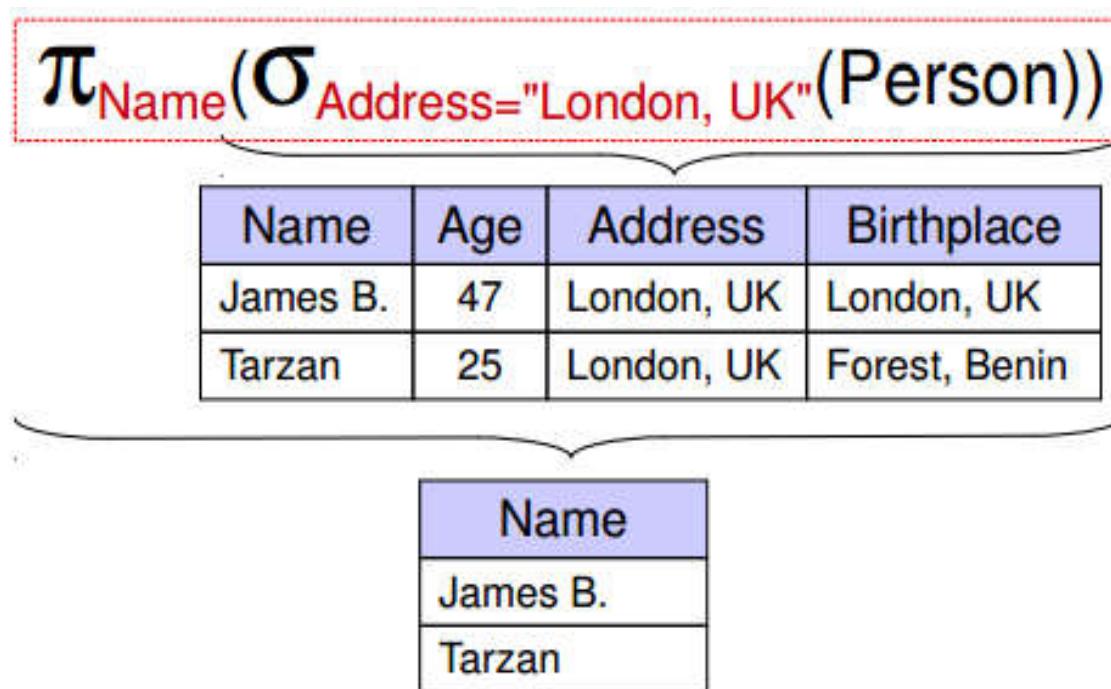
$\pi_{\text{Name, Age}}(\text{Person})$

Name	Age
James B.	47
Clark K.	35
Lois L.	28
Tarzan	25

3- Algèbre relationnelle (5/11)

Projection (conditionnée)

- ***Exemple*** : les noms des personnes qui habitent à **London, UK**.



3- Algèbre relationnelle (6/11)

Produit cartésien

- Chaque ligne de la table1 sera combinée avec toutes les occurrences de la table2.

Table ₁ \otimes Table ₂			
Man	Name	Address	Woman
	James B.	London	
	Clark K.	Metropolis	
	Tarzan	London	

Man \otimes Woman

Name ₁	Address ₁	Name ₂	Address ₂
James B.	London	Lois L.	Metropolis
James B.	London	Catwoman	Gotham City
Clark K.	Metropolis	Lois L.	Metropolis
Clark K.	Metropolis	Catwoman	Gotham City
Tarzan	London	Lois L.	Metropolis
Tarzan	London	Catwoman	Gotham City

3- Algèbre relationnelle (7/11)

Produit cartésien (conditionné)

- Paires **hommes-femmes** habitant à la même adresse. (**Quelles sont les tables de départ ?**)

$$\sigma_{\text{Address}_1=\text{Address}_2}(\text{Man} \bowtie \text{Woman})$$

Name ₁	Address ₁	Name ₂	Address ₂
James B.	London	Lois L.	Metropolis
James B.	London	Catwoman	Gotham City
Clark K.	Metropolis	Lois L.	Metropolis
Clark K.	Metropolis	Catwoman	Gotham City
Tarzan	London	Lois L.	Metropolis
Tarzan	London	Catwoman	Gotham City

Name ₁	Address ₁	Name ₂	Address ₂
Clark K.	Metropolis	Lois L.	Metropolis

3- Algèbre relationnelle (8/11)

Jointure

- La jointure est un produit cartésien assortie d'une sélection. Le critère de faisabilité est que les tables en jeu doivent avoir au moins un domaine de valeurs identique (attribut pas forcément du même nom).
- Soit deux tables **A et B**, le résultat de **leur jointure est une relation qui combine toutes les occurrences** (enregistrements) de la table **A avec leur(s) correspondance(s)** dans table **B** sur la base du **domaine de valeur identifié**.

3- Algèbre relationnelle (9/11)

Jointure

Table₁ \bowtie Table₂
condition

Man

Name	Address
James B.	London
Clark K.	Metropolis
Tarzan	London

City

CName	Population	Country
London	7,556,900	UK
Metropolis	8,391,881	USA

- Dans quel pays habite les hommes de la table Man ?

$$\text{Man} \bowtie \text{City} \Leftrightarrow \sigma_{\text{Address}=\text{CName}}(\text{Man} \otimes \text{City})$$

Address=CName

Name	Address	CName	Population	Country
James B.	London	London	7,556,900	UK
Clark K.	Metropolis	Metropolis	8,391,881	USA
Tarzan	London	London.	7,556,900	UK

3- Algèbre relationnelle (10/11)

Union

Man

Name	Address
James B.	London
Clark K.	Metropolis
Tarzan	London

Woman

Name	Address
Lois L.	Metropolis
Catwoman	Gotham City

Man \cup Woman

Name	Address
James B.	London
Clark K.	Metropolis
Tarzan	London
Lois L.	Metropolis
Catwoman	Gotham City

$\sigma_{\text{Address}=\text{"Metropolis"}}(\text{Man} \cup \text{Woman})$

Name	Address
Clark K.	Metropolis
Lois L.	Metropolis

3- Algèbre relationnelle (11/11)

Différence

Man

Name	Address
James B.	London
Clark K.	Metropolis
Tarzan	London

Woman

Name	Address
Lois L.	Metropolis
Catwoman	Gotham City

Adresse chez les femmes pas présentes (ou aucun homme habite) chez les hommes ?

$$\pi_{\text{Address}}(\text{Woman}) - \pi_{\text{Address}}(\text{Man})$$

Address
Gotham City

Chapitre 3

Langage d'interrogation des bases de données (SQL)

Le langage SQL

Le langage de manipulation de données (DML) SQL

- Langage d'interrogation et de manipulation de données
 - ▶ Insertion, mise-à-jour, destruction.
 - ▶ Norme SQL2, implantée +/- complètement dans la plupart des SGBDR.
- Langage *déclaratif*^a qui permet d'interroger la BD sans se soucier de
 - ▶ La représentation interne (physique) des données
 - ▶ Leur localisation,
 - ▶ Des algorithmes.
- S'adresse à une large communauté (pas seulement informaticienne)
- Peut être utilisé:
 - ▶ De manière interactive
 - ▶ En association avec des langages de programmation.

^aVoilà ce que je veux, débrouille-toi pour le faire.

Le langage SQL

Le langage de manipulation de données (DML) SQL

• Requêtes simples SQL

- ▶ Sélections simples - Les stations se trouvant aux Antilles

```
SELECT nomStation  
      FROM Station  
     WHERE region = 'Antilles'
```

- SELECT: Liste des attributs constituant le résultat.
- FROM: Indique les tables dans lesquelles ont trouvé les attributs utiles
 - ★ Attribut dont on souhaite afficher le contenu.
 - ★ Attribut dont on souhaite qu'il ait une valeur particulière.
- WHERE: Conditions que doivent satisfaire les n-uplets du résultat.

nomStation
Venusa
Santalba

- Utilisation combinée de la sélection (σ) et de la projection (π).

Le langage SQL

Le langage de manipulation de données (DML) SQL

• Requêtes simples SQL

- ▶ Sélections simples
- Fonctions applicables aux valeurs des attributs
 - ★ Opérateurs arithmétiques (+, -, *, /, ...) pour les attributs numériques
 - ★ Manipulation de chaînes de caractères: concaténation, sous chaîne, mise en majuscule, ...

```
SELECT libelle, prix / 6.56, 'Cours de 1\'euro = ', 6.56
FROM Activite
WHERE nomStation = 'Santalba'
```

libelle	?column?	?column?	?column?
Kayac	7.62	Cours de l'euro	6.56

- ★ Sous certains systèmes, les noms des attributs du résultat sont par défaut ceux indiqués dans la clause SELECT.

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**
 - ▶ Sélections simples

Renommage

```
SELECT    libelle, prix / 6.56 AS prixEnEuros,  
          'Cours de l\'euro = ' AS cde, 6.56 AS cours  
FROM      Activite  
WHERE     nomStation = 'Santalba'
```

libelle	prixEnEuros	cde	cours
Kayac	7.62	Cours de l'euro	6.56

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**

- ▶ **Doublons**

- ★ SQL permet l'existence de doublons (\neq de l'algèbre relationnelle).
 - ★ Les clés permettent d'éviter les doublons dans les relations stockées.

```
SELECT    libelle  
FROM      Activite
```

libelle
Voile
Plongee
Plongee
Ski
Piscine
Kayac

- ▶ Élimination des doublons

```
SELECT    DISTINCT libelle    FROM Activite
```

Le langage SQL

Le langage de manipulation de données (DML) SQL

● Requêtes simples SQL

► Tri du résultat

- ★ Clause ORDER BY suivie de la liste des attributs servant de critère au tri

```
SELECT      *
FROM        Station
ORDER BY    tarif, nomStation
```

- Trie en ordre ascendant les stations par leurs tarifs
- Pour un même tarif, présente les stations selon l'ordre lexicographique.
- Tri par **ordre descendant**: mot clé DESC après la liste des attributs
- Le caractère * dans le SELECT: intégralité des colonnes.
 - ★ Équivaut à lister tous les attributs

Le langage SQL

Le langage de manipulation de données (DML) SQL

• Requêtes simples SQL

- ▶ La clause WHERE
 - ★ Condition booléenne portant sur les attributs des relations du FROM
- ▶ Connecteurs logiques: AND, OR, NOT
- ▶ Opérateurs de comparaison: <, <=, >, <=, <> (!=), BETWEEN, IN ...

```
SELECT nomStation, libelle
FROM Activite
WHERE nomStation = 'Santalba'
AND (prix > 50 AND prix < 120)
```

```
SELECT nomStation, libelle
FROM Activite
WHERE nomStation = 'Santalba'
AND prix BETWEEN 50 AND 120
```

Le langage SQL

Le langage de manipulation de données (DML) SQL

● Requêtes simples SQL

- ▶ Chaînes de caractères
- ▶ Différence entre chaînes de longueur fixe et de longueur variable.
 - ★ Les premiers sont complétées par des blancs (' ') et pas les secondes.
- ▶ SQL distingue les majuscules des minuscules pour les chaînes.
 - ★ 'SANTALBA' ≠ 'Santalba'
- ▶ Clause **LIKE**: recherche par motif (*pattern matching*)
 - ★ Le caractère '_' désigne n'importe quel caractère
 - ★ Le caractère '%' désigne n'importe quelle chaîne de caractères.

```
SELECT nomStation  
FROM Station  
WHERE nomStation LIKE '%a'
```

Stations dont le nom se termine par 'a'.

```
SELECT nomStation  
FROM Station  
WHERE nomStation LIKE 'V____'
```

Commence par 'V' et comprend 6 caractères.

Le langage SQL

Le langage de manipulation de données (DML) SQL

• Requêtes simples SQL

- ▶ Dates
- ▶ Possibilité de manipulation de dates.
- ▶ Spécification de date en SQL2:
 - ★ Mot-clé DATE
 - ★ Suivi d'une chaîne de caractère au format 'aaaa-mm-jj'
- ▶ Exemple de date: DATE '1998-01-01'
- ▶ Requête: *Id des clients qui ont commencé un séjour en juillet 1998*

```
SELECT    idClient
FROM      Sejour
WHERE     debut BETWEEN DATE '1998-07-01' AND DATE '1998-07-31'
```

- ▶ Beaucoup de fonctions proposées par les systèmes:
 - ★ Calcul d'écart de dates, ajout de mois, d'années, etc.

Le langage SQL

Le langage de manipulation de données (DML) SQL

● Requêtes simples SQL

▶ Valeurs nulles (1/3)

- ★ SQL admet que la valeur de certains attributs soit inconnue
- ★ On parle de *valeur nulle*, désignée par le mot clé **NULL**.
- ▶ La *valeur nulle* n'est pas une valeur, mais une absence de valeur !
 - ★ Tout opération appliquée à NULL donne pour résultat NULL.
 - ★ Toute comparaison avec NULL donne un résultat qui n'est ni vrai, ni faux, mais une troisième valeur booléenne **UNKNOWN**.
- ▶ Supposons que les définitions $\text{TRUE} = 1$, $\text{FALSE} = 0$, $\text{UNKNOWN} = 1/2$
 - ★ $x \text{ AND } y = \min(x, y)$
 - ★ $x \text{ OR } y = \max(x, y)$
 - ★ $\text{NOT } x = 1 - x$

Le langage SQL

Le langage de manipulation de données (DML) SQL

● Requêtes simples SQL

► Valeurs nulles (2/3)

- ★ Exemple d'une table SEJOUR avec des informations manquantes

SEJOUR			
idClient	station	debut	nbPlaces
10	Passac	1998-07-01	2
20	Santalba	1998-08-03	
30	Passac		3

- ★ La présence de NULL peut avoir des effets surprenants. La requête:

```
SELECT    station
FROM      Sejour
WHERE     nbPlaces <= 10 OR nbPlaces >= 10
```

- ★ Devrait ramener toutes les stations de la table.
- ★ 'Santalba' ne figurera pas dans le résultat car nbPlaces est NULL.

Le langage SQL

Le langage de manipulation de données (DML) SQL

• Requêtes simples SQL

► Valeurs nulles (3/3)

- ★ Autre piège: `NULL` est un mot-clé, pas une constante
- ★ Une comparaison comme `nbPlaces = NULL` est incorrecte
- ★ Prédicat pour tester l'absence de valeur: `IS NULL`

Ex: *Quels sont les séjours dont on connaît le nombre de places.*

```
SELECT *
FROM   Sejour
WHERE  nbPlaces IS NOT NULL
```

- ★ La présence de `NULL` est une source de problèmes: dans la mesure du possible il faut l'éviter en spécifiant la contrainte `NOT NULL` ou en donnant une valeur par défaut.

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes sur plusieurs tables**

- ★ Manipulation simultanée de plusieurs tables
- ★ Expression des opérations binaires de l'algèbre rel.: \bowtie , \times , \cup , \cap , $-$

- ▶ **Jointures**

- ★ "*Nom des clients avec le nom des stations où ils ont séjourné*"
- ★ Attributs répartis dans les tables Client et Sejour.

```
SELECT    nom, station
FROM      Client, Sejour
WHERE     id = idClient
```

Le langage SQL

Le langage de manipulation de données (DML) SQL

● Requêtes sur plusieurs tables

▶ Jointures

- ★ Problèmes d'ambiguïté
- ★ Attribut partagé par plusieurs tables impliquées dans la jointure
- ★ Sol. 1: préfixation de l'attribut par le nom de la table

```
SELECT  Station.nomStation, tarif, libelle, prix
FROM    Station, Activite
WHERE   Station.nomStation = Activite.nomStation
```

- ★ Sol. 2: Association d'un synonyme, et utilisation en tant que préfixe.

```
SELECT  S.nomStation, tarif, libelle, prix
FROM    Station S, Activite A
WHERE   S.nomStation = A.nomStation
```

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes sur plusieurs tables**

- ▶ **Jointures**

- ★ Problèmes d'ambiguïté

Sol. 2 (Utilisation de synonymes): Indispensable dans certaines situations: jointure d'une relation avec elle-même

- ★ Exemple: Couples de stations situées dans la même région.

```
SELECT  S1.nomStation, S2.nomStation  
FROM    Station S1, Station S2  
WHERE   S1.region = S2.region
```

Le langage SQL

- **Requêtes sur plusieurs tables**

- ▶ **Union, intersection et différence UNION**

- ★ **Conditions:** même nombre de colonnes et même type d'attributs
 - *Tous les noms de région dans la base*

```
SELECT      region    FROM     Station  
UNION  
SELECT      region    FROM     Client
```

- *Régions où l'on trouve à la fois des clients et des stations*

```
SELECT      region    FROM     Station  
INTERSECT  
SELECT      region    FROM     Client
```

- *Régions où l'on trouve des stations, mais pas de clients*

```
SELECT      region    FROM     Station  
EXCEPT  
SELECT      region    FROM     Client
```

Le langage SQL

- **Requêtes imbriquées**

- ▶ Expression de conditions sur des *relations*.
- ▶ **Exemple:** *Noms des stations où ont séjourné des clients parisiens*
 - ★ Résultat avec une jointure classique.

```
SELECT    station
FROM      Sejour, Client
WHERE     id = idClient AND ville = 'Paris'
```

- ★ Résultat avec une requête imbriquée.

```
SELECT    station
FROM      Sejour
WHERE     idClient IN  (SELECT id FROM Client
                        WHERE ville = 'Paris')
```

- **IN** peut être remplacé par **=** si **on est sûr que la sous-requête ramène un et un seul tuple**.

Le langage SQL

- **Requêtes imbriquées**

- ▶ Conditions que l'on peut exprimer sur une relation R construite avec une requête imbriquée (peuvent être préfixée par **NOT** pour la négation):

① **EXIST R**

- ▶ Renvoie TRUE si R n'est pas vide, FALSE sinon.

② $t \text{ IN } R$

- ▶ Où t est un tuple de même type que R . TRUE si $t \in R$, FALSE sinon.

③ $v \text{ cmp ANY } R$

- ▶ Où cmp est un comparateur SQL ($<$, $>$, $=$, ...). Renvoie TRUE si la comparaison avec *au moins un* des tuples de la relation R renvoie TRUE.

④ $v \text{ cmp ALL } R$

- ▶ Où cmp est un comparateur SQL ($<$, $>$, $=$, ...). Renvoie TRUE si la comparaison avec *tous* les tuples de la relation R renvoie TRUE.

Le langage SQL

• Requêtes imbriquées

★ *Où (station, lieu) ne peut-on pas faire du ski ?*

```
SELECT nomStation, lieu
FROM Station
WHERE nomStation NOT IN (SELECT nomStation FROM Activite
                           WHERE libelle = 'Ski')
```

★ *Quelle station pratique le tarif le plus élevé ?*

```
SELECT nomStation
FROM Station
WHERE tarif >= ALL (SELECT tarif FROM Station)
```

★ *Station où l'on pratique une activité au même prix qu'à Santalba ?*

```
SELECT nomStation, libelle
FROM Activite
WHERE prix IN (SELECT prix FROM Activite
                WHERE nomStation = 'Santalba')
```

Le langage SQL

• Agrégation

- ▶ Expression de conditions sur des *groupes de tuples*.
- ▶ Constitution du résultat par *agrégation de valeurs* dans chaque groupe.

SQL fournit:

- ★ Le moyen de partitionner une relation en *groupes* selon certains critères.
- ★ Le moyen d'exprimer des conditions sur ces groupes
- ★ Des fonctions d'agrégation.

Il existe un groupe par défaut: c'est la relation toute entière.

▶ Fonctions d'agrégation

Fonctions qui s'appliquent à une colonne, en général de type numérique

- ★ COUNT qui compte le nombre de valeurs *non nulles*.
- ★ MAX et MIN.
- ★ AVG qui calcule la moyenne des valeurs de la colonne.
- ★ SUM qui effectue le cumul.

Le langage SQL

• Agrégation

► Fonctions d'agrégation - Exemples.

```
SELECT COUNT(nomStation), AVG(tarif), MIN(tarif), MAX(tarif)
FROM Station
```

- **Remarque:** on ne peut pas utiliser simultanément dans la clause **SELECT** des fonctions d'agrégation et des noms d'attributs (sauf dans le cas d'un **GROUP BY**).
- Pourquoi la requête suivante est-elle incorrecte ?

```
SELECT nomStation, AVG(tarif)
FROM Station
```

- *Combien de places a réservé Mr Kerouac pour l'ensemble des séjours ?*

```
SELECT SUM(nbPlaces)
FROM Client, Sejour
WHERE nom = 'Kerouac' AND id = idClient
```

Le langage SQL

• Agrégation

► La clause GROUP BY

- ★ Construction de groupes en associant les tuples partageant la même valeur pour une ou plusieurs colonnes.
- ★ Afficher les régions avec le nombre de stations

```
SELECT      region, COUNT(nomStation)
FROM        Station
GROUP BY    region
```

- Il n'est pas nécessaire de faire figurer tous les attributs du GROUP BY dans la clause SELECT
 - ★ On souhaite consulter le nombre de places réservées par client.

```
SELECT      nom , SUM(nbPlaces)
FROM        Client, Sejour
WHERE       id = idClient
GROUP BY    id, nom
```

Le langage SQL

● Agrégation

► La clause HAVING

- ★ Pour faire porter des *conditions sur les groupes*.
- ★ **WHERE** ne peut exprimer des conditions que sur les tuples pris un à un.
- ★ **HAVING** est pour le groupe ce que **WHERE** est pour le tuple.

- ★ *On souhaite consulter le nombre de places réservées, par client, pour les clients ayant réservé plus de 10 places.*

```
SELECT      nom , SUM(nbPlaces)
FROM        Client, Sejour
WHERE       id = idClient
GROUP BY    id, nom
HAVING     SUM(nbPlaces) >= 10
```

- ★ La condition porte sur une propriété de l'ensemble des tuples du groupe.
- ★ La clause HAVING est exprimée sur le résultat de fonctions d'agrégation.

Le langage SQL

- **Mises-à-jour**

- ▶ **Insertion**

- ★ Elle s'effectue avec la commande **INSERT**. Syntaxe:

```
INSERT INTO R(A1, A2, ..., An) VALUES (v1, v2, ..., vn)
```

- R: nom d'une relation.
 - A1, A2, ... An sont les noms des attributs dans lesquels on veut placer une valeur. Les autres attributs seront donc à NULL (ou à la valeur par défaut). Tous les attributs spécifiés NOT NULL (et sans valeur par défaut) doivent figurer dans la clause SELECT.
 - v1, v2, ..., vn sont les valeurs des attributs.

```
INSERT INTO Client (id, nom, prenom) VALUES (40, 'Moriarty', 'Dean')
```

```
INSERT INTO R VALUES (v1, v2, ..., vn)
```

```
INSERT INTO Sites (lieu, region)  
SELECT lieu, region FROM Station
```

Le langage SQL

- **Mises-à-jour**

- ▶ **Destruction**

- ★ Elle s'effectue avec la clause DELETE. Syntaxe:

```
DELETE    FROM R  
WHERE     condition
```

- ★ *Destruction de tous les clients dont le nom commence par 'M'*

```
DELETE    FROM Client  
WHERE     nom LIKE 'M%'
```

Le langage SQL

- **Mises-à-jour**
 - ▶ **Modification**

- ★ Elle s'effectue avec la clause UPDATE. Syntaxe:

```
UPDATE    R SET A1=v1, A2=v2, ..., An=vn  
WHERE      condition
```

- ★ Augmenter le prix des activités de la station Passac de 10%

```
UPDATE    Activite  
SET        prix = prix × 1.1  
WHERE      nomStation = 'Passac'
```

Remarque importante:

- ★ Toutes les mises-à-jour ne deviennent définitives qu'à l'issue d'une validation par *commit*.
 - ★ Elles peuvent être annulées par *rollback*

FIN DU COURS

Documentation

Dr. CHEIKH BA (SI-BD)



Cours Master MIAGE 2012

&

Antoine Zimmermann (ACSI)



adapté du cours de Laurent Vercouter