Pr. Youness KHOURDIFI, PhD en Informatique
Professeur à la Faculté Polydisciplinaire – Khouribga –
Université Sultan Moulay Slimane – Béni Mellal –
Consultant IT : SQL 2016 Database Administration, Core Infrastructure 2016, Azure Solutions Architect Expert, Data Analyst Associate, Ingénieur DevOps.
y.khourdifi@usms.ma

BASES DE DONNÉES

Année Universitaire : 2023/2024

Scannez ce code QR pour nous rejoindre dans le cours des bases de données

Règle du cours

- Respect de l'horaire
- Respect des deadlines de remises des comptes rendus
- Respect des deadlines de remises des projets

Système de notation

- Contrôles continus → 60%
- Travaux pratiques → 40%

PLAN

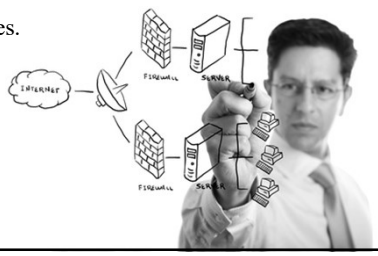
- Introduction générale
- Introduction aux systèmes d'information et bases des données :
  - Présentation de la méthode Merise;
  - Bases de données relationnelles;
- Les Systèmes de Gestion des Bases des Données (SGBD);
- Les principaux modèles de données :
  - Modèle conceptuel de données: MCD;
  - Modèle logique de données: MLD;
  - Le modèle relationnel :
    - Schéma de la base de donnée;
- Langage SQL (Structured Query Language);
- Conclusion et Evaluation.

Chapitre 0 Introduction générale

Chapitre 0 Introduction générale

Introduction générale :

- Dans l'activité de chaque organisation, on utilise une quantité importante d'informations.
- Pour être utilisables pour la prise des décisions, ces informations doivent être stockées, diffusées, traitées et transformées.
- Toute entreprise utilise de l'information pour son fonctionnement ou pour communiquer avec son environnement.
- Le gestionnaire doit être capable de traiter (ou accompagner le traitement) l'information et la rendre utile et rentable pour l'entreprise.
- Lorsqu'on parle d'informatisation, les systèmes d'information sont incontournables.
- Il est normal que dans une filière qui concerne l'informatisation, les systèmes d'informations soient largement étudiés.



Introduction générale :

9

Pyramide de l'information (Rainer & al., 2009) :

- La pyramide DICS, de l'anglais DIKW pour data, information, knowledge et wisdom), également connue sous le nom de hiérarchie DICS, hiérarchie de la sagesse, hiérarchie des connaissances, hiérarchie de l'information, pyramide de l'information et pyramide des données, se réfère à une classe de modèles représentant une liaison hiérarchique, sous forme pyramidale, entre les données, l'information, la connaissance et la sagesse.

- En règle générale, l'information est définie en termes de données, la connaissance en termes d'informations et la sagesse en termes de connaissances.

Introduction générale :

10

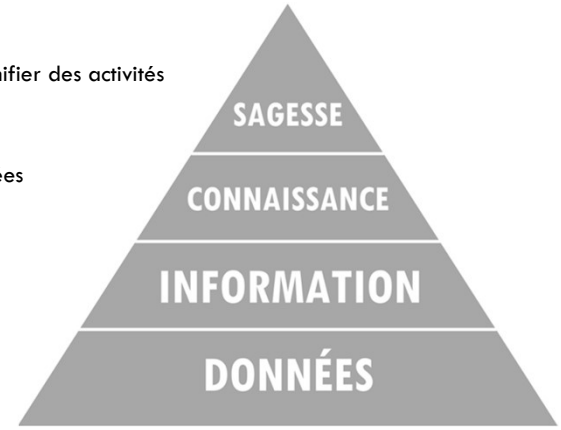
Pyramide de l'information (Rainer & al., 2009) :

En utilisant la connaissance des tendances climatiques pour planifier des activités agricoles saisonnières de manière optimale.

Une analyse des tendances climatiques sur plusieurs années basée sur les données de température.

Une liste de chiffres avec une étiquette "Température quotidienne en degrés Celsius"

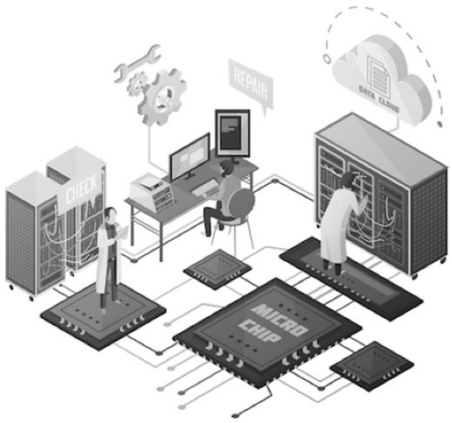
Une liste de chiffres - 1, 2, 3, 4, 5.



Introduction générale :

11

Le système d'information (SI) c'est l'ensemble des ressources de l'entreprise qui permettent la gestion de l'information. Le SI est généralement associé aux technologies (matériel, logiciel et communication), aux processus qui les accompagnent, et aux hommes qui les supportent.



Introduction générale :

12

Système Informatique (IT)

Système d'Information (SI)

Un système informatique (Information Technology IT) est un ensemble de matériels (hardware) et de logiciels (software) permettant le traitement automatique de l'information.

Source Wikipédia

Le Système Informatique est l'ensemble des actifs matériels et logiciels de l'entreprise ayant pour vocation à automatiser le traitement de l'information. C'est la partie visible à laquelle tout le monde pense quand on parle de projets et d'infrastructures informatiques.

Le système d'information (SI) est un ensemble organisé de ressources qui permet de collecter, stocker, traiter et distribuer de l'information.

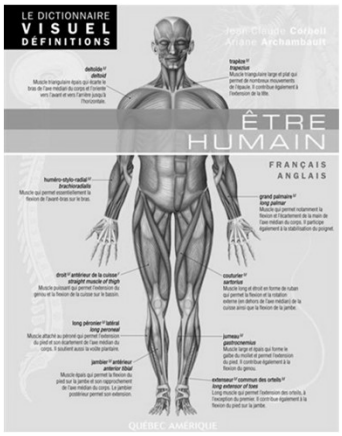
Source Wikipédia

Le Système d'Information se construit dans le contexte du Système Informatique, et consiste en les applications spécialisées, les règles de gestion, les traitements, les données, les éditions et affichages, qui fonctionnent ensemble pour produire des résultats souhaités pour atteindre certains objectifs du métier.

Introduction générale :

13

Quel est le plus ancien système d'information autonome ??



l'information doit être :

Collecter ;

Stocker ;

Traiter ;

Distribuer.

Chapitre 1 Introduction aux systèmes d'information et bases des données

14

I . Introduction aux systèmes d'information et bases des données :

15

Objectifs

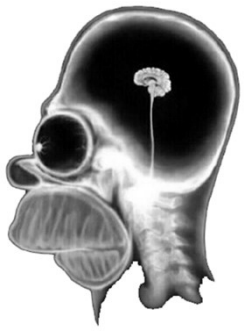
- Analyser un système d'information sur le plan informationnel, organisationnel et technique ;
- Comprendre l'interaction des systèmes d'information avec la stratégie, l'organisation et la culture de l'entreprise;
- Connaître le cycle de vie et les phases de mise en place d'un système d'information;
- Intégrer un logiciel au sein d'un SI existant ;
- Employer une méthode professionnelle d'ingénierie de conception des systèmes d'information.

I . Introduction aux systèmes d'information et bases des données :

16

Analogie avec systèmes biologiques :

- Le SI peut être comparé à une sorte de système nerveux primaire de l'organisation
- Circulation rapide d'une information de qualité entre les différents « organes »
- Délivrer la bonne information, au bon interlocuteur, au bon moment
  - Prise de décisions appropriées
  - Action de l'entreprise adaptée à la situation
- Le SI contribue donc de manière évidente aux performances de l'organisation



I . Introduction aux systèmes d'information et bases des données :

Introduction

Le concept de système d'information né dans les domaines de l'informatique et des télécommunications, et s'applique maintenant à l'ensemble des organisations.

Le système d'information coordonne grâce à l'information les activités de l'organisation et lui permet ainsi d'atteindre ses objectifs. Il est le véhicule de la communication dans l'organisation.

De plus, le système d'information représente l'ensemble des ressources (les hommes, le matériel, les logiciels) organisées pour : collecter, stocker, traiter et communiquer les informations.

- Un système d'information (SI) représente l'ensemble des éléments participant à la gestion, au traitement et à la diffusion de l'information au sein de l'organisation.

I . Introduction aux systèmes d'information et bases des données :

Analyse Systémique de l'Entreprise :

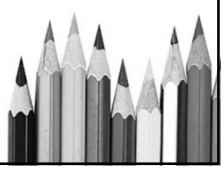
Avant l'année 1970

- L'entreprise était considérée comme une addition de services aux fonctions délimitées
- Les employés perçoivent cela comme ayant parfois des visées contradictoires, voire antagonistes

Apparue dans les années 1970

Entreprise = Système

- « Ensemble d'éléments en interaction dynamique, organisé en fonction d'un **but** »
  Joël De Rosnay « Le macroscope », éditions du seuil, 1975
- L'entreprise est alors considérée comme un ensemble d'éléments (des moyens humains, matériels, financiers et techniques) **en interrelations**
- Toute organisation humaine (l'État, une famille, …) peut être perçue comme un système



I . Introduction aux systèmes d'information et bases des données :

Analyse Systémique de l'Entreprise :

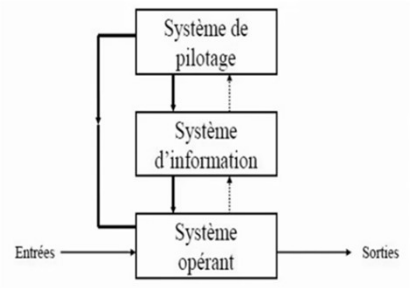
- Comme tout système, l'entreprise est un système :
  - Ouvert sur l'environnement
  - Il est finalisé (but = profit…)
  - Il est en constante évolution
- Pour parvenir à son but, le système tient compte de son environnement et **régule** son fonctionnement en **s'adaptant** aux changements
- Les éléments du système sont eux-mêmes des systèmes (ou sous-systèmes)

I . Introduction aux systèmes d'information et bases des données :

Vision globale d'une entreprise :

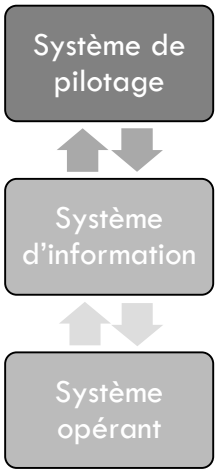
Une organisation (Système d'entreprise) est composée de trois systèmes :

Le système opérant qui constitue la machine proprement dite de production et de transformation des entrées en produits finis, le système de pilotage appelé aussi système de gestion qui pilote l'organisation et constitue son cerveau pensant et enfin le système d'information.



I . Introduction aux systèmes d'information et bases des données :

Vision globale d'une entreprise :



Activité :
- Réfléchir : adaptation à l'environnement, conception
- Décider : prévisions, allocation, planification
- Contrôler : qualité

Activité :
- Générer des informations
- Mémoriser
- Diffuser
- Traiter

Activité :
- Transformer
- Produire.

I . Introduction aux systèmes d'information et bases des données :

Le système de pilotage :
(appelé également système de décision)

- ❑ Exploite les informations qui circulent
- ❑ Organise le fonctionnement du système
- ❑ Décide des actions à conduire sur le système opérant
- ❑ Raisonne en fonction des objectifs et des politiques de l'entreprise

I . Introduction aux systèmes d'information et bases des données :

Le système opérant :

- Reçoit les informations émises par le système de pilotage
- Se charge de réaliser les tâches qui lui sont confiées
- Génère à son tour des informations en direction du système de pilotage
  - Qui peut ainsi contrôler les écarts et agir en conséquence
- Il englobe toutes les fonctions liées à l'activité propre de l'entreprise :
  - Facturer les clients, régler les salaires, gérer les stocks, …

I . Introduction aux systèmes d'information et bases des données :

Le système d'information :

- Pour organiser son fonctionnement, le système a besoin de mémoriser des informations
  - Pour comparer, prévoir, …
- Ce rôle est joué par le Système d'Information
- Ce système a aussi la charge de :
  - Diffuser l'information
  - Réaliser tous les traitements nécessaires au fonctionnement du système

I . Introduction aux systèmes d'information et bases des données :

Définition :

Un système est un **ensemble d'éléments** (matériels ou non) transformant des **éléments d'entrées** en **éléments de sorties** que l'on considère comme des **flux**.

Exemple :

Une entreprise qui commercialise une liste de produits:

- **Flux d'entrées**: Les produits achetés, commandes, paiements,
- **Flux de sortie**: Les produits vendus, factures, paiements.

Notions de Système d'Information (SI) :

*Le SI d'une entreprise est l'ensemble des **informations** circulant dans l'entreprise, des **moyens** et **méthodes** mises en œuvre pour les gérer.*

Système d'information = Humain + Technologie + Processus

I . 1. Présentation de la méthode Merise

Les informations :

- ❑ Écrite: lettres, factures, fiches techniques …
- ❑ Picturale: dessins , schémas de bâtiment, graphiques, photographies …
- ❑ Orale: discussions, conversations téléphoniques,
- ❑ Autres formes …

Les moyens :

- ❑ Humains : Personnes qui reçoivent, manipulent et émettent l'information,
- ❑ Matériels : Machines permettant de recevoir, conserver, manipuler et émettre de l'information (machine à écrire, photocopieur, télécopieurs, ordinateur, réseaux, …)

I . 1. Présentation de la méthode Merise

Méthodes :

- Outils et règles de travail :
  - Les modèles, les modes opératoires
  - Les algorithmes, les programmes et logiciels …

Notions de méthodes :

Besoin en méthode :

Pour concevoir le SI d'une entreprise, on doit faire appel à une méthode d'analyse pour les raisons suivantes :

- Complexité de la structure de données,
- Volume de données très grand,
- Complexité des traitements effectués,
- Fortes contraintes de performances exigées.

I . 1. Présentation de la méthode Merise

But d'une méthode :

Modéliser la **réalité** de l'entreprise par une **représentation virtuelle** qui fait ressortir les points essentiels pour l'étude.

Composantes d'une méthode :

Une méthode est composée des éléments suivants :

- Des principes fondamentaux,
- Une démarche: Étapes de mise en œuvre,
- Des outils : Des Langages , Des modèles.

I . 1. Présentation de la méthode Merise

29

Classes de méthodes : (Recherche a rendre)

Méthodes fonctionnelles : Les méthodes fonctionnelles ou cartésiennes consistent à décomposer hiérarchiquement une application en un ensemble de sous applications.

« Orientée traitements ». Exemple : SADT, SSADM, Yourdon,

Méthodes systémiques : La méthode systémique est axée sur un ensemble des organes cohérents s'influençant les uns les autres, dépendant les uns des autres, et agissant les uns sur les autres.

« Orientée données ». Exemple : **Merise**, AXIAL, MEGA.

Méthodes orientées objets : La méthode d'analyse et de conception d'applications orientées objet est fondée sur une démarche participative par prototypage incrémental permettant aux utilisateurs d'intervenir très tôt dans le processus de développement du logiciel.

« Orientée données et traitements ». Exemple : UML, OMT.

I . 1. Présentation de la méthode Merise

30

Historique :

En 1977/1978, demande du Ministère de l'industrie :

Choix de société de conseil en informatique pour la constitution d'une méthode de conception des systèmes d'information :

- Équipe de J-L. Lemoigne (Univ. D'Aix / Marseille)
- CTI (Centre d'études Techniques de l'équipement)
- CETE (Centre d'études Techniques de l'équipement)

Méthode MERISE (1979) :

- ✓ Conception du S.I par étapes validées;
- ✓ Séparation des données et des traitements
- ✓ Vérifier la concordance entre données et traitements
- ✓ Vérifier que toutes les données nécessaires aux traitements sont présentes
- ✓ Vérifier qu'il n'y a pas de données superflues

Méthode MERISE 2ème génération en 1992

I . 1. Présentation de la méthode Merise

La méthode Merise : (Méthode d'Étude et de Réalisation Informatique par les Sous-Ensembles ou pour les Systèmes d'Entreprise – 1979)

C'est une méthode systémique de conception des systèmes d'information. Elle est en relation avec le développement des bases de données relationnelles.

Principes :

- **Vision globale** sur le système,
- Méthode pour Rassembler les Idées Sans Effort
- Formalisation par **niveaux d'abstraction**,
- Séparation entre **modèles de données** (formalisme entité-association) et **modèles de traitements**.
- Trois phases principales: **analyse** (diagnostic), **conceptualisation** (modélisation) et **développement** (Bases de données et programmes)

I . 1. Présentation de la méthode Merise

Approche Données/Traitements :

La méthode MERISE est basé sur la séparation des **données** et des **traitements** à effectuer en plusieurs modèles **Conceptuels**, **Organisationnel** (logiques), et **Opérationnel** (physiques).

La séparation des données et des traitements assure :

- Réutilisation;
- Facilité de maintenance.

I . 1. Présentation de la méthode Merise

33

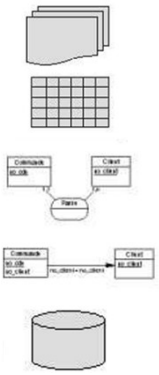
la méthode Merise préconise 3 niveaux d'abstraction :

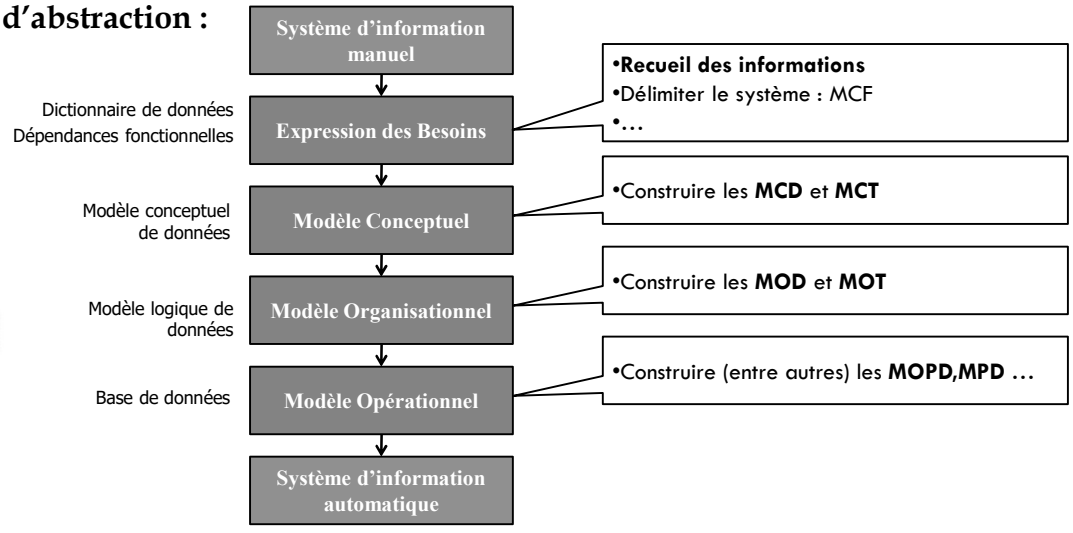
- **Le Niveau Conceptuel** qui décrit la statique et la dynamique du système d'information en se préoccupant uniquement du point de vue du gestionnaire.
- **Le Niveau Organisationnel (logique)** décrit la nature des ressources qui sont utilisées pour supporter la description statique et dynamique du système d'information. Ces ressources peuvent être humaines et/ou matérielles et logicielles.
- **Le Niveau Opérationnel (physique)** dans lequel on choisit les techniques d'implantation du système d'information (données et traitements),

I . 1. Présentation de la méthode Merise

34

Cycle d'abstraction :





I . 1. Présentation de la méthode Merise

Recueil et organisation des informations :

- Faire l'inventaire des éléments d'informations circulant dans le système : **existants et demandés.**
- Plusieurs approches sont possibles :

- La plus basique (orientée données) repose sur :
  - la création d'un dictionnaire de données et,
  - la matrice des dépendances fonctionnelles.

- La plus complète (orientée traitement) est basée sur :
  - la création du Modèle Conceptuel d'Activité (MCA) et,
  - du Modèle Conceptuel de Communication (MCC).

I . 1. Présentation de la méthode Merise

Modèle Conceptuel de Données (MCD) :

- Toute donnée recensée doit être mémorisée.

- Le **MCD** modélise cette mémoire (collective) du système.

- Un formalise de référence :
  - Le modèle Entité-Association.
  - Concepts d'entités et d'associations.
  - Particulièrement adapté aux Base de Données relationnelles.

- **Redondance interdite !**

I . 1. Présentation de la méthode Merise

Modèle Conceptuel de Traitement (MCT) :

- Son objectif est la description de la **transformation des informations**.

- Se base sur plusieurs notions :
  - **Activité :** décrit perception globale du fonctionnement du système, et est, par le fait, complexe.
  - **Traitement :** décrit l'un des composants de l'activité du système.
  - **Action :** décrit une fonctionnalité atomique dans un traitement (consultation, mise à jour…).

I . 1. Présentation de la méthode Merise

Modèles Organisationnelles de Données et de Traitements (MOD, MOT) :

- Concepts identiques à ceux du MCD et MCT sauf que …

- L'intégration de notions supplémentaires, comme
  - Les lieux (où ?),
  - Les personnes (qui ?),
  - Les ressources (comment ?),
  - …
  - En bref, les contraintes spatiales et temporelles,

- Imposent que,
  - La redondance de données soit tolérée et que,
  - Les traitements soient raffinés. .

I . 1. Présentation de la méthode Merise

Modèles Opérationnels : Logique et Physique

- **Le modèle logique** représente un choix logiciel pour le système d'information.
- **Le modèle physique** reflète un choix matériel pour le système d'information.

I . 1. Présentation de la méthode Merise

Les modèles :

6 modèles de base :

	Données	Traitements
Niveau conceptuel	Modèle conceptuel de données (MCD)	Modèle conceptuel de traitements (MCT)
Niveau logique / organisationnel	Modèle logique de données (MLD)	Modèle organisationnel de traitements (MOT)
Niveau physique / opérationnel	Modèle physique des données (MPD	Modèle opérationnel des traitements (MOP)

I . 1. Présentation de la méthode Merise

La démarche Merise :

Quatre étapes :

- Etude préalable
- Etude détaillée
- Réalisation
- Mise en œuvre

I . 1. Présentation de la méthode Merise

Etude préalable :

- Recueil des données (existant, entretiens, …)
  - Cerner le projet,
  - Comprendre les besoins
  - Identifier des concepts (règles de gestion, règles d'organisation …)
  - Proposer une première solution
- Diagramme de flux
- Dossier d'étude préalable

Étude détaillée

- Décrire complètement, au plan fonctionnel, la solution à réaliser
- Débouche sur un dossier de spécifications détaillées

I . 1. Présentation de la méthode Merise

Réalisation

- Production du code informatique
- Débouche sur un dossier de réalisation

Mise en œuvre

- Formation
- Documentation
- Installation
- Initialisation des données

I . 2. Introduction aux bases de données

Objectifs :

- Comprendre le rôle des bases de données au sein d'une organisation.
- Identifier la logique et l'utilité des SGBD.
- Identifier l'objectif de la conception d'une base de données.
- Comprendre le principe des dépendances fonctionnelles.
- Réaliser le modèle conceptuel d'une base de données à l'aide des concepts Merise.
- Passer du modèle conceptuel au modèle logique.
- Comprendre la normalisation.
- Créer et manipuler une BD.

I . 2. Introduction aux bases de données

Définition :

Une base de données représente un ensemble de données **structurées** ou **non structurées** et **mémorisées** sur un **support permanent**, qui est **utilisé par de nombreuses personnes** et dont l'organisation est régie par un modèle de données (décrit la manière dont sont représentées les données dans une organisation).

Par exemple, dans une entreprise constituée de plusieurs services (service commercial, service d'approvisionnement, etc.), la BD sera partagée et utilisée par différents services qui n'ont pas les mêmes besoins.

I . 2. Introduction aux bases de données

Exemple d'une base de données :

Données de vente d'un magasin:

- Date
- No. d'article
- Nom d'article
- Montant
- Vendeur

Date	No. article	Nom d'article	Montant	Vendeur
2018/03/05	B1234	Casserole	53.50	Ahmed
2018/03/05	A928	Nappe tissue	16.30	Ilham
...				
2018/03/06	B7645	Poêle à frire	32.85	Mouna
...				

I . 2. Introduction aux bases de données

Traitements possible sur la BD :

Statistique:

- le volume de vente d'un jour/mois, …
- le volume de vente d'un vendeur
- les articles populaires (les plus vendus)
- ...

Extraction d'une partie de données:

- Les ventes par Ilham
- Les ventes < 50 Dhs

Date	No. article	Nom d'article	Montant	Vendeur
2018/03/05	B1234	Casserole	53.50	Ahmed
2018/03/05	A928	Nappe tissue	16.30	Ilham
…				
2018/03/06	B7645	Poêle à frire	32.85	Mouna
…				

I . 2. Introduction aux bases de données

Fonctionnalités principales d'une bases de données :

Les principes fondamentaux:

- ❑ Fidélité: représentation du monde réel ;
- ❑ Unicité: gestion des informations cohérentes et non-redondantes;
- ❑ Indépendance: des programmes par rapport aux données;
- ❑ Sécurité: confidentialité des données;
- ❑ Concurrence: Partage des données.

I . 2. Introduction aux bases de données

Représentation du monde réel :

Une image aussi fidèle que possible de la réalité à tout instant;
Une représentation fidèle implique une information fiable et à jour.

Contraintes d'intégrité :

- définit la cohérence d'une donnée ou d'un ensemble de données;
- Exprimées simplement;
- Vérifiées automatiquement à chaque insertion, modification ou suppression des données.

I . 2. Introduction aux bases de données

Gestion des informations cohérentes et non-redondantes :

- Pas de duplication de l'information;

Indépendance des programmes par rapport aux données

- Modifications apportées à la structure de la base par un changement du monde réel Et non Pour une application particulière,

Sécurité et confidentialité des données;

- Données partagées:
  - Les informations confidentielles ne sont accessibles qu'aux personnes habilitées;
    Assigner à chaque utilisateur des droits d'accès aux données;
- Sécurité et protection des supports physiques des informations contre toute altération ou destruction (résistance aux pannes)
  - Récupérer les données dans l'état dans lequel elles étaient avant la modification;

I . 2. Introduction aux bases de données

Partage des données

- Bien que partageant des ressources communes, les applications doivent être performantes.
- Permettre aux utilisateurs d'accéder aux mêmes données au même moment (la concurrence d'accès).

I . 2. Introduction aux bases de données

Exemple : **Gestion d'une entreprise de transport public**

L'entreprise « ALSA Maroc » s 'occupe des transports publics de la ville de casablanca, désire se doter d'un système informatique pour la gestion de son réseau. Celui-ci comprend des lignes, des véhicules ainsi que des chauffeurs.

Le chauffeur « ALI » est en congé le lundi 30 octobre. Le 31 octobre, il assure la ligne 9 avec le véhicule 56….

Questions :

1) Un véhicule doit-il toujours assurer la même ligne?
2) Qui a assuré la ligne B le 3 octobre entre 16h et 18h?
3) Comment enregistrer et utiliser les informations concernant l'entreprise?

I . 2. Introduction aux bases de données

Les données :

Exemple :

le chauffeur ALI assure la ligne 9 avec le véhicule 56.

le chauffeur ALI assure la ligne B avec le véhicule 4.

le chauffeur AHMED assure la ligne 9 avec le véhicule 86.

Les données :

Chauffeur = {ALI, AHMED}

Véhicule = {4,56,86}

Ligne = {B,9}

I . 2. Introduction aux bases de données

Les données structurées :

Description ou schéma de la BD

- Chauffeur: chaine de caractères (50)
- Ligne : chaine de caractères (2)
- Véhicule: nombre (10)

Chauffeur	Ligne	Véhicule	Horaire
ALI	9	56	31 Octobre
ALI	B	4	3 Octobre
AHMED	9	86	30 Octobre

Questions :

1) Un véhicule doit-il toujours assurer la même ligne?
2) Qui a assuré la ligne B le 3 octobre entre 16h et 18h?
3) Comment enregistrer et utiliser les informations concernant l'entreprise?

Chapitre 2 Les Systèmes de Gestion des Bases des Données (SGBD)

II. Les Systèmes de Gestion des Bases des Données (SGBD)

Qu'est-ce qu'un SGBD ?

- Un ensemble de logiciels permettant aux utilisateurs de définir, créer, maintenir, contrôler et accéder à la BD;
- Le SGBD rend transparent le partage des données.

Historique :

- Les SGBD ont vu le jour dans les années 1960 ;
- Gérer d'importants volume de données ;
- Systèmes propriétaires appartenant à des constructeurs d'ordinateurs (Ex : IBM) ;
- Fonctionnent sur des grands systèmes (mainframes) ;
- Schéma d'organisation « hiérarchique » ou « réseau ».

IBM S/360, Le premier mainframe

II. Les Systèmes de Gestion des Bases des Données (SGBD)

Les modèles :

Historiquement, les modèles des bases de données ont été définis comme suit, dans un ordre chronologique :

- Modèle hiérarchique (structure de données «arbre»)
- Modèle réseau (structure de données «graphe»)
- Modèle relationnel (structure de données «tableau de n-uplets»)
- Modèle objet (structure de données «classes, attributs, méthodes»)

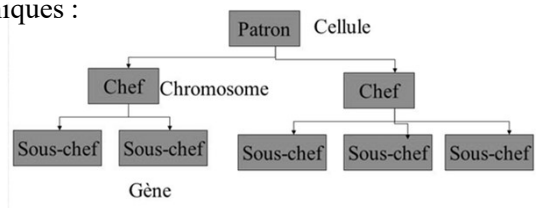
II. Les Systèmes de Gestion des Bases des Données (SGBD)

Le modèle hiérarchique :

- BD construite selon un modèle en arborescence, avec une racine et plusieurs niveaux de sous-arbres;
- Chaque élément comporte juste un lien vers le niveau inférieur ;
- Les accès aux données commencent par la racine et descendent l'arborescence jusqu'aux détails recherchés.
- Les structures de données hiérarchiques ont été utilisées dans les premiers systèmes de gestion de base de données de type mainframe.

Quelques bases connues de bases de données hiérarchiques :

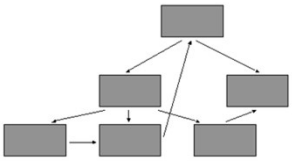
- ADABAS,(AG Software, fin des années 1970)
- IMS (IBM, 1966 pour le programme APollo)
- System 2000 (1967)



II. Les Systèmes de Gestion des Bases des Données (SGBD)

Le modèle réseau :

- Le modèle réseau a été proposé par le groupe DBTG du comité CODASYL
- Ce modèle est une extension du modèle précédent (hiérarchique)
- Nombreux liens entre les différents éléments de données ;
- Accès aux données réalisés par des cheminements divers.
- Le langage de manipulation de données du CODASYL est fortement lié à COBOL, bien que généralisé et utilisable depuis d'autres langages de 3e génération tel Fortran.



II. Les Systèmes de Gestion des Bases des Données (SGBD)

Le modèle relationnel

- Une base de données relationnelle est une base de données structurée suivant les principes de l'algèbre relationnelle;
- Inventé par le chercheur Edgar Frank Codd chez IBM à la fin des années 1960, il a utilisé la théorie des ensembles et la logique des prédicats du premier ordre afin de résoudre des difficultés telles que la redondance des données, l'intégrité des données ou l'indépendance de la structure de la base de données avec sa mise en œuvre physique;
- Modèle conceptuel reposant sur une représentation unifiée de l'information sous forme de tables ;
- Permet une grande indépendance entre les applications, les données et le support physique ;
- Supporte le langage SQL;
- Propose une démarche cohérente et unifiée pour :
  - La description des données (LDD : Langage de Description des Données) ;
  - L'interrogation des données (LMD : Langage de Manipulation des Données).

II. Les Systèmes de Gestion des Bases des Données (SGBD)

61

Quelques SGBD relationnel :

Les bases de données Micro :

- dBase (Borland) : gestionnaire de fichiers structurés + langage de programmation
- FoxPro
- Access (Microsoft)
- Paradox (Borland)

Les bases de données Macro :

- Oracle
- Ingres
- Informix
- Sybase
- DB2 (IBM)

II. Les Systèmes de Gestion des Bases des Données (SGBD)

62

Le modèle objet :

- Détermine au départ un langage de programmation (1982, Smalltalk)
- L'idée est que la description (l'objet) détermine les actions (méthodes)
- L'organisation en « base de données » est relativement récente
- Les objets sont classés les uns par rapport aux autres (relations de classe)
- Une démarche objet évoluée basé sur le langage UML

Quelques SGBD (Système de Gestion de Base de Données) objet :

- PostgreSQL, SGBD relationnel orienté objet logiciel libre
- Wakanda, base NoSQL avec interface objet JavaScript
- O2 édité par O2 Technology
- Matisse
- db4o Système de gestion de bases de données orientées objet (open source)

II. Les Systèmes de Gestion des Bases des Données (SGBD)

Objectifs d'un SGBD :

- Langage de manipulation des données;
- Indépendance données/SGBD;
- Fournir un accès efficace aux données;
- Contrôler la redondance des données;
- Cohérence des données;
- Partage des données;
- Sécurité des données;

II. Les Systèmes de Gestion des Bases des Données (SGBD)

Fonctionnalités :

- L'utilisation d'un SGBD suppose de comprendre (et donc de savoir utiliser) les fonctionnalités suivantes:

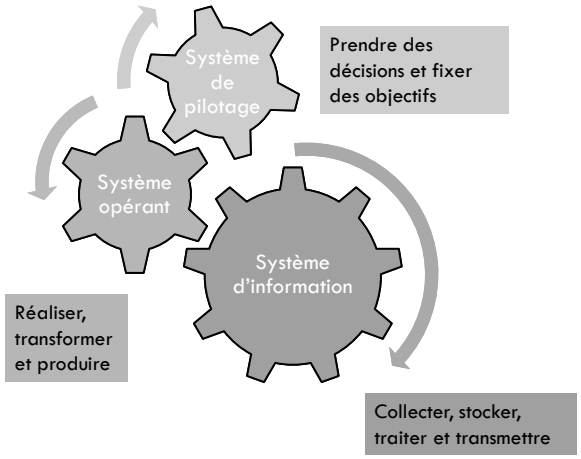
1. Définition du schéma de données en utilisant les modèles de données du SGBD (LDD).
2. Opérations sur les données: recherche, mises-à-jour, etc..(LMD).
3. Partage les données entre plusieurs utilisateurs selon les autorisations (LCD).
4. Optimisation les performances, par le réglage de l'organisation physique des données.

Chapitre 3 Les principaux modèles de données

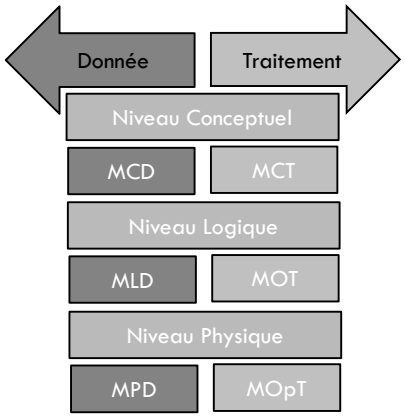
III. 1. Modèle conceptuel de données: MCD

Rappel Chapitre 1 et 2

Organisation



MERISE



III. 1. Modèle conceptuel de données: MCD

Les différents modèles proposés par Merise à chaque niveau :

Niveau	Préoccupation	Données	Traitements
Conceptuel	Quoi ?	MCD	MCT
Organisationnel ou Logique	Qui fait quoi ? Ou ? Quand ?	MLD	MOT
Physique ou Opérationnel	Avec quels moyens ?	MPD	MopT

LA MÉTHODE MERISE

Une description du système d'information en trois niveaux :

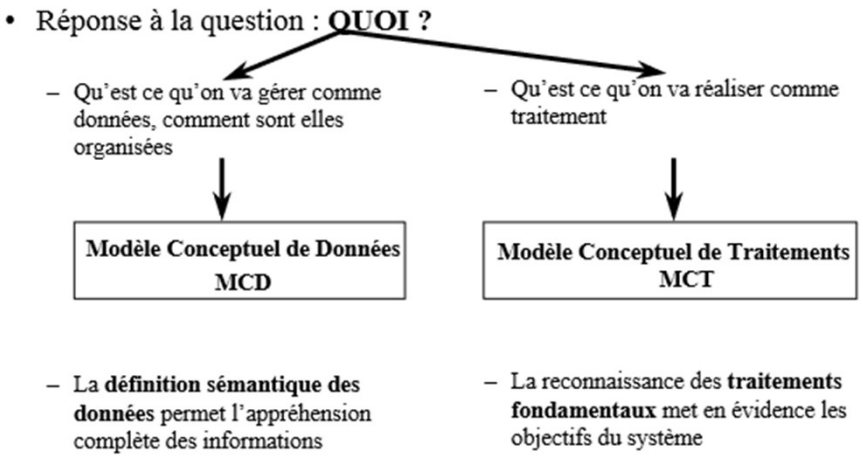
- Le niveau conceptuel : Le quoi ?
- Le niveau logique :Qui fait quoi?, ou et quand ?
- Le niveau physique : Comment ?(avec quels moyens ?)

LA MÉTHODE MERISE

- Niveau conceptuel (Le QUOI ?) :
  - Dans ce niveau, nous nous concentrons sur le "quoi".
  - Quelles sont les principales entités du système?
  - Quelles sont leurs relations?
  - À ce niveau, nous cherchons à définir les besoins fondamentaux du système sans entrer dans les détails techniques.
  - Pensez-y comme à la création d'une carte ou d'un plan général du système.

LA MÉTHODE MERISE

- Niveau conceptuel (Le QUOI ?) :

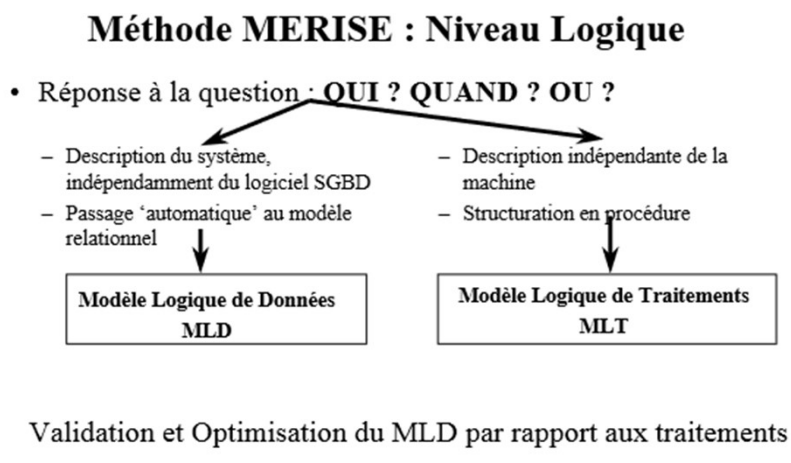


LA MÉTHODE MERISE

- Niveau logique (QUI FAIT QUOI?, OÙ et QUAND ?) :
  - Une fois que nous avons une compréhension claire du "quoi", nous commençons à décomposer les responsabilités.
  - Qui est responsable de quoi?
  - Où ces actions sont-elles exécutées? Et quand?
  - Ce niveau est crucial pour définir les rôles et les responsabilités dans le système, garantissant que chaque élément a une fonction claire.

LA MÉTHODE MERISE

- Niveau logique (QUI FAIT QUOI?, OÙ et QUAND ?) :

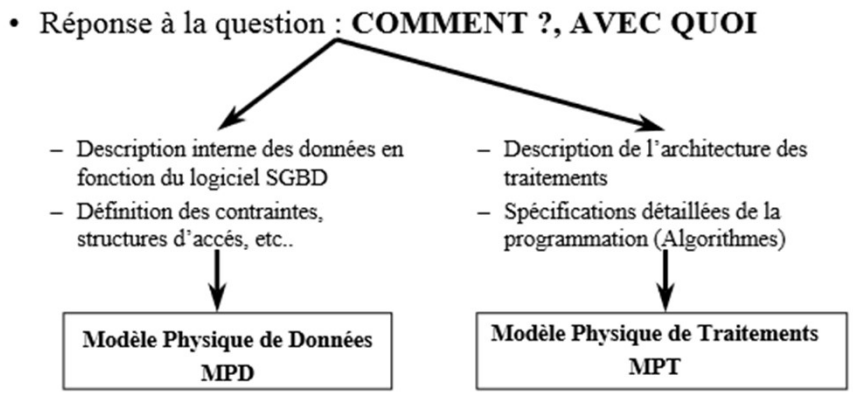


LA MÉTHODE MERISE

- Niveau physique (COMMENT? Avec quels moyens?) :
  - C'est le niveau le plus détaillé.
  - Après avoir défini le "quoi" et le "qui", nous abordons le "comment".
  - Comment les processus seront-ils mis en œuvre techniquement?
  - Quels outils, technologies ou équipements seront nécessaires?
  - Cette étape se penche sur la mise en œuvre concrète, garantissant que le système est non seulement bien conçu mais également réalisable avec les ressources disponibles.

LA MÉTHODE MERISE

- Niveau physique (COMMENT? Avec quels moyens?) :

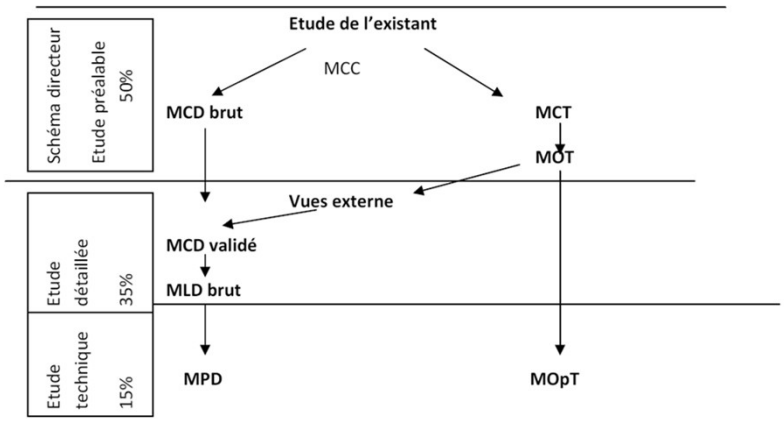


CYCLE D'ABSTRACTION POUR LA CONCEPTION DES S.I

Niveau	Statique (donnée)	Dynamique (traitement)	
Conceptuel	**MCD** Décrivant les données et les liens entre ces données	**MCT** Décrivant les traitement, règles et les contraintes	**Quoi**
Logique ou Organisationnel	**MLD** Décrivant la structure des données	**MOT** Décrivant les contraintes dues à l'environnement (Organisationnel, spatial et temporel)	**Ou** **Qui** **Quand**
Opérationnel ou **Physique**	**MPD** Décrit la façon d'implémenter le modèle des données dans le SGBD	**MOPT** Définit la structure interne des applications: • Décomposition des application en modules • Description des traitements(algorithme, fonctions….)	**Comment**

III. 1. Modèle conceptuel de données: MCD

Les différents modèles proposés par Merise à chaque niveau :



III. 1. Modèle conceptuel de données: MCD

77

Base de données et SGBD :

- Une base de données contient l'ensemble des données informatisées d'un système d'information. Cette base est implantée physiquement sur le disque d'un ordinateur sous la forme d'un ou plusieurs fichiers. Le logiciel spécialisé dans la gestion d'une base de données s'appelle un SGBDR (Système de Gestion de Bases de Données Relationnelles).

- Un SGBD représente donc l'ensemble des programmes assurant structuration, stockage, maintenance, mise à jour et recherche des données d'une base + interfaces nécessaires aux différentes formes d'utilisation de la base.

III. 1. Modèle conceptuel de données: MCD

78

Introduction :

- Un système d'information est définie par deux composantes : les données qui constituent l'aspect statique et les traitements qui constituent l'aspect dynamique.
- Merise possède l'avantage, qui est d'ailleurs l'un des points clés de sa réussite, de décrire les données indépendamment des traitement.
- L'objectif poursuivi est la définition et l'élaboration de la structure globale des données de manière indépendamment de toute contrainte organisationnelle ou technologique. La structure est appelé **modèle conceptuel des données (MCD).**
- Au niveau conceptuel de la méthode, on élabore pour les données, le modèle conceptuel des données (MCD) et pour les traitements, le modèle conceptuel des traitements.

On s'intéressera dans ce cours aux modèles de représentation des données.

III. 1. Modèle conceptuel de données: MCD

Introduction :

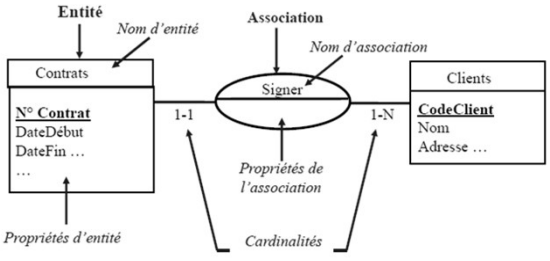
- Le modèle conceptuel des données (MCD) décrit la signification des données sur lesquelles reposent les systèmes d'information et les structures.
- Le MCD est l'élément le plus connu de MERISE et certainement le plus utile. Il permet d'établir une représentation claire des données du S.I. et définit les dépendances fonctionnelles de ces données entre elles.
- Le MCD permet une représentation conceptuelle de l'ensemble des données manipulées et des règles de gestion auxquelles elles sont soumises.

III. 1. Modèle conceptuel de données: MCD

Concepts fondamentaux :

Le MCD s'articule autour des concepts suivants :

- ✓ **Entité *;**
- ✓ **Propriété *;**
- ✓ **Identifiant;**
- ✓ **Occurrence;**
- ✓ **Association *;**
- ✓ **Cardinalités.**



→ On dit le modèle entité-association ou bien le modèle MCD

* Les éléments de base constituant un modèle conceptuel des données

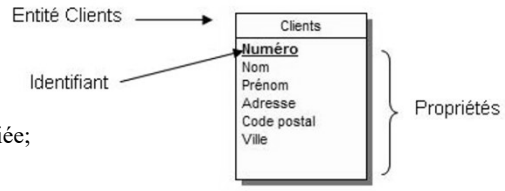
III. 1. Modèle conceptuel de données: MCD

1. Entité :

- Une entité est un ensemble de propriétés correspondant à un type d'objet (éléments) ayant un intérêt dans le SI et une existence propre.
  - **Exemples :** article, employé, client, fournisseur, commande …
- Une entité est la représentation d'un objet matériel (concret) ou immatériel (abstrait) du monde réel.
- Une entité est identifiable et ne doit représenter qu'un seul et même concept sémantique.
- Parmi les propriétés d'une entité, il existe un sous-ensemble qui joue le rôle d'identifiant
  - **Exemple :** référence article, matricule employé, CNE étudiant …
- Un identifiant permet de connaître sans ambiguïté toutes occurrences de l'entité.
  - Le plus souvent, l'identifiant est un numéro, code, référence …

III. 1. Modèle conceptuel de données: MCD

1. Entité :



- Elle est définie par :
  - Une existence propre et une utilité pour l'organisation étudiée;
  - Des occurrences multiples ;
  - Des propriétés (au moins une) dont un identifiant.
- Une entité est représentée dans le MCD par un rectangle muni d'un cartouche qui indique son nom et elle contient la liste de toutes ses propriétés.

L'identifiant est placé en tête des propriétés et souligné.

III. 1. Modèle conceptuel de données: MCD

2. Propriété :

- Les propriétés décrivent l'entité ou l'association. Elles apportent l'information utile et nécessaire au système d'information.
- Donnée élémentaire représentant la plus petite partie (atomique) manipulée dans le SI et ayant un sens.
  - Atomique → non décomposable.

III. 1. Modèle conceptuel de données: MCD

2. Propriété :

Règles :

- ✓ Une propriété ne doit pas être composée
  - Attention : date, adresse ?
- ✓ Une propriété ne doit pas être calculée (prix TTC, durée, âge…).
- ✓ Une propriété ne doit jamais être redondante dans le MCD :
  - Pas de synonymes (ex : référence article et N° produit).
  - Pas de polysémies : même signifiant pour plusieurs signifiés (ex : "adresse" qui désigne "adresse client" et "adresse fournisseur").
    - On crée deux propriétés avec deux nom différents.

III. 1. Modèle conceptuel de données: MCD

3. Identifiant :

- C'est un groupe minimal d'attributs tels qu'il n'existe pas deux occurrences ayant les mêmes valeurs pour ces propriétés. L'identifiant d'une entité permet de distinguer chaque occurrence de l'entité par rapport à toutes les autres.

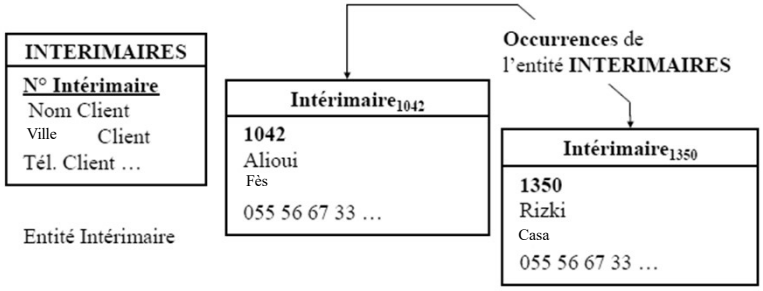
- Exemple : référence article, matricule employé, CNE étudiant …

- Un identifiant permet de connaître sans ambiguïté toutes occurrences de l'entité.
- Le plus souvent, l'identifiant est un numéro, code, référence …

III. 1. Modèle conceptuel de données: MCD

4. Occurrences :

- Une occurrence d'une entité est connue par les valeurs spécifiques prises par chacune des propriétés de l'entité. Ces propriétés sont communes à toutes les occurrences de l'entité.
- Exemple:



III. 1. Modèle conceptuel de données: MCD

5. Dépendances fonctionnelles :

- Le rôle de l'établissement des dépendances fonctionnelles est de nous aider à comprendre les liens existants entre chaque donnée. Cette démarche de recherche des dépendances fonctionnelles est la pierre angulaire de toute l'analyse des données. En effet, cette activité étant la première dans l'élaboration de l'analyse, si elle est négligée c'est tout l'ensemble qui en subira les conséquences.
- Une propriété B dépend fonctionnellement d'une propriété A si à toute valeur de A correspond une, et une seule, valeur B.
  - On dit aussi que A détermine B.
  - On note A→ B.

III. 1. Modèle conceptuel de données: MCD

5. Dépendances fonctionnelles :

- Exemples :
  - Moyenne → Mention
  - CNE → Nom étudiant
  - Code Postal → Ville
  - Matricule employé → Nomemployé
  - Matricule employé → Date d'embauche
  - ~~Date d'embauche → Salaire employé~~

III. 1. Modèle conceptuel de données: MCD

5. Dépendances fonctionnelles :

5.1. Dépendances fonctionnelles composées :

- Une dépendance fonctionnelle peut porter sur la concaténation de plusieurs propriétés :

A1, A2, …, An → B.

- Une dépendance fonctionnelle qui comporte plusieurs attributs est dite composée.

- Exemples :
  - N° Bon de Commande, RefProduit → Quantité commandée.
  - Code Client, Nom Client → AdrClient

III. 1. Modèle conceptuel de données: MCD

5. Dépendances fonctionnelles :

5. 2. Dépendance fonctionnelle élémentaire :

- Une dépendance fonctionnelle A → B est élémentaire s'il n'existe pas une donnée C, sous-ensemble de A, décrivant une dépendance fonctionnelle de type C → B

- Exemple :
  - RéférenceProduit → Désignation
  - NuméroCommande, RéférenceProduit → Quantité
  - NuméroCommande, RéférenceProduit → Désignation
  - Code Client, Nom Client → AdrClient
  - Code Etudiant, N°Livre, Date emprunt → Date retour

III. 1. Modèle conceptuel de données: MCD

5. Dépendances fonctionnelles :

❑ 5.3. Dépendance fonctionnelle élémentaire directe :

❑ On dit que la dépendance fonctionnelle A → B est directe s'il n'existe aucun attribut C tel que l'on puisse avoir A → C et C → B. En d'autres termes, cela signifie que la dépendance fonctionnelle entre A et B ne peut pas être obtenue par transitivité.

❑ Exemples :

- RefArticle → Catégorie
- Catégorie → Taux TVA
- RefArticle → Taux TVA
- RefArticle → Catégorie → Taux TVA
  - Donc RefArticle → Taux TVA n'est pas directe, elle est transitive

III. 1. Modèle conceptuel de données: MCD

5. Dépendances fonctionnelles :

❑ 5.3. Dépendance fonctionnelle élémentaire directe :

❑ Exemple 1:

Soient les dépendances fonctionnelles :

NumFacture → NumReprésentant

NumReprésentant → NomReprésentant

NumFacture → NumReprésentant ??

❑ Exemple 2:

Soient les dépendances fonctionnelles :

NumFacture → NumReprésentant

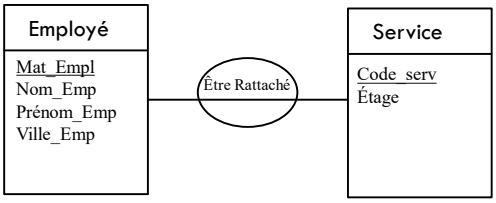
NumReprésentant → NomReprésentant

NumFacture → NomReprésentant ??

III. 1. Modèle conceptuel de données: MCD

5. Dépendances fonctionnelles :

- 5.4. Dépendance fonctionnelle entre entités :
- Une entité E2 dépend fonctionnellement d'une autre entité E1 si toute occurrence de E1 détermine une et une seule occurrence de E2.
- On note E1 → E2



- Exemples :
  - Employé → Service
  - Commande → Client
  - Compte Mail → Utilisateur

III. 1. Modèle conceptuel de données: MCD

6. Association (Relation) :

- C'est le lien qui relie deux entités (ou plus).
- L'association est schématisée par un ovale barré. Dans la partie supérieure, on inscrit le nom de l'association, souvent un verbe (à l'infinitif), qui caractérise le type de relation entre les entités. et dans l'autre, la liste des propriétés (association porteuse).

- Exemples :
  - Un service comprend des employés (un employé est affecté à un service)
  - Une commande concerne des articles.
  - Un employé est chef d'un employé.

III. 1. Modèle conceptuel de données: MCD

6. Association (Relation) :

- Exemple :

Règles de gestion:

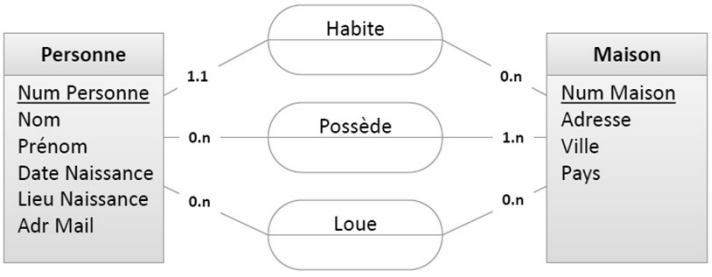
- Un professeur enseigne au moins une matière. Il peut enseigner plusieurs.
- Une matière peut ne pas être enseigner. Elle peut être enseignée par plusieurs professeur

Une association peut être porteuse ou non de propriétés, (Relation vide ou non).

III. 1. Modèle conceptuel de données: MCD

6. Association (Relation) :

- Exemple :



- Quelles sont les règles de gestion correspondantes à ce modèle?

III. 1. Modèle conceptuel de données: MCD

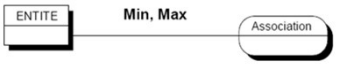
6. Association (Relation) :

- Règles de gestion:
  - Une personne habite une seule maison;
  - Une personne peut posséder plusieurs maisons;
  - Une personne peut louer plusieurs maisons;
  - Une maison peut être vide;
  - Une maison est possédé au moins par une personne;
  - Une maison peut ne pas être louée.

III. 1. Modèle conceptuel de données: MCD

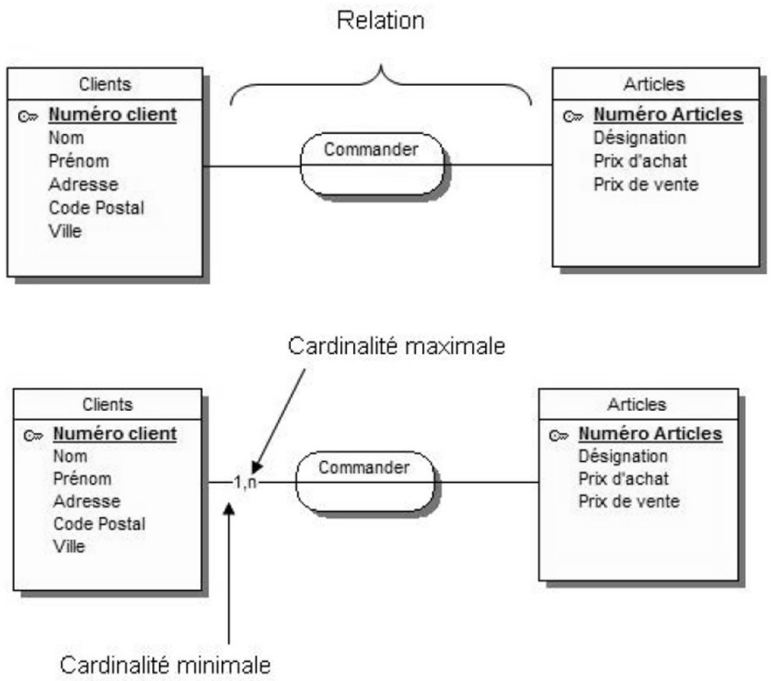
7. Cardinalités :

- Les cardinalités d'une entité dans une association exprime le nombre de fois qu'une occurrence de cette entité est impliquée dans l'association, au minimum et au maximum.
- Les cardinalités représente pour chaque couple (entité, association) les nombres minimum et maximum d'occurrences de l'association que peut avoir un objet.



- Chaque sens de lecture d'une association est entièrement décrit lorsqu'on précise le couple (cardinalité mini, cardinalité maxi).

- Exemple :
  - ✓ Un service comprend un ou plusieurs employé.
  - ✓ Un employé est rattaché à un ou plusieurs services (en cours du temps)



99

III. 1. Modèle conceptuel de données: MCD

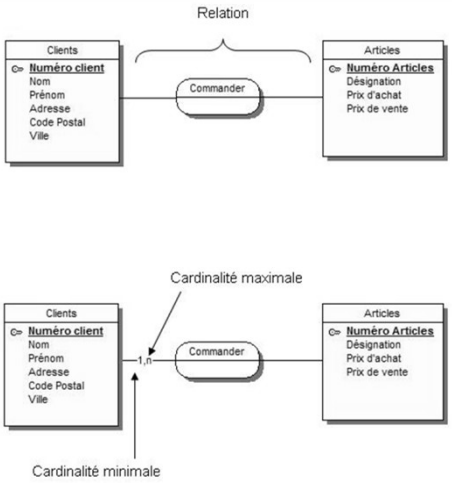
100

7. Cardinalités :

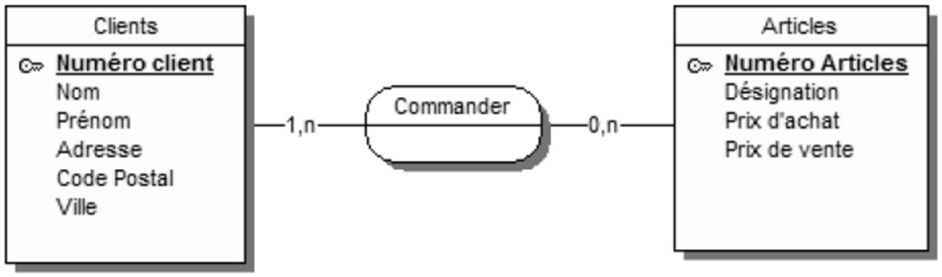
❑ Exemple : Un client peut commander des articles.

Dans notre exemple on peut se poser les questions suivantes :

- ✓ Combien de fois au minimum un client peut il commander un article ?
- ✓ Combien de fois au maximum un client peut il commander un article ?



- ▪ À la première question, nous pouvons répondre qu'un client, pour être client, doit commander au moins un article.
- ▪ À la deuxième question, nous pouvons répondre qu'un client peut commander plusieurs articles.



101

III. 1. Modèle conceptuel de données: MCD

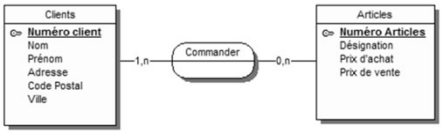
102

7. Cardinalités :

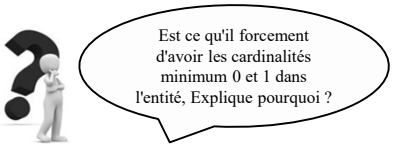
❑ Exemple : Un client peut commander des articles.

Le n représente la notion de « plusieurs » ; ici nous avons représenté le fait qu'un client peut commander un ou plusieurs articles. Il faut que nous nous posions les mêmes questions pour l'article :

- ✓ Combien de fois au minimum un article peut il être commandé par un client ?
- ✓ Combien de fois au maximum un article peut il être commandé par un client ?



- Pour le minimum, nous pouvons l'interpréter de la façon suivante :
  - A-t-on des articles qui ne peuvent jamais être commandés ?
  - Si nous répondons oui dans ce cas la cardinalité minimale est 0.
- Pour le maximum :
  - A-t-on des articles qui peuvent être commandés plusieurs fois ?
  - Nous pouvons espérer que oui, dans ce cas la cardinalité maximale sera n.



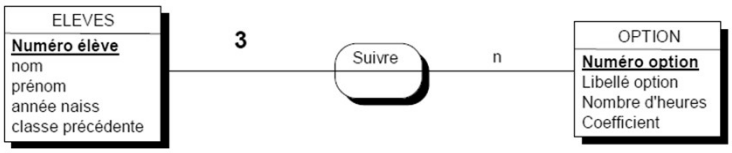
III. 1. Modèle conceptuel de données: MCD

7. Cardinalités :

Il arrive (mais c'est rare) qu'une cardinalité maximale ait une valeur limitée.

Exemple :

- Règles de gestion : Un étudiant peut suivre au maximum 3 options.



III. 1. Modèle conceptuel de données: MCD

7. Cardinalités :

- Il y a trois valeurs typiques : 0, 1 et N (plusieurs)
- Qualificateurs possibles :
  - (1,1)
  - (0,n)
  - (1,n)
  - (0,1)

Règles:

- La valeur n ne peut jamais être à gauche
- La valeur 0 ne peut jamais être à droite

- Les cardinalités traduisent des règles de gestion.
- Ce sont des règles propres à l'organisation étudiée, qui sont décidées par les gestionnaires et décideurs. Ces règles expriment des contraintes sur le modèle.

III. 1. Modèle conceptuel de données: MCD

Dimension d'une association :

- L'ensemble d'entités intervenant dans une association constitue une collection.
- La dimension de l'association est le nombre d'entités entrant dans sa collection.

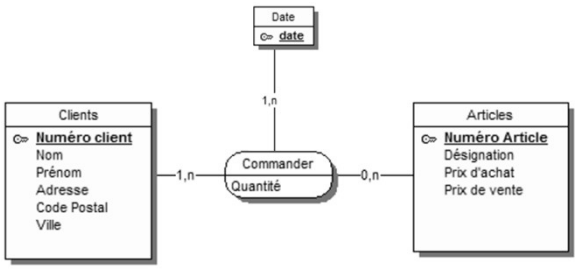
En général le nombre d'entités participant à la relation:

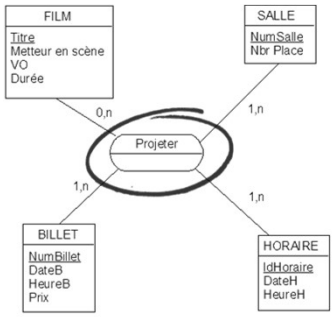
- Une relation entre deux objets est appelée : relation binaire.
- Une relation entre trois objets est appelée : relation ternaire.
- Une relation entre n objets est appelée : relation n-aire.

Dimension = nombre d'entité liées

III. 1. Modèle conceptuel de données: MCD

Dimension d'une association :

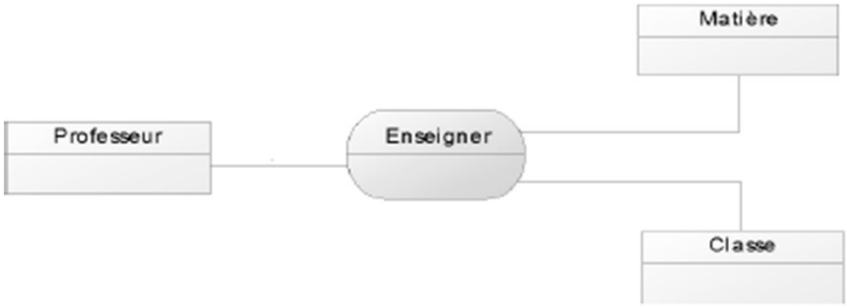




III. 1. Modèle conceptuel de données: MCD

Dimension d'une association :

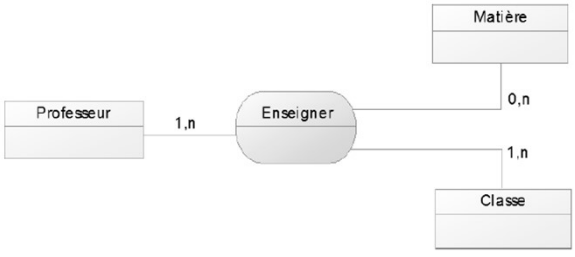
- Exemple : association entre professeur, cours et classe.
  - Déterminer les cardinalités.



III. 1. Modèle conceptuel de données: MCD

Dimension d'une association :

- Exemple : association entre professeur, cours et classe.
- Règles de gestion :
  - Un professeur fait au moins un enseignement, il peut en faire plusieurs.
  - Une matière peut ne pas être enseignée. Si elle l'est, elle peut l'être plusieurs fois.
  - Une classe a au moins un enseignement et peut en avoir plusieurs.

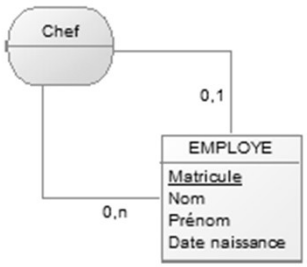


III. 1. Modèle conceptuel de données: MCD

Les relations réflexives (récursive) :

Une relation réflexive est une relation d'une entité sur elle-même.

- Exemple : hiérarchie dans l'entreprise.



- Nous interprétons donc qu'un employé peut dirigé par zéro ou un seul chef, et le chef peut dirigé personne ou plusieurs employés.

III. 1. Modèle conceptuel de données: MCD

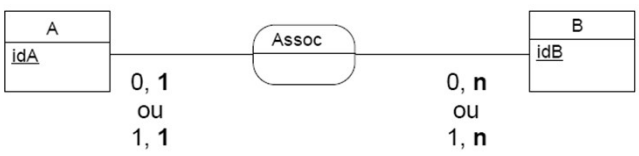
Les différents types d'associations :

On distingue deux catégories d'associations en fonction des cardinalités maximales de ses branches:

- Les associations hiérarchiques encore appelées associations [1, n] ou associations fonctionnelles
- Les associations non hiérarchiques, encore appelées associations [n, n] ou non fonctionnelles

- **Les associations hiérarchiques [1,n] :**

Ce sont les associations où d'un côté la cardinalité maximale est à 1 et de l'autre côté la cardinalité maximale est à n.



III. 1. Modèle conceptuel de données: MCD

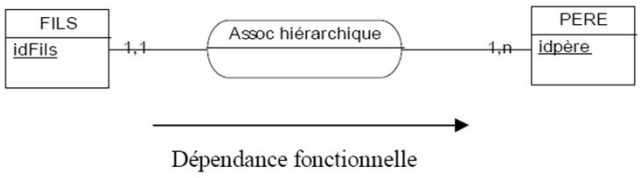
Les différents types d'associations :

- **Les associations hiérarchiques [1,n] :**
  - Cela signifie qu'une occurrence de A est reliée au plus à une seule occurrence de B.
  - C'est-à-dire si on connaît une occurrence de A alors on saura forcément quelle est la seule occurrence de B qui correspond (si elle existe).
  - On dit que A détermine B. C'est un lien de dépendance fonctionnelle. B dépend fonctionnellement de A.
  - Une association forte et hiérarchique. Sans entité parent, il ne peut pas y avoir d'entité enfant.
  - Ce type d'association est toujours vide.

III. 1. Modèle conceptuel de données: MCD

Les différents types d'associations :

- **Les associations hiérarchiques [1,n] :**
  - L'entité qui correspond à la branche du côté du 1 est parfois appelée entité fils et l'entité correspondant à la branche du côté n est parfois appelée entité père.
  - Cette appellation découle de l'analogie : un fils n'a qu'un seul père, et un père peut avoir plusieurs fils.



III. 1. Modèle conceptuel de données: MCD

Les différents types d'associations :

- **Les associations non hiérarchiques [n , n] :**

- Dès lors que nous avons la valeur max=n sur les branches de l'association, on dit que c'est association non hiérarchique.
- Il n'y a pas de dépendance entre les entités.

III. 1. Modèle conceptuel de données: MCD

Les différents types d'associations :

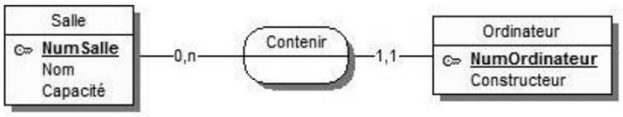
- **Notion de contrainte d'intégrité fonctionnelle (CIF) :**

- Une Contrainte d'Intégrité Fonctionnelle est un type d'association entre 2 entités.
- Une cardinalité maxi à 1 (0,1 ou 1,1) sur l'une des pattes induit obligatoirement une dépendance fonctionnelle.
- Dans l'exemple suivant, on dit usuellement que la relation "est située dans" est (porteuse d') une dépendance fonctionnelle.

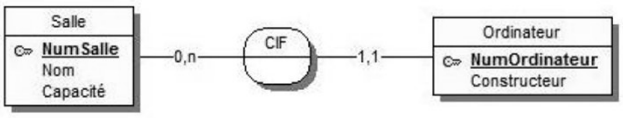
III. 1. Modèle conceptuel de données: MCD

Les différents types d'associations :

❑ **Notion de contrainte d'intégrité fonctionnelle (CIF) :**

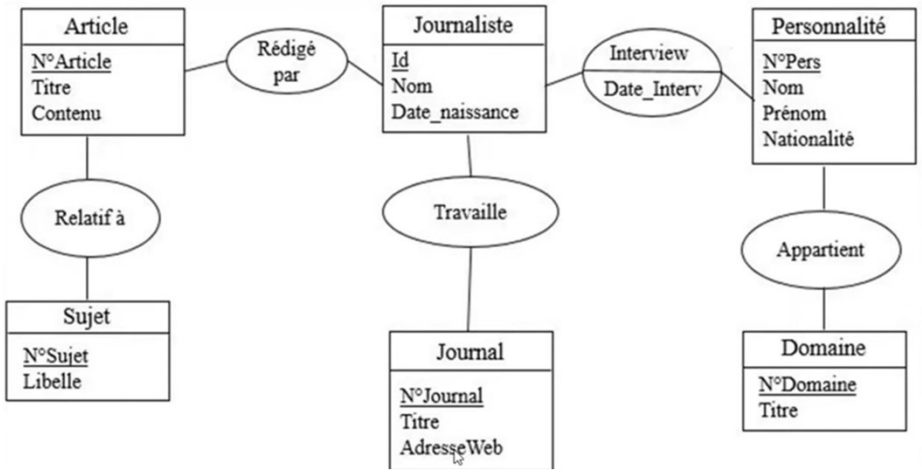


- Certains auteurs proposent une écriture de ce type :



Modèle conceptuel de données: MCD

- Exercice : Trouvez les cardinalités de ce MCD :

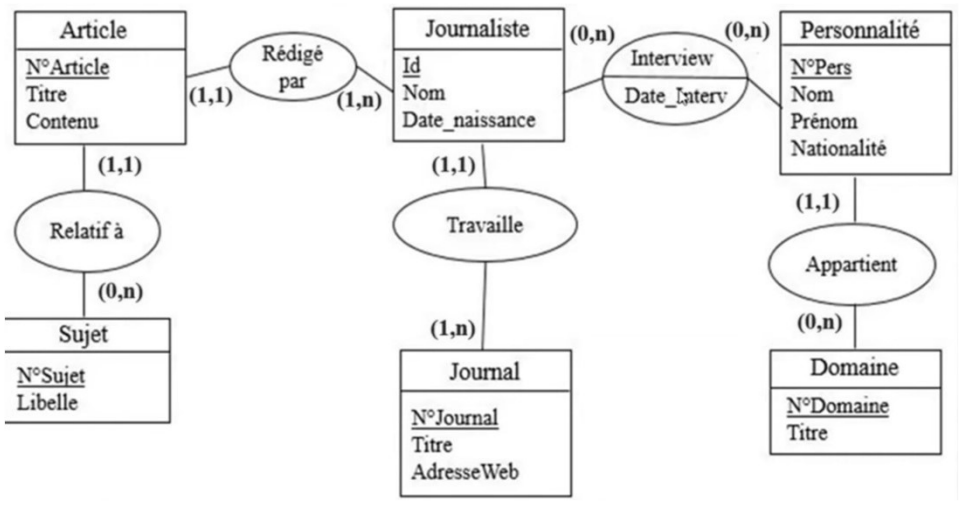


Modèle conceptuel de données: MCD

- Les règles de gestion :
  - Un article est rédigé par un et un seul journaliste, mais un journaliste peut rédiger plusieurs articles.
  - Un article est relatif à un et un seul sujet, mais un sujet peut être abordé dans plusieurs articles ou aucun.
  - Un journaliste travaille pour un et un seul journal, mais un journal peut avoir plusieurs journalistes ou aucun.
  - Un journaliste peut réaliser plusieurs interviews ou aucune et chaque interview est réalisée par un ou plusieurs journalistes.
  - Une personnalité appartient à un et un seul domaine, mais un domaine peut avoir plusieurs personnalités ou aucune.
  - Une personnalité peut être interviewée lors de plusieurs interviews ou aucune.

Modèle conceptuel de données: MCD

- Exercice : Trouvez les cardinalités de ce MCD :



III. 1. Modèle conceptuel de données: MCD Résumé des règles merise

Entités :

- Toute entité doit comporter un identifiant qui permet de distinguer entre elles toutes les occurrences d'une même entité.
- L'identifiant est placé en tête des propriétés et il est souligné.
- Pour chaque occurrence d'une entité, il ne doit y avoir qu'une seule valeur pour chacune des propriétés à un instant donné.
- Cette valeur peut changer au cours du temps, mais à un instant donné, il n'y en a qu'une seule.

Propriétés :

- Toutes les propriétés d'un MCD doivent être différentes les unes des autres.
- Toute propriété ne doit avoir à un moment donné qu'une seule valeur pour une occurrence.
- Un nom de propriété doit toujours être au singulier.
  - **Exemple :** Dans la propriété note d'une entité ELEVE, on ne peut avoir qu'une seule note. Si on veut la note de plusieurs matières, il faut mettre autant de propriétés qu'il y a de matières, par exemple noteINFO, noteCOMPTA, noteCOMM, etc.

Associations :

- L'identifiant d'une association est implicitement formé par la concaténation des identifiants des entités liées.
- On ne représente pas cet identifiant au niveau du MCD.
- Deux occurrences d'association ne peuvent pas avoir le même identifiant.
- Pour une occurrence, l'identifiant ne doit jamais changer de valeur.
- On peut avoir des propriétés, mais ce n'est pas obligatoire.

III. 1. Modèle conceptuel de données: MCD

Exercice d'application :

On souhaite gérer un parc d'animaux, on cite les entités intervenantes dans ce système.

Animal, Espèce, Personne et Aliment

Suite aux règles de gestions suivantes:

- Un animal appartient à une espèce et une seule.
- Un animal peut être aimé par plusieurs personne ou aucun.
- Un animal mange au minimum un aliment.
- Un aliment peut être mangé par plusieurs animaux ou aucun.
- Un aliment mangé par une personne n'est pas un aliment.
- Une personne peut aimer plusieurs animaux ou aucun.

Exemples :

- Animal : Chat, cheval, ...
- Espèce : Mammifère, reptile, ...
- Personne: Omar, Fayrouz, ...
- Aliment : Algues, céréales, ...

Travail à faire :

1/ Etablir un modèle permettant de relier les entités ci-dessus par des associations convenables.
2/ Inscrire les cardinalités sur le modèle.

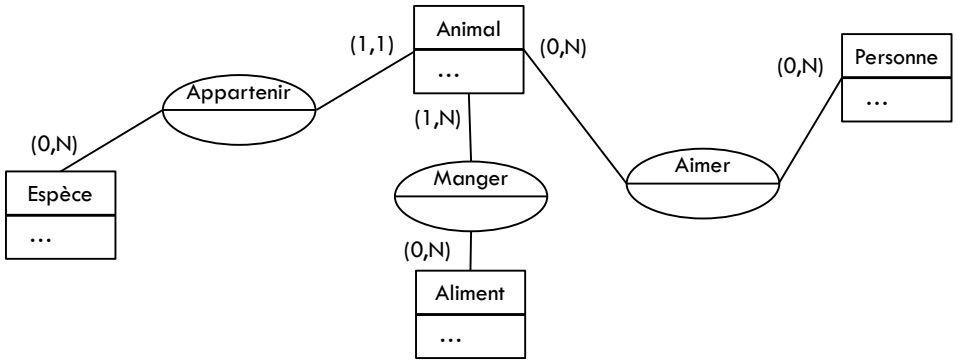
III. 1. Modèle conceptuel de données: MCD

121

Animal, Espèce, Personne et Aliment

Les règles de gestions :

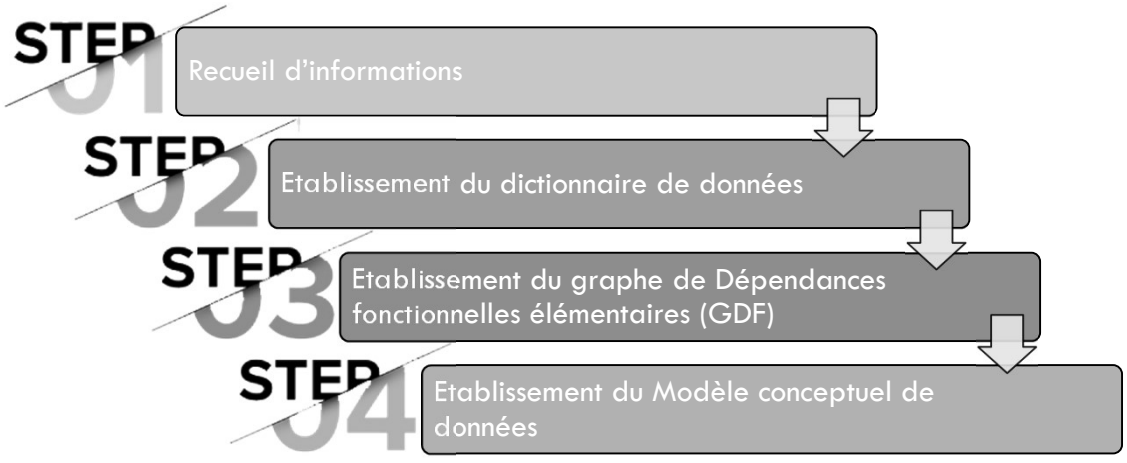
- Un animal appartient à une espèce et une seule.
- Un animal peut être aimé par plusieurs personne ou aucun.
- Un animal mange au minimum un aliment.
- Un aliment peut être mangé par plusieurs animaux ou aucun.
- Un aliment mangé par une personne n'est pas un aliment.
- Une personne peut aimer plusieurs animaux ou aucun.



III. 1. Modèle conceptuel de données: MCD

122

Etapes de construction d'un MCD :



III. 1. Modèle conceptuel de données: MCD

123

Recueil d'informations :

- Le recueil d'information est la première étape à l'informatisation d'un SI. Toutes les informations concernant le sujet doivent être rassemblées.

Pour recenser les informations, on utilise essentiellement :

- L'étude de documents;
- Les entrevues;
- Les questionnaires …

III. 1. Modèle conceptuel de données: MCD

124

Recueil d'informations :

On distingue trois classes des données :

- **Les données élémentaires :** par exemple, un nom, un email,...Ces données doivent être recensées de manière complète.
- **Les données calculées :** elles sont obtenues par l'application d'un traitement mathématique ou logique et souvent associées à des règles de calcul.
- **Les données composées :** certaines données sont regroupées en une même entité sémantique (Adresse, date de naissance, ...)

III. 1. Modèle conceptuel de données: MCD

STEP 01 Recueil d'informations :

Pour être traitées de manière informatisée, les données doivent être décrites dans un formalisme compris par le système informatique qui va les gérer, et pour les formats génériques utilisés on trouve :

- **Le type alphabétique** (rien que des caractères).
- **Le type alphanumérique** (des caractères, des chiffres…).
- **Le type numérique** (les nombres).
- **Le type date.**
- **Le type logique** (0-1, Vrai-Faux, Oui-Non).

Suite à l'interview et la collecte des documents il est nécessaire de centraliser toutes les informations et règles de gestions (calcul d'un taux de remise par exemple) au sein d'un document.

Ce document se nomme le **dictionnaire des données**.

III. 1. Modèle conceptuel de données: MCD

STEP 02 Etablissement du dictionnaire de données :

- Le dictionnaire des données est un document qui permet de recenser, de classer et de trier toutes les informations (les données) collectées lors des entretiens ou de l'étude des documents.
- Ce dictionnaire est un outil important car il constitue la référence de toutes les études effectuées ensuite. .

Exemple :

Nom de la donnée	Format	Longueur	Type		Règle de calcul	Règle de gestion	Document
			Élémentaire	Calculé			

Les données sont présentées dans un tableau.

III. 1. Modèle conceptuel de données: MCD

STEP 2 Etablissement du dictionnaire de données :

- **Exemple d'un dictionnaire de données :**

Nom de la donnée	Format	Longueur	Type		Règle de calcul	Règle de gestion	Document
			E	C			
Nom client	Alphabétique	30	X				Facture

III. 1. Modèle conceptuel de données: MCD

STEP 2 Etablissement du dictionnaire de données :

- Établir la liste à partir de chaque document recueillis (étude de l'existant).
- Une propriété apparait sous deux forme dans un document :
  - Valeur → interpréter la valeur pour identifier la propriété.
    - Exemple : 01/01/1999 → Date
  - Propriété : valeur → propriété explicite
    - Exemple : Prix : 100 DH → Prix

Règles à respecter:

- ✓ Pas de synonymes;
- ✓ Propriétés élémentaires;
- ✓ Propriétés non composées;
- ✓ Propriétés non calculée.

III. 1. Modèle conceptuel de données: MCD

129

STEP 2

Etablissement du dictionnaire de données :

- Lister toutes les propriétés utiles au système étudié.
- Chacune de ces propriétés est définie par :
  - Un nom,
  - Une description (pour éviter toute ambiguïté sur la compréhension de la donnée),
  - Un type de données (numérique, texte, booléen, date, etc.).
- Les entités ainsi que leurs propriétés respectives seront représentées dans le dictionnaire des données.
- Le dictionnaire des données est représenté sous forme de tableau.
- L'identifiant de chaque entité doit être précisé

III. 1. Modèle conceptuel de données: MCD

130

STEP 2

Etablissement du dictionnaire de données :

- **Exemple d'application :**

Suite à une demande d'un membre de notre famille, président d'une association, nous devons établir le dictionnaire des données de la gestion des adhérents.

Une représentation d'une fiche d'adhérent

Nom de la donnée	Format	Longueur	Type		Règle de calcul	Règle de gestion	Document
			Élémentaire	Calculé			

III. 1. Modèle conceptuel de données: MCD

STEP 02

Etablissement du dictionnaire de données :

❑ **Exemple d'application :**

Le dictionnaire des données :

Nom	Format	Longueur	Type		Règle de calcul	Règle de gestion	Document
			E	C			
Numéro	Numérique		X				Fiche
Nom	Alphabétique	30	X				//
Prénom	Alphabétique	30	X				//
Adresse	Alphabétique	50	X				//
Code Postal	Alphanumérique	10	X				//
Ville	Alphabétique	50	X				//
Téléphone	Alphanumérique	15	X				//
Mail	Alphanumérique	50	X				//
Date d'adhésion	Date		X				//

III. 1. Modèle conceptuel de données: MCD

STEP 3 Établissement du Graphe de Dépendances Fonctionnel (GDF) :

- Dépendance fonctionnelle : Propriété1 → Propriété2 si la valeur de propriété 1 détermine celle de propriété 2
- Exemple:
  - NumEtudiant → Prénom
  - NumEtudiant → Nom
- Dépend. fonctionnelle Elémentaire :Si aucune partie stricte de Propriété1 n'entraine Propriété 2
- Exemple:
  - N°bon_de_commande, Réf_produit → Qté_commandée.

III. 1. Modèle conceptuel de données: MCD

STEP 4 Etablissement du Modèle conceptuel de données (MCD) :

- Recherche des entités du système étudié;
- Recherche des propriétés à gérer (dictionnaire des données);
- Recherche des relations entre entités;
- Recherche des cardinalités (règles de gestion);
- Vérification et validation du modèle conceptuel des données.

III. 1. Modèle conceptuel de données: MCD

STEP 4 Etablissement du Modèle conceptuel de données (MCD) :

Règles de transformation du GDF en MCD :

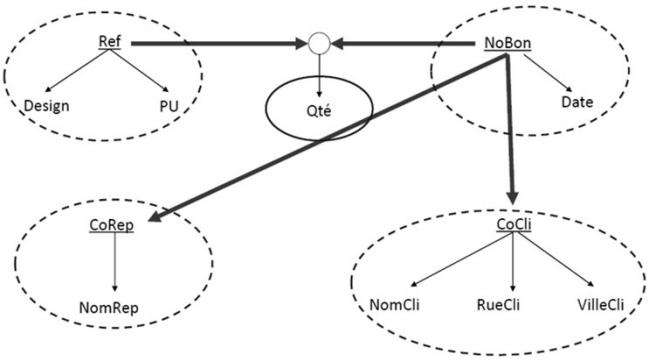
- R0 : Toute donnée du GDF devient une propriété dans le MCD.
- R1 : Les données sources d'au moins une DF (celles qui sont soulignées sur le GDF) représentent les identifiants des entités dont les propriétés sont les cibles de ces DF.
- R2 : Les flèches restantes deviennent des associations. Les données déterminées par une DF conjointe deviennent des propriétés portées par l'association.
- R3 : Les règles de gestion doivent permettre de trouver les cardinalités.

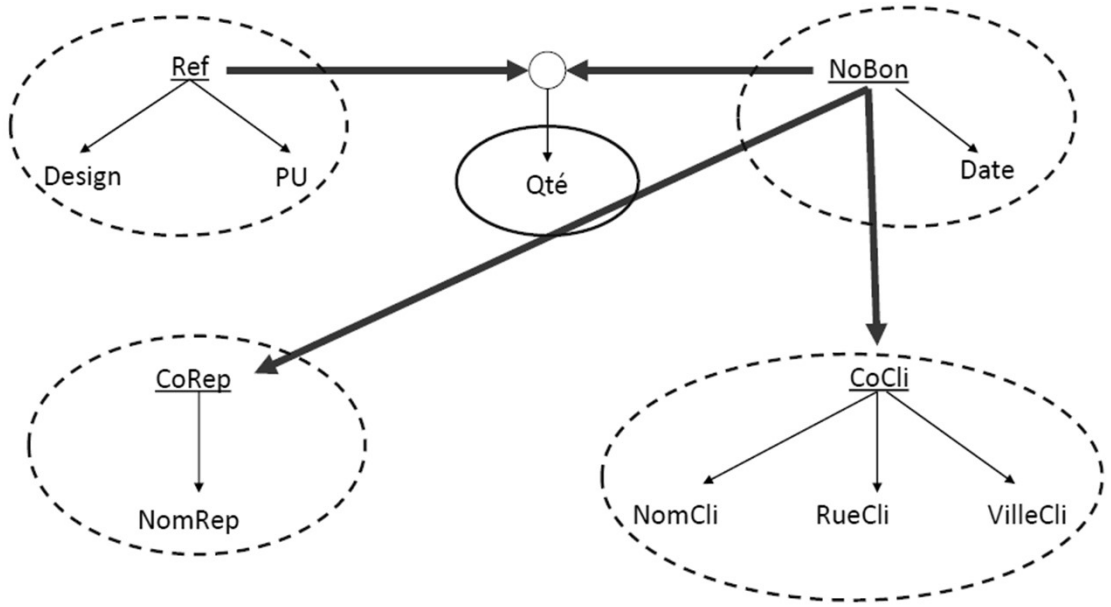
III. 1. Modèle conceptuel de données: MCD

STEP 4 Etablissement du Modèle conceptuel de données (MCD) :

Exemple :

- Transformation du GDF en MCD.





137

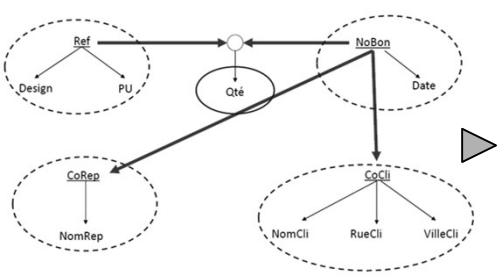
III. 1. Modèle conceptuel de données: MCD

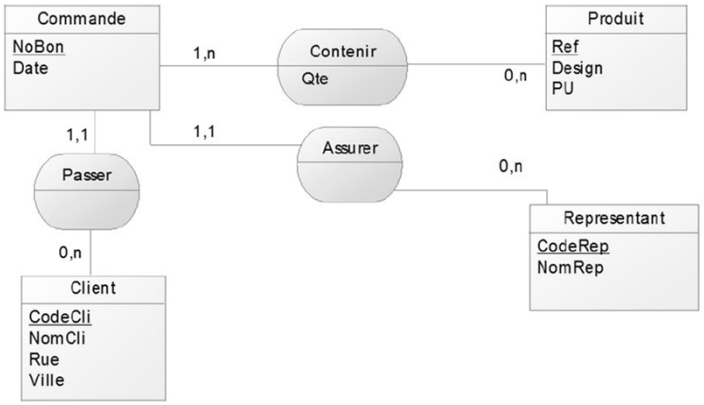
138

STEP 4 **Etablissement du Modèle conceptuel de données (MCD) :**

Exemple :

- Transformation du GDF en MCD.





III. 1. Modèle conceptuel de données: MCD

Etapes de construction d'un MCD :

Exercice :

On veut représenter la gestion d'une bibliothèque:

Nous devons représenter :

- Des livres avec : numéro du livre (ISBN), Titre du livre
- Des auteurs avec : Numéro de l'auteur, nom de l'auteur
- Des éditeurs avec : Numéro d'éditeur, nom de l'éditeur
- Des dépôts avec : Numéro de dépôt, nom du dépôt

Attention : ici un « livre » n'est pas le « livre physique » (un exemplaire) mais plutôt une «édition »

L'investigation du domaine a permis de définir les règles suivantes :

- Un livre peut être : Écrit par plusieurs auteurs
- Édité par plusieurs éditeurs, mais une seul fois par chacun d'entre eux.
- Pour distinguer, on donne alors l'année éditions
- Stocké dans plusieurs dépôts, et cela pour chaque éditeur.
- Chaque livre stocké est stocké avec une quantité définie.

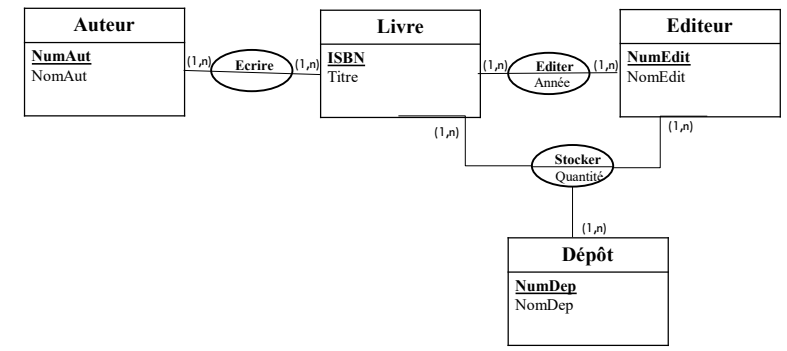
Donner le MCD et les éventuels contraintes d'intégrité correspondant à cet énoncé.

III. 1. Modèle conceptuel de données: MCD

Etapes de construction d'un MCD :

Exercice :

MCD:

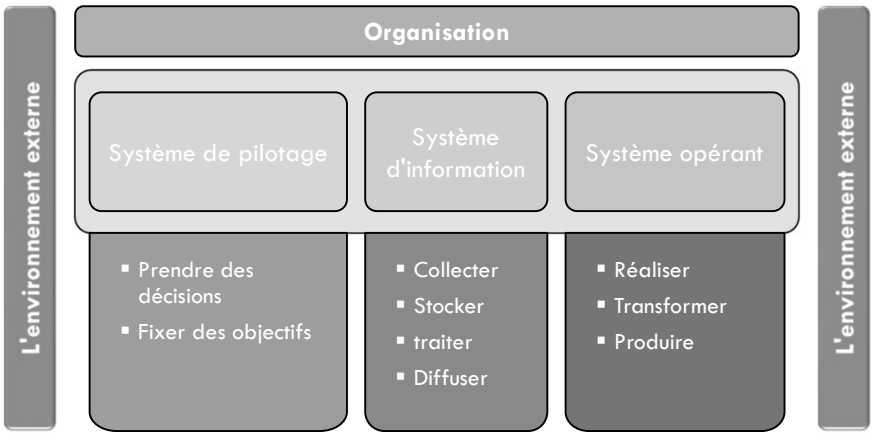


TD 1 Élaboration d'un Modèle conceptuel de données

Travaux dirigés : Série I

I :

- Quelle est la structure d'une organisation, ses composants, les interactions et le rôle de chacun.



III. 1. Modèle conceptuel de données: MCD

Cas pratique :

Vous êtes le propriétaire d'un camping dans les Pyrénées orientales, et vous voulez le gérer d'une façon informatisé a travers d'un système d'information pour la partie des achats de l'épicerie ou du bar.

Le camping est ouvert du 1er juin au 30 septembre. Ils disposent de cinquante emplacements sur un terrain d'une superficie totale de quarante hectares.

Vous avez déjà un logiciel spécialisé dans la réservation des emplacements qui fonctionne très bien mais qui ne permet pas de gérer les achats de l'épicerie ou du bar selon vos règles de gestion. En effet, les vacanciers ne payent leurs achats qu'à la fin de leur séjour. Concrètement, les achats sont inscrits manuellement sur une fiche bristol créée pour chaque famille de vacanciers. À la fin du séjour, les cumuls sont réalisés et une facture manuelle concernant les achats est établie.

Vous souhaitez disposer d'un logiciel pour automatiser la création de la facture grâce à la saisie journalière des achats.

Camping de la source

Liste des Achats

Nom : BAPTISTE
Prénom : Jean-Luc
Adresse : Rue de la forêt
Code Postal : 12000
Ville : Rodez
Téléphone : 05-65-42-00-00

Date	Désignation	Qté	Prix	Total
14/7/08	Repas « Cargolade »	4	22	88
15/7/08	Café	1	1,20	1,20
15/7/08	Glace « Magnum »	2	2,10	4,20
16/7/08	Baguette	1	1,15	1,15

Total dû : 94,55

Une représentation de la fiche bristol

III. 1. Modèle conceptuel de données: MCD

Cas pratique :

Travail à faire :

- ❑ Etablir le dictionnaire de données.
- ❑ Etablir le graphe de Dépendances fonctionnelles.
- ❑ Etablir le MCD

III. 1. Modèle conceptuel de données: MCD

Cas pratique :

1. Dictionnaire des données :

- Il va être nécessaire de rajouter deux informations non présentes : le numéro du client et le code de l'article.

Nom	Format	Longueur	Type		Règle de calcul	Règle de gestion	Document
			E	C			
NumCli	Numérique		X				Bristol
Nom	Alphabétique	30	X				//
Prénom	Alphabétique	30	X				//
Adresse	Alphabétique	50	X				//
Code Postal	Alphanumérique	10	X				//
Ville	Alphabétique	50	X				//
CodeArticle	Alphanumérique	15	X				//
Désignation	Alphabétique	50	X				//
PrixUnitaire	Numérique		X				//
Qté	Numérique		X				//
Date	Date		X				//
TotalLigne	Numérique			X	PrixUnitaire x Qté		//
TotalFacture	Numérique			X	Somme des TotalLigne		//

III. 1. Modèle conceptuel de données: MCD

Cas pratique :

2.1. Détermination des dépendances fonctionnelles :

- Dépendances fonctionnelles élémentaires pour les clients
  - Numcli → Prénom
  - Numcli → Adresse
  - Numcli → Code Postal
  - Numcli → Ville
- Dépendances fonctionnelles élémentaires pour les articles :
  - CodeArticle → Désignation
  - CodeArticle → PrixUnitaire

III. 1. Modèle conceptuel de données: MCD

Cas pratique :

2.1. Détermination des dépendances fonctionnelles :

- Nous nous rendons compte que cette donnée Qté fait partie d'une dépendance fonctionnelle composée.
  - (Numcli, CodeArticle, Date) → Qté

Les DF auraient pu s'écrire de la façon suivante :

Numcli → (Nom, Prénom, Adresse, Code Postal, Ville),

CodeArticle → (Désignation, PrixUnitaire).

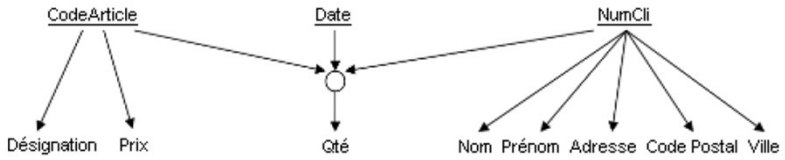
(Numcli, CodeArticle, Date) → Qté

III. 1. Modèle conceptuel de données: MCD

Cas pratique :

2.2. Graphe des dépendances fonctionnelles :

- Le graphe des dépendances fonctionnelles est une étape intéressante car il épure le dictionnaire en ne retenant que les données non déduites et élémentaires et il permet une représentation spatiale de ce que sera le futur modèle conceptuel des données.



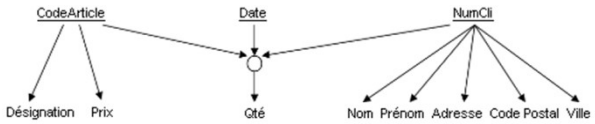
III. 1. Modèle conceptuel de données: MCD

Cas pratique :

3. Etablissement du Modèle conceptuel de données (MCD) :

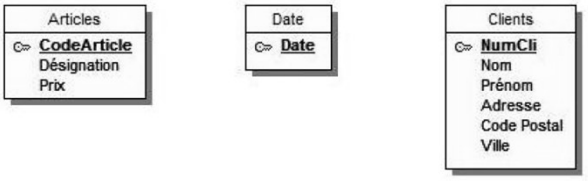
Commençons par déterminer les entités. Par rapport au graphe, nous pouvons remarquer trois sources de dépendances fonctionnelles :

- CodeArticle
- Date
- NumCli



Chacune de ces sources peut représenter une entité :

- Articles
- Date
- Clients



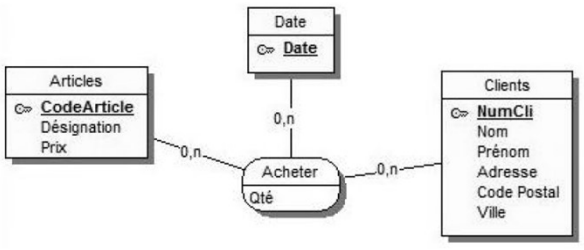
III. 1. Modèle conceptuel de données: MCD

Cas pratique :

3. Etablissement du Modèle conceptuel de données (MCD) :

Traçons les relations :

- Nous savons qu'une quantité d'articles est achetée par un client à une date donnée.
- Nous voyons qu'il existe une relation entre les trois entités.



III. 1. Modèle conceptuel de données: MCD

151

Exercice d'application :

Soit une liste des données recensées dans un établissement scolaire :

- Nom de l'étudiant
- Prénom de l'étudiant
- Email de l'étudiant
- Libellé matière
- Nombre d'heures
- Code filière
- Libellé filière
- Note
- Numéro de l'étudiant
- Numéro de la matière
- Langue d'enseignement

Les règles de gestion appliquées dans cet établissement:

- **RG1 :** Chaque étudiant est attribué à une et une seule filière.
- **RG2 :** Une matière est enseignée pour différentes filières avec des nombres d'heures différents.
- **RG3 :** Pour tout étudiant, chaque matière est évaluée par une note.

Travail à faire :

- Citer les différentes dépendances fonctionnelles.
- Déduire le GDF associé.
- Transformer le GDF en modèle conceptuel de données (MCD)

III. 1. Modèle conceptuel de données: MCD

152

Exercice d'application :

Soit une liste des données recensées dans un établissement scolaire :

- Nom de l'étudiant
- Prénom de l'étudiant
- Email de l'étudiant
- Libellé matière
- Nombre d'heures
- Code filière
- Libellé filière
- Note
- Numéro de l'étudiant
- Numéro de la matière
- Langue d'enseignement

N° ETUDIANT → Nom étudiant, Prénom étudiant, Email

N° MATIERE → Libellé matière; Langue

CODE FILIERE → Libellé filière

N° ETUDIANT, N° MATIERE → Note

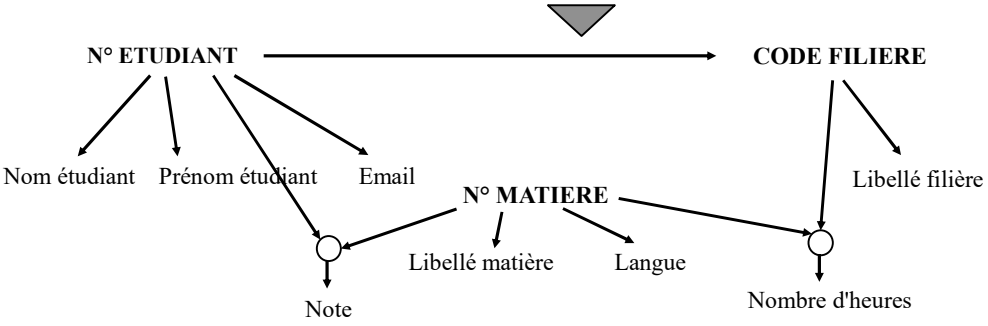
N° MATIERE, CODE FILIERE → Nombre d'heures

III. 1. Modèle conceptuel de données: MCD

153

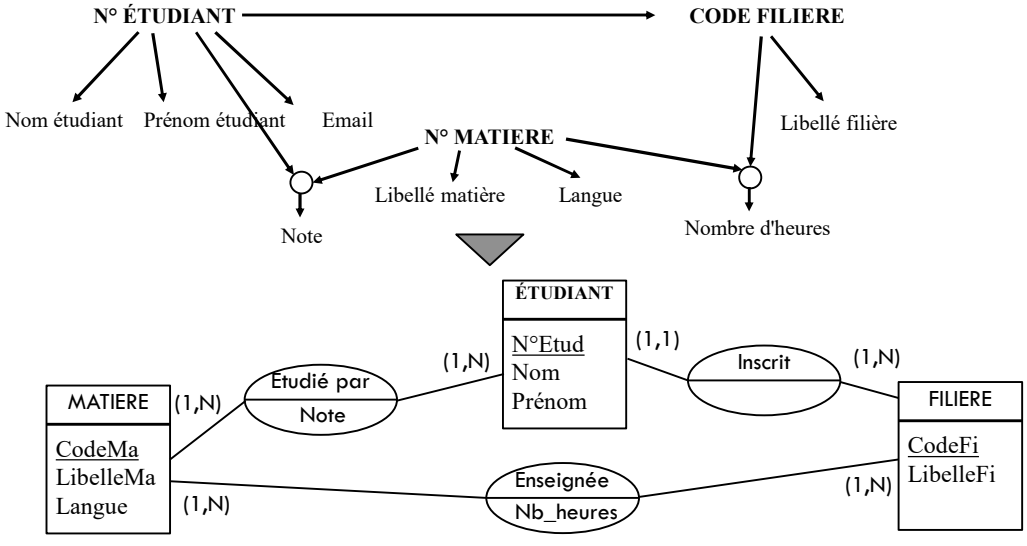
Exercice d'application :

N° ETUDIANT → Nom étudiant, Prénom étudiant, Email
N° MATIERE → Libellé matière; Langue
CODE FILIERE → Libellé filière
N° ETUDIANT, N° MATIERE → Note
N° MATIERE, CODE FILIERE → Nombre d'heures



III. 1. Modèle conceptuel de données: MCD

154



III. 2. Modèle logique de données : MLD

Introduction au Modèle Logique des Données :

- Le Modèle Logique des Données (MLD) est la suite normale du processus Merise.
- Le MLD constitue une étape intermédiaire entre le modèle conceptuel et le modèle physique de données.
- Son but est de nous rapprocher au plus près du modèle physique.
- Pour cela, nous partons du Modèle Conceptuel des Données et nous lui enlevons les relations, mais pas n'importe comment, il faut en effet respecter certaines règles;
- A ce niveau, on doit choisir le mode d'organisation des données :
  - Modèle relationnel
  - Modèle objet
  - Modèle hiérarchique
  - Modèle réseau
  - …

III. 2. Modèle logique de données : MLD

Modèle Relationnel de Données :

- Modèle défini en 1970 par Codd
  - « A Relational Model of Data for Large Shared Data Banks » (1)
- Un modèle très simple et efficace.
- La majorité des SGBD (Systèmes de Gestion de Base de Données) commercialisés se basent sur le modèle relationnel.
- Les premiers SGBD bâtis sur ce modèle : SQL/DS et DB2 de IBM.
  - D'où est né le langage de manipulation de bases relationnelles, SQL (Structured Query Language).

(1) Codd, Edgar F. **"A relational model of data for large shared data banks."** *Software pioneers*. Springer, Berlin, Heidelberg, 2002. 263-294.

III. 2. Modèle logique de données : MLD

Modèle Relationnel de Données :

- Une base de données = Ensemble de **relations (ou tables)**
- Chaque relation a un ensemble d'**attributs (ou colonnes)**
- Chaque **ligne (ou tuple) contient une valeur pour chaque** attribut de la relation.
- Chaque attribut a un **type (ou domaine).**

Etudiant

ID	Nom	Prenom	Moyenne
238	Majidi	Ahmed	13
239	Karimi	Nisrine	12
240	Latifi	Yassine	15
...	...	...	...

Institut

Nom_Institut	Ville	Effectif
FP	Khouribga	3000
FSJES	Settat	20000
...	...	...

III. 2. Modèle logique de données : MLD

Modèle Relationnel de Données :

- **Schéma Relationnel = Description structurelle des relations de** la base de données
  - **Etudiant** (ID, Nom, Prenom, Moyenne)
  - **Ecole** (Nom_Institut, Ville, Effectif)
- **Instance d'une relation = contenu à un instant de temps donné**

Etudiant

ID	Nom	Prenom	Moyenne
238	Majidi	Ahmed	13
239	Karimi	Nisrine	12
240	Latifi	Yassine	15
...	...	...	...

Institut

Nom_Institut	Ville	Effectif
FP	Khouribga	3000
FSJES	Settat	20000
...	...	...

III. 2. Modèle logique de données : MLD

Modèle Relationnel de Données :

- **NULL = Valeur spéciale pour « inconnu » ou « indéfini »**
  - **Attention: NULL ≠ 0**
- **Clé = attribut à valeur unique pour chaque tuple,** ou combinaison d'attributs à valeurs combinées uniques.

Etudiant

ID	Nom	Prenom	Moyenne
238	Majidi	Ahmed	13
239	Karimi	Nisrine	12
240	Latifi	Yassine	15
...	...	...	...

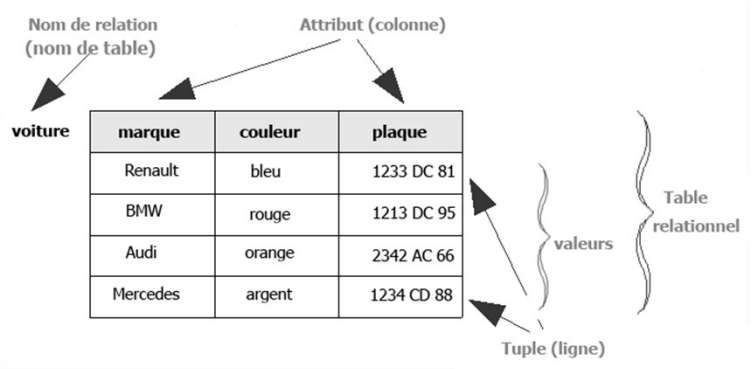
Institut

Nom_Institut	Ville	Effectif
FP	Khouribga	3000
FSJES	Settat	20000
...	...	...

III. 2. Modèle logique de données : MLD

Modèle Relationnel – Vocabulaire :

- Domaines;
- Relations;
- Attributs;
- Schéma d'une relation;
- Tuples;
- Clé primaire;
- Clé étrangère;
- Schéma d'une BDR;
- Base de données relationnelle.



III. 2. Modèle logique de données : MLD

Modèle Relationnel – Vocabulaire :

- Domaine :
  - Un domaine D est un ensemble de valeurs atomiques (distinctes).
  - C'est l'ensemble des valeurs que peut prendre une entité du monde réel.
  - Un domaine s'apparente souvent à un type simple (entier, réel, caractère, …), mais peut également être composé (date, image …)

Exemple :

- L'ensemble des entiers est un domaine
- {'assez bien', 'bien', 'très bien'}
- {'Mohammed', 'Ahmed', ….}
- {oui, non}

III. 2. Modèle logique de données : MLD

Modèle Relationnel – Vocabulaire :

- Relation :
  - Une relation, au sens mathématique, est un sous-ensemble du produit cartésien d'un certain nombre de domaines.
    - **R = (D1 X D2 X … X Dn)** est la relation R définie sur les domaines D1, D2, D3, …Dn.
  - La réalisation d'une relation est représentée par une table à colonnes D1, D2, …, Dn dont les lignes seront concrétisées par les différentes valeurs possibles prises dans les domaines D1, D2, …, Dn.

Réf	Désignation	PU
AB1234	Ordinateur de bureau HP	10 000
C6543	Table pour ordinateur	1000
…	…	…

III. 2. Modèle logique de données : MLD

Modèle Relationnel – Vocabulaire :

❑ Attribut :

- Un attribut (d'une relation) est le nom donné à une colonne d'un tableau représentant une relation.

Exemples :

Réf, Désignation, Nom, Prénom, Date, Couleur …

III. 2. Modèle logique de données : MLD

Modèle Relationnel – Vocabulaire :

❑ Schéma de relation :

- Un schéma de relation est composé du nom de la relation suivi de ses attributs et de leurs domaines.
  - Notation : R (A1 : D1 , A2 : D2 , ... , An : Dn).
  - Si le choix des domaines est évident, on simplifie la notation :
  - R (A1 , A2 , ... , An).

Exemples :

- Produit (Référence, Désignation, PrixUnitaire)
- Etudiant (CNE, Nom, Prénom, DateNaissance, Adresse)

III. 2. Modèle logique de données : MLD

Modèle Relationnel – Vocabulaire :

- Tuples :
  - Un tuple ou n-uplet (élément) correspond à une ligne d'une relation.

Exemples :

- (AB1234, 'Ordinateur de bureau HP , 10000)

- Clé primaire d'une relation :
  - Attribut à valeur unique pour chaque tuple. Ou combinaison d'attributs à valeurs combinées uniques.

Exemples :

- Réf pour la relation produit.

III. 2. Modèle logique de données : MLD

Modèle Relationnel – Vocabulaire :

- Clé étrangère d'une relation :
  - Les clés étrangères permettent de gérer les relations entre plusieurs tables, et garantissent la cohérence des données.

- Schéma d'une BDR :
  - Le schéma d'une base de données relationnelle est l'ensemble des schémas des relations qui la compose.

- Base de données relationnelle :
  - Une BDR est constituée de l'ensemble des tuples de toutes les relations définies dans le schéma de la base.

III. 2. Modèle logique de données : MLD

Règles d'intégrité :

- Les règles d'intégrité (R.I.) sont des **conditions** qui doivent être vérifiées à tout moment par les données contenues dans la base de données :
  - Intégrité de **domaine;**
  - Intégrité de **relation(table);**
  - Intégrité de **référence;**
  - Intégrité **d'entité,**

III. 2. Modèle logique de données : MLD

Règles d'intégrité :

1. Intégrité de Domaine :

- **Contrôle des valeurs des attributs**
- **Contrôle entre valeurs des attributs**
  - Age [0 .. 130]
  - Date_Début < Date_Fin
  - Montant > 0

2. Intégrité de relation (Contrainte De Clé)

Chaque relation doit posséder une clé primaire → Unicité des tuplets

III. 2. Modèle logique de données : MLD

Règles d'intégrité :

3. Intégrité de référence (Clé étrangère)

- Impose que tout tuple d'une relation **R1**qui se réfère à une relation **R2** doit se référer à un tuple existant dans la relation **R2**.
- Elle s'applique sur des relations qui décrivent des associations.

Exemple :

On définira qu'un livre a un ou plusieurs auteurs. Une contrainte d'intégrité référentielle interdira l'effacement d'un auteur, tant que dans la base de données il existera au moins un livre se référant à cet auteur. Cette contrainte interdira également d'ajouter un livre si l'auteur n'est pas préalablement inscrit dans la base de données.

III. 2. Modèle logique de données : MLD

Règles d'intégrité :

4. Intégrité d'entité

- Impose que tout attribut faisant partie de la clé d'une relation soit **non nul**

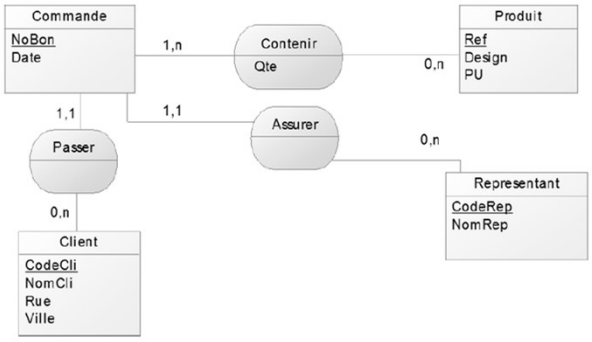
III. 2. Modèle logique de données : MLD

171

Construction d'un modèle relationnel (MLD) :

- Le modèle relationnel peut être déduit à partir du modèle MCD en appliquant des règles de passage MCD-MLD simples

Exemple : **Déduire le MLD correspondant au MCD suivant :**



III. 2. Modèle logique de données : MLD

172

Construction d'un modèle relationnel (MLD) :

Règles de passages MCD → MLD :

- **Règle 1 :**
  - Entité → Table
  - Propriété → Attribut
  - Identifiant → Clé primaire
- Exemple :
  - **Commande (<u>NoBon</u>, Date)**
  - **Produit (<u>Ref</u>, Design, PU)**
  - **Client (<u>CodeCli</u>, NomCli, Rue, Ville)**
  - **Représentant (<u>CodeRep</u>, NomRep)**

III. 2. Modèle logique de données : MLD

Construction d'un modèle relationnel (MLD) :

Règles de passages MCD → MLD :

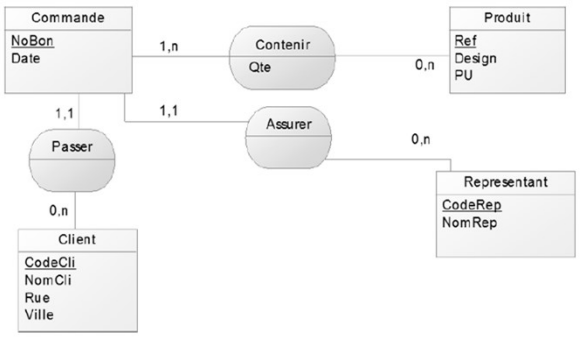
- **Règle 2 :**
- Une association binaire ayant des cardinalités (x, 1) et (x, n), x étant égale à 0 ou 1, se traduit par :
  - La migration de l'identifiant de l'entité ayant la cardinalité (x, n) vers l'entité ayant la cardinalité (x,1)
  - Cet identifient devient une **clé étrangère.**

III. 2. Modèle logique de données : MLD

Construction d'un modèle relationnel (MLD) :

Règles de passages MCD → MLD :

- **Règle 2 :**
- Exemple :
  - Commande (NoBon, #CodeCli, #CodeRep, Date)
  - Produit (Ref, Design, PU)
  - Client (CodeCli, NomCli, Rue, Ville)
  - Représentant (CodeRep, NomRep)

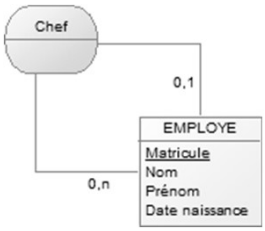


III. 2. Modèle logique de données : MLD

Construction d'un modèle relationnel (MLD) :

Règles de passages MCD → MLD :

- **Règle 2 :**
- Exemple (association réflexive) :
  - Employé (Matricule, #MatriculeChef, nom, prénom, date_naissance)



III. 2. Modèle logique de données : MLD

Construction d'un modèle relationnel (MLD) :

Règles de passages MCD → MLD :

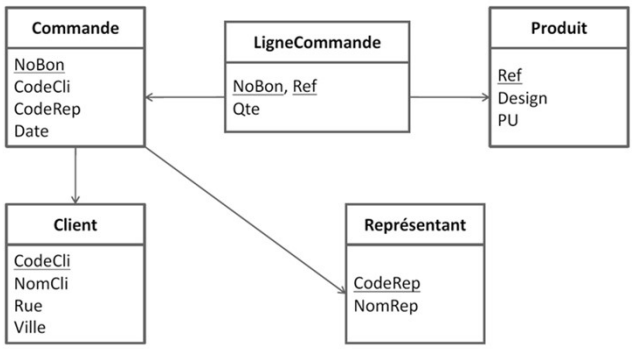
- **Règle 3 :**
- Une association binaire ayant des cardinalités (x, n) et (x, n), x étant égale à 0 ou 1, se traduit par :
  - la création d'une nouvelle relation
  - la migration de l'identifiant de chacune des entités vers la nouvelle relation. L'ensemble de ces identifiant constituent la **clé primaire.**
  - Chacun des identifiants devient une clé étrangère.
  - Les propriétés de l'association constituent le reste des attribues de la nouvelle table.
- Dans le cas d'association réflexive, l'identifiant est dupliqué puis renommé.

III. 2. Modèle logique de données : MLD

Construction d'un modèle relationnel (MLD) :

Règles de passages MCD → MLD :

- Exemple: MLD



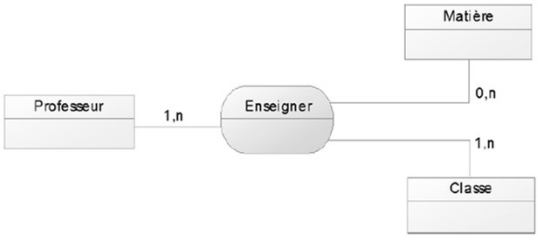
III. 2. Modèle logique de données : MLD

Construction d'un modèle relationnel (MLD) :

Règles de passages MCD → MLD :

❑ **Règle 4 :**

- Une association n-aire (de dimension n > 2) porteuse ou non de propriétés, se transforme en une relation ayant comme clé primaire la composition de l'ensemble des identifiants de la collection et comme attributs ceux de l'association.
- Exemple:
  - Enseignement (CodeProf, CodeMat , CodeClasse)



III. 2. Modèle logique de données : MLD

Construction d'un modèle relationnel (MLD) :

Règles de passages MCD → MLD :

- Le passage du modèle conceptuel au Modèle Logique des Données est purement mécanique, il suffit de respecter les règles dédiées.
- Il n'y a plus de travail de conceptualisation ou de réflexion proprement dit. Lorsque nous réalisons un Modèle Logique des Données nous ne faisons que « détruire » un Modèle Conceptuel des Données pour recréer un autre modèle.

III. 3. Modèle physique de données : MPD

Construire le Modèle Physique des Données :

- Construire le Modèle Physique des Données consiste à transformer le Modèle Logique des Données en une suite de relations.
- Cette étape finalise le processus de traitement des données.
- L'implémentation des bases de données peut être réalisée de façon optimale.
- Passer du modèle logique de données au modèle physique des données ne présente aucune difficulté.
- On abandonne juste la représentation graphique pour une représentation plus linéaire.

III. 3. Modèle physique de données : MPD

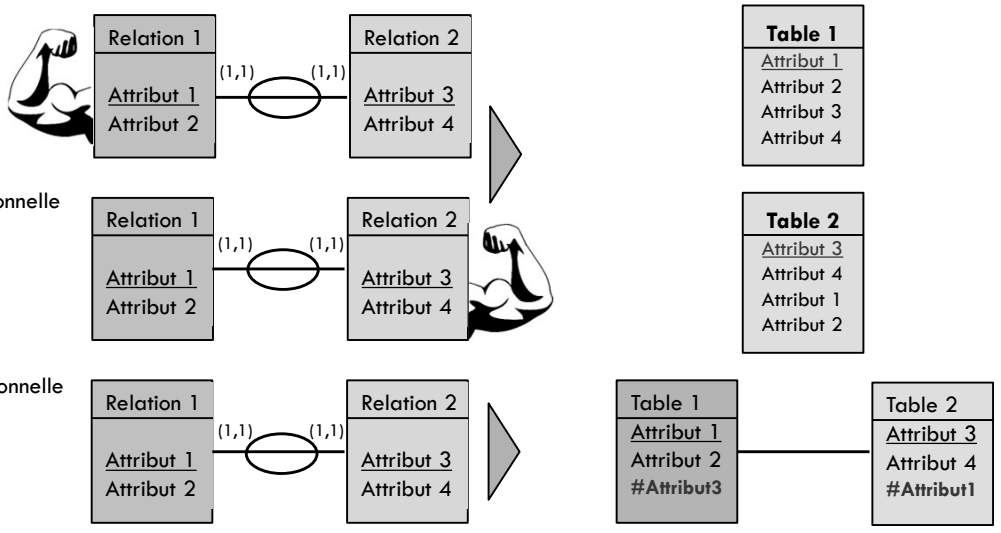
Construire le Modèle Physique des Données :

- Exemple d'un MPD :
  - Commande (NoBon, #CodeCli, #CodeRep, Date)
  - Produit (Ref, Design, PU)
  - Client (CodeCli, NomCli, Rue, Ville)
  - Représentant (CodeRep, NomRep)
  - Contenir (#NoBon, #Ref, Qte)
    - LigneCommande (#NoBon, #Ref, Qte)

Cas particuliers MCD > MLD : Associations (x,1), (x,1)

Les relations binaires possibles : **(1,1)-(1,1)**; (0,1)-(1,1); (1,1)-(0,1); (0,1)-(0,1)

❑ (1,1) - (1,1)



Avec importance fonctionnelle

❑ (1,1) - (1,1)

Sans importance fonctionnelle

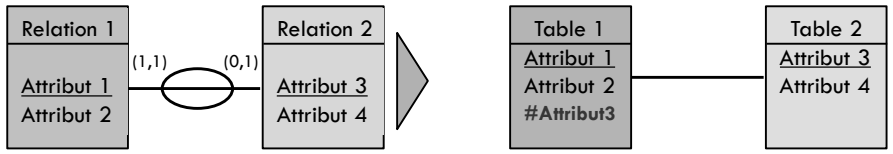
❑ (1,1) - (1,1)

Cas particuliers MCD > MLD : Associations (x,1), (x,1)

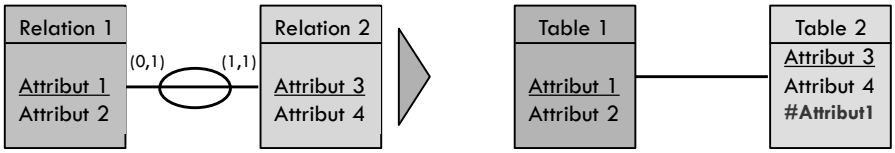
Les relations binaires possibles : (1,1)-(1,1); **(0,1)-(1,1); (1,1)-(0,1)**; (0,1)-(0,1)

183

- (1,1) - (0,1)



- (0,1) - (1,1)

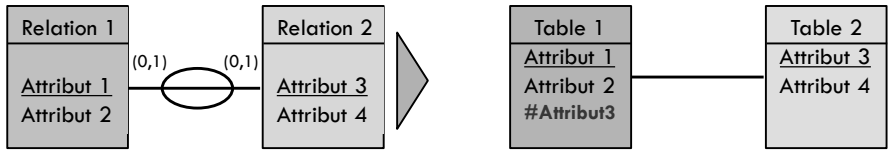


Cas particuliers MCD > MLD : Associations (x,1), (x,1)

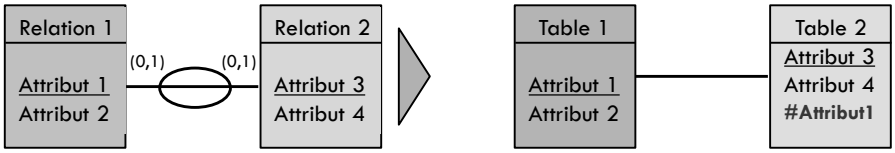
Les relations binaires possibles : (1,1)-(1,1); (0,1)-(1,1); (1,1)-(0,1); **(0,1)-(0,1)**

184

- (0,1) - (0,1)



- (0,1) - (0,1)



III. 4. Normalisation: formes normales

Les formes normales :

Pour être parfaite, les relations doivent respecter certaines règles.

Cet ensemble de règles se nomme : **les formes normales**.

Cette théorie a été élaborée par E.F. Codd en 1970. Son objectif est d'éviter les anomalies dans les bases de données relationnelles :

- **Problèmes de mise à jour.**
- **Suppression des redondances d'informations.**
- **Simplification de certaines contraintes d'intégrité.**

Pour parfaire une base de données relationnelle, il est nécessaire de connaître les trois premières formes normales.

III. 4. Normalisation: formes normales

Les formes normales :

4.1. 1ère forme normale (1FN) :

- Une entité est en 1ère FN si tout attribut est atomique et contient un identifiant unique;
- Les attributs ne contiennent pas de valeurs répétitives.

- Conséquences :
  - Un attribut représente une donnée élémentaire du monde réel
  - Un attribut ne peut désigner, ni une donnée composée d'entités de nature quelconque, ni une liste de données de même nature.

III. 4. Normalisation: formes normales

Les formes normales :

4.1. 1ère forme normale (1FN) :

❑ Exemple : ***Clients (NumCli, Nom, Prénom, Adresse, Téléphone)***

- Cette relation n'est pas en première forme normale, car **Adresse n'est pas atomique**.

NumCli	Nom	Prénom	Adresse	Téléphone
1	Baptiste	JeanLuc	25, rue de la forêt 12000 Rodez	0565420000
2	Auguy	Joel	Impasse des lys 15000 Aurillac	0471670000
3	Rascalou	André	2, rue droite 12000 Rodez	0565450000

- Cette représentation en pratique génère un accès aux données plus lent.
- Extraire les habitants d'une ville précise devra mettre en œuvre des procédures d'extraction sans fournir de garantie quant au résultat retourné.

III. 4. Normalisation: formes normales

Les formes normales :

4.1. 1ère forme normale (1FN) :

❑ Exemple : **Voici une représentation 1FN correcte :**

Clients (NumCli, Nom, Prénom, Adresse, CodeP, Ville, Téléphone)

NumCli	Nom	Prénom	Adresse	Code Postal	Ville	Téléphone
1	Baptiste	JeanLuc	25, rue de la forêt	12000	Rodez	0565420000
2	Auguy	Joel	Impasse des lys	15000	Aurillac	0471670000
3	Rascalou	André	2, rue droite	12000	Rodez	0565450000

- Récupérer les habitants d'une ville précise ne pose plus aucun problème, une simple requête SQL y parviendra de façon rapide et fiable.

III. 4. Normalisation: formes normales

189

Les formes normales :

4.2. 2ème forme normale (2FN) :

Une relation est en deuxième forme normale si :

- Elle est en première forme normale.
- Si tous les attributs non clés ne dépendent pas d'une partie de la clé primaire.
  - Autrement dit, toute propriété de la relation doit dépendre intégralement de toute la clé (dépendances fonctionnelle élémentaires).

.

III. 4. Normalisation: formes normales

190

Les formes normales :

4.2. 2ème forme normale (2FN) :

- Exemple : ***Commande (<u>Numcli, CodeArticle, Date</u>, Qté commandée, Désignation)***
  - Cette relation est elle en première forme normale ? **Oui.**
  - Est elle en deuxième forme normale ? **Non**

Car la propriété Désignation ne dépend pas intégralement de la clé (Numcli, CodeArticle, Date).

- Connaissant {1,Art1,28/02/2009} pouvons-nous connaître de façon sûre et unique « Bocal d'un kilo de Tripoux » ?
- La réponse est évidemment non ! « Bocal d'un kilo de Tripoux » ne dépend pas intégralement de la clé {1,Art1,28/02/2009}.

NumCli	CodeArticle	Date	Qté commandée	Désignation
1	Art1	28/02/2009	5	Bocal d'un kilo de Tripoux
3	Art2	28/02/2009	9	Aligot congelé
5	Art3	28/02/2009	10	Cèpes séchés
6	Art41	28/02/2009	15	Bouteille de Marcillac rouge

III. 4. Normalisation: formes normales

Les formes normales :

4.2. 2ème forme normale (2FN) :

- Exemple : Voici comment corriger :
  - Commandes (Numcli, CodeArticle, date, Qté commandée)
  - Articles (CodeArticle, Désignation)

Commandes

NumCli	CodeArticle	Date	Qté commandée
1	Art1	28/02/2009	5
3	Art2	28/02/2009	9
5	Art3	28/02/2009	10
6	Art41	28/02/2009	15

Articles

CodeArticle	Désignation
Art1	Bocal d'un kilo de Tripoux
Art2	Aligot congelé
Art3	Cèpes séchés
Art41	Bouteille de Marcillac rouge

III. 4. Normalisation: formes normales

Les formes normales :

4.3. 3ème forme normale (3FN) :

L'objectif de cette 3ème FN est l'élimination des redondances dues aux DF déduites par transitivité.

Une relation est en 3ème FN si :

- Elle est en 2ème FN.
- Si toutes les dépendances fonctionnelles par rapport à la clé sont directes
  - Autrement dit, tous les attributs n'appartenant pas à la clé ne dépendent pas d'un attribut non clé.

III. 4. Normalisation: formes normales

Les formes normales :

4.3. 3ème forme normale (3FN) :

❑ Exemple :

La relation Commande (NuméroCommande, #CodeClient, Nom client, #RefArticle) est elle en troisième forme normale ?

Est elle en première forme normale ? **Oui.**

Est elle en deuxième forme normale ? **Oui.**

Est elle en troisième forme normale ? **Non !**

NuméroCommande	CodeClient	Nom client	RefArticle
1	C1	Baptiste	Art25
3	C5	Savary	Art20
5	C2	Martinez	Art10
6	C1	Baptiste	Art15

En effet Nom client dépend d'une propriété non clé : CodeClient

III. 4. Normalisation: formes normales

Les formes normales :

4.3. 3ème forme normale (3FN) :

❑ Exemple : Voici comment corriger :

Commande (NuméroCommande, #CodeClient, #RefArticle) Clients (CodeClient, Nom client)

NuméroCommande	CodeClient	RefArticle
1	C1	Art25
3	C5	Art20
5	C2	Art10
6	C1	Art15

Commande

Clients

CodeClient	Nom client
C1	Baptiste
C2	Martinez
C5	Savary

III. 2. Modèle logique de données : MLD

Exercice :

- La relation suivante décrit des commandes faites par des clients, avec les produits et quantités commandées par client.

  Commandes (NumCom, DateCom, NumCli, AdrCli, NumProd, Prix, Qte)

  a. Quelle est la clé de cette relation ?

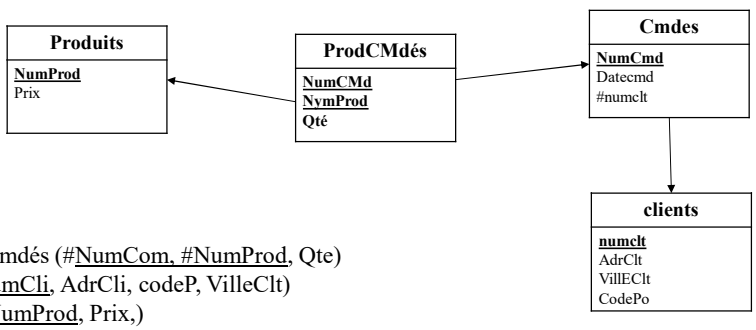
  b. En quelle forme normale elle est ?

  c. Mettez la relation en 3FN le cas échéant.

III. 2. Modèle logique de données : MLD

Exercice :

- Commandes (NumCom, DateCom, NumCli, AdrCli, NumProd, Prix, Qte)



Produits_Cmdés (#NumCom, #NumProd, Qte)
Clients (NumCli, AdrCli, codeP, VilleClt)
Produits (NumProd, Prix,)
Commandes (NumCom, DateCom, #NumCli)

Chapitre 4 Langage SQL (Structured Query Language)

IV. 1. Langage de définition de données (LDD)

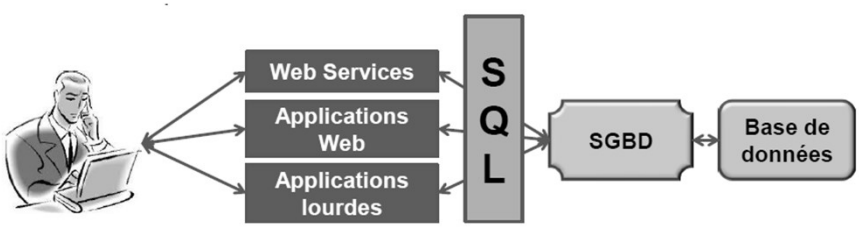
1. SGBD : Fonctionnalités et Principes :

- Un Système de Gestion de Base de données (SGBD) Pourquoi faire ?
  - Gestion du stockage des données
  - Traitements des données
    - Modification
    - Interrogation, Extraction

IV. 1. Langage de définition de données (LDD)

2. Introduction :

- **SQL:** Structured Query Language est un langage intuitif et complet sert à faire la gestion de bases de données relationnelles (définir, manipuler et récupérer des données).



IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.1. La création d'une base de données :

- Pour créer une base de données qui sera appelée « ma_base » il suffit d'utiliser la requête suivante qui est très simple :

```
CREATE DATABASE ma_base ;
```

- Avec MySQL, si une base de données porte déjà ce nom, la requête retournera une erreur.
- Pour éviter d'avoir cette erreur, il convient d'utiliser la requête suivante pour MySQL :

```
CREATE DATABASE IF NOT EXISTS  ma_base ;
```

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.2. La suppression d'une base de données :

- En SQL, la commande DROP DATABASE permet de supprimer totalement une base de données et tout ce qu'elle contient.
- Pour supprimer la base de données « ma_base » , la requête est la suivante :

```
DROP DATABASE ma_base ;
```

- Pour ne pas afficher d'erreur si la base n'existe pas:

```
DROP DATABASE IF EXISTS ma_base ;
```

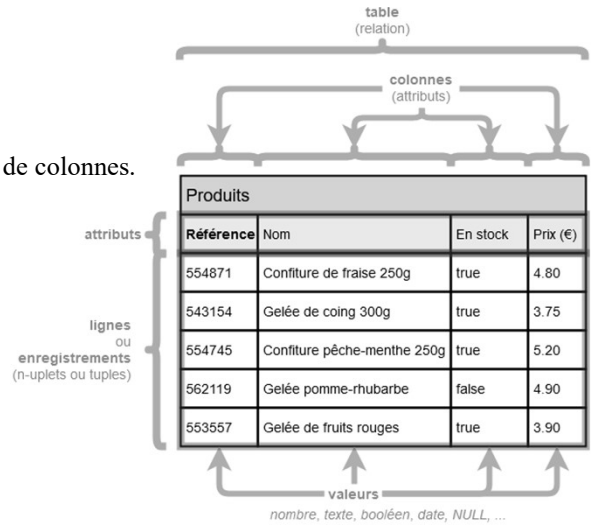
IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

- Qu'est ce qu'une TABLE?

Unité de stockage élémentaire, composée de lignes et de colonnes.



IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

❑ La création d'une table :

CREATE NOM_TABLE

Vous devez indiquer :

❑ Le nom de la table,

❑ Le nom, le type de données et la taille des colonnes.

```
CREATE TABLE nom_table
(
nom_colonne_1   type_colonne_1,
nom_colonne_2   type_colonne_2,
…
nom_colonne_n   type_colonne_n
)
```

Instruction SQL

```
CREATE TABLE bdsoutou.Compagnie
 (comp    CHAR(4),
  nrue    INTEGER(3),
  rue     CHAR(20),
  ville   CHAR(15) DEFAULT 'Paris'
          COMMENT 'Par defaut : Paris',
  nomComp CHAR(15) NOT NULL);
```

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

❑ La création d'une table : Types de données

Type de donnée	Syntaxe	Description
Type alphanumérique	VARCHAR(n)	Chaîne de caractères de n caractères maximum (n<16383)
Type numérique	NUMBER(n,[d])	Nombre de n chiffres [optionnellement *d* après la virgule]
Type numérique	FLOAT	Nombre à virgule flottante
Type horaire	DATE	Date sous la forme 16/07/99
Type horaire	TIME	Heure sous la forme 12:54:24.85
Type horaire	TIMESTAMP	Date et Heure

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

❑ Vérifiez la création de la table :

DESCRIBE NOM_TABLE // desc dept;

```
DESCRIBE dept;
```

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

❑ Modification d'une table :

ALTER TABLE NOM_TABLE

Permet :

- ❑ d'ajouter une nouvelle colonne,
- ❑ de modifier une colonne existante,
- ❑ de supprimer une colonne.

```
ALTER TABLE table
ADD             (column datatype [DEFAULT expr]
                [, column datatype]...);
```

```
ALTER TABLE table
MODIFY     (column datatype [DEFAULT expr]
                [, column datatype]...);
```

```
ALTER TABLE table
DROP          (column);
```

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

❑ Modification d'une table : Ajouter une colonne

- La clause **ADD** permet d'ajouter des colonnes.

```
ALTER TABLE dept80
ADD              (job_id VARCHAR(9));
Table altered.
```

- La nouvelle colonne est placée à la fin de la table. d'ajouter une nouvelle colonne,

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
149	Zlotkey	126000	29-JAN-00
174	Abel	132000	11-MAY-96
176	Taylor	103200	24-MAR-98

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

❑ Modification d'une table :

Modifier une colonne

- La clause **MODIFY** de modifier le type de données, la taille et la valeur par défaut d'une colonne.

```
ALTER TABLE   dept80
MODIFY              (last_name VARCHAR(30));
Table altered.
```

Supprimer une colonne

- La clause **DROP COLUMN** permet de supprimer d'une table les colonnes qui ne sont plus utiles.

```
ALTER TABLE  dept80
DROP COLUMN  job_id;
Table altered.
```

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

❑ Suppression d'une table :

DROP TABLE

- La structure et l'ensemble des données de la table sont supprimées.
- Vous ne pouvez pas annuler une instruction DROP TABLE.

```
DROP TABLE dept80;
Table dropped.
```

❑ Renommer une table :

RENAME

- Permet de renommer une table

```
RENAME dept TO detail_dept;
Table renamed.
```

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

❑ Vider une table :

TRUNCATE TABLE

- Supprime toutes les lignes d'une table,
- Libère l'espace de stockage utilisé par la table .

```
TRUNCATE TABLE detail_dept;
Table truncated.
```

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ Qu'est-ce qu'une contrainte ?

- Les contraintes appliquent des règles au niveau d'une table.
- Elles sont Utilisées pour empêcher l'insertion de données invalides dans la base
- Les types de contrainte suivants sont utilisés :
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ **Contraintes :**

- **`UNIQUE`** Indique que les valeurs dans la colonne doivent être toutes différentes
- **`NOT NULL`** Indique qu'on est obligé de mettre une valeur dans la colonne
- **`CHECK`** impose un domaine de valeurs ou une condition concernant la colonne.
  - **`CHECK (note BETWEEN 0 AND 20)`**
- **`PRIMARY KEY`** déclare la clé primaire de la table.
- **`FOREIGN KEY`** déclare une clé étrangère entre une table enfant (child) et une table mère (parent).

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ **Contraintes : La contrainte NOT NULL**

- La contrainte **`NOT NULL`** interdit la présence de valeurs **`NULL`** dans la colonne concernée.
- Par défaut, les colonnes qui ne sont pas associées à la contrainte **`NOT NULL`** peuvent contenir des valeurs **`NULL`**.

Exemple:

```
CREATE TABLE EMP(
  id_emp NUMBER(6),
  nom VARCHAR2(50) NOT NULL,
  prenom VARCHAR2(50) CONSTRAINT emp_prenom_nn NOT NULL
  …);
```

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ **Contraintes : La contrainte NOT NULL**

Exemple:

EMPLOYEE_ID	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
100	King	SKING	515.123.4567	13-JUN-87	AD_PRES	24000	80
101	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP		20
102	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	50
103	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG		60
104	Emst	BERNEST	590.423.4568	21-MAY-91	IT_PROG	6000	80

Contrainte **`NOT NULL`** (aucune ligne de cette colonne ne peut contenir de valeur NULL)

Absence de contrainte **`NOT NULL`** (les lignes de cette colonne peuvent contenir une valeur NULL)

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ **Contraintes : La contrainte UNIQUE**

- La contrainte d'intégrité **`UNIQUE`** exige que chaque valeur d'une colonne soit unique, c'est-à-dire qu'elle n'existe pas sur plusieurs lignes de cette colonne

Exemple:

```
CREATE TABLE EMP(
  id_emp NUMBER(6),
  email VARCHAR2(50) CONSTRAINT emp_email_u UNIQUE,
  …);
```

IV. 1. Langage de définition de données (LDD)

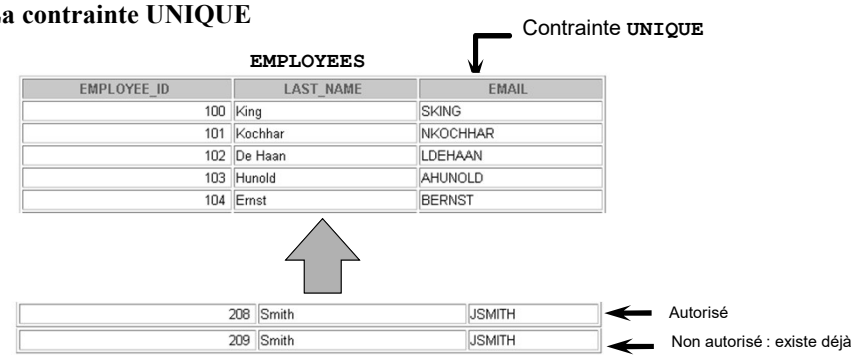
3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ **Contraintes : La contrainte UNIQUE**

Exemple:



IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ **Contraintes : La contrainte PRIMARY KEY**

- **`PRIMARY KEY`** permet de créer la clé primaire de la table.
- **`PRIMARY KEY`** = UNIQUE + NOT NULL

Exemple:

```
CREATE TABLE EMP(
  id_emp NUMBER(6) PRIMARY KEY,
  email VARCHAR2(50),
  …);
```

```
CREATE TABLE EMP(
  id_emp NUMBER(6),
  CONSTRAINT emp_email_pk PRIMARY KEY(id_emp),
  …);
```

IV. 1. Langage de définition de données (LDD)

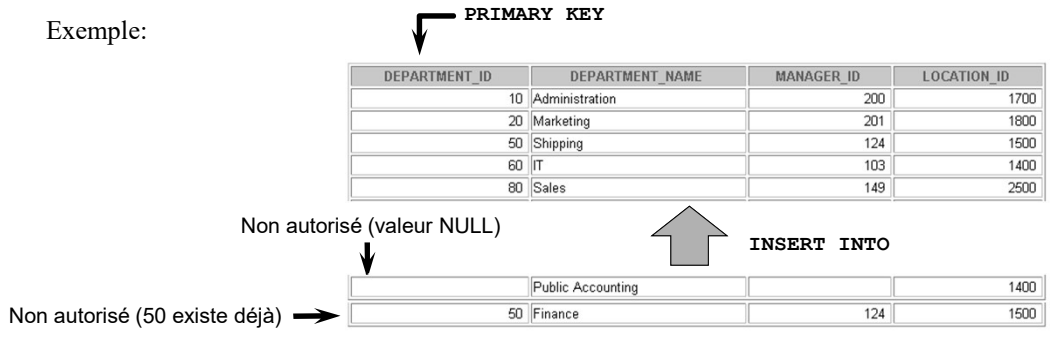
3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ **Contraintes : La contrainte PRIMARY KEY**

Exemple:



IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ **Contraintes : La contrainte FOREIGN KEY**

- **`FOREIGN KEY`** est une colonne qui fait référence à une colonne clé primaire d'une autre table.

Exemple:

```
CREATE TABLE EMP(
  id_emp NUMBER(6),
  nom VARCHAR2(50) NOT NULL,
  CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
      REFERENCES departments(department_id), …);
```

IV. 1. Langage de définition de données (LDD)

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ **Contraintes :**

La contrainte FOREIGN KEY

DEPARTEMENT

PRIMARY KEY →

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500

EMPLOYEE

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	
100	King	80	
101	Kochhar	20	
102	De Haan	50	
103	Hunold	60	
104	Emst	80	
107	Ford	90	← Non autorisé (90 n'existe pas)
200	Ford	60	← 60 est Autorisé

← FOREIGN KEY

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ **Contraintes : La contrainte CHECK**

- **`CHECK`** Définit une condition que chaque ligne doit satisfaire.

Exemple:

```
CREATE TABLE EMP(
  id_emp NUMBER(6),…
   commission NUMBER(4,2) CONSTRAINT emp_comm_check
CHECK(commission>=0),
  …);
```

Id_emp	commission	…
12	0,00	…
13	200,00	..
16	100,00	…

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ **Ajouter une contrainte :**

- Contrainte de clé primaire :

```
ALTER TABLE maTable
ADD CONSTRAINT PK_maTable
  PRIMARY KEY (colonnesConstituantLaCléPrimaire)
```

- Contraintes de clé étrangère :

```
ALTER TABLE maTable
ADD CONSTRAINT FK_maTable_colonneDeCleEtrangere
  FOREIGN KEY(colonneDeCleEtrangere)
  REFERENCES tableContenantLaClePrimaireAReferencer(colonneDeClePrimaire)
```

- Contrainte de domaine (CHECK)

```
ALTER TABLE maTable
ADD CONSTRAINT CHK_maTable_maColonne
CHECK(maColonne [formuleDeVerification] )
```

IV. 1. Langage de définition de données (LDD)

3. Créer et gérer une base de données :

3.3. Créer et gérer des tables :

3.3. 1. Inclure des contraintes :

❑ **Ajouter une contrainte :**

- Contrainte d'unicité :

```
ALTER TABLE maTable
ADD CONSTRAINT UQ_maTable_colonnesConstituantLeTupleQuiDoitEtreUnique
  UNIQUE (colonnesConstituantLeTupleQuiDoitEtreUnique)
```

- Contrainte de valeur par défaut :

```
ALTER TABLE maTable
MODIFY nom_colonne type DEFAULT 'valeur'
```

IV. 2. Langage de manipulation de données (LMD)

1. Insertion des lignes :

❑ Pour insérer des données dans une base, il y a 2 syntaxes principales :

- ❑ Insérer une ligne en indiquant les informations pour chaque colonne existante (en respectant l'ordre)
- ❑ Insérer une ligne en spécifiant les colonnes que vous souhaiter compléter. Il est possible d'insérer une ligne renseignant seulement une partie des colonnes

❑ La syntaxe pour remplir une ligne avec cette méthode est la suivante :

```
INSERT INTO table VALUES ('valeur 1', 'valeur 2', ...) ;
```

IV. 2. Langage de manipulation de données (LMD)

1. Insertion des lignes :

- Cette deuxième solution est très similaire, excepté qu'il faut indiquer le nom des colonnes avant « **VALUES** ».
- La syntaxe est la suivante :

```
INSERT INTO table (nom_colonne_1, nom_colonne_2, ...) VALUES ('valeur 1', 'valeur 2', ...);
```

Placez les valeurs de type caractère et date entre quottes .

IV. 2. Langage de manipulation de données (LMD)

1. Insertion des lignes :

- Il est possible d'ajouter plusieurs lignes à un tableau avec une seule requête. Pour ce faire, il convient d'utiliser la syntaxe suivante :

```
INSERT INTO client (prenom, nom, ville, age)
 VALUES
 ('Rébecca', 'Armand', 'Saint-Didier-des-Bois', 24),
 ('Aimée', 'Hebert', 'Marigny-le-Châtel', 36),
 ('Marielle', 'Ribeiro', 'Maillères', 27),
 ('Hilaire', 'Savary', 'Conie-Molitard', 58);
```

IV. 2. Langage de manipulation de données (LMD)

2. modifications sur des lignes :

- La syntaxe basique d'une requête utilisant UPDATE est la suivante :

```
UPDATE table
SET nom_colonne_1 = 'nouvelle valeur'
WHERE condition;
```

- Pour spécifier en une seule fois plusieurs modification, il faut séparer les attributions de valeur par des virgules. Ainsi la syntaxe deviendrait la suivante :

```
UPDATE table
SET colonne_1 = 'valeur 1', colonne_2 = 'valeur 2', colonne_3 =
'valeur 3'
WHERE condition ;
```

IV. 2. Langage de manipulation de données (LMD)

3. Supprimer des lignes dans une table :

- La syntaxe pour supprimer des lignes est la suivante :

```
DELETE FROM table
WHERE condition;
```

- Il est possible de supprimer une ligne en effectuant la requête SQL suivante :

```
DELETE FROM utilisateur
WHERE 'id' = 1;
```

IV. 2. Langage de manipulation de données (LMD)

3. Supprimer des lignes dans une table :

- Si l'ont souhaite supprimer les utilisateurs qui se sont inscrit avant le 10/04/2012, il va falloir effectuer la requête suivante :

```
DELETE FROM utilisateur
WHERE date_inscription < '2012-04-10';
```

- Pour supprimer toutes les lignes d'une table il convient d'utiliser la commande DELETE sans utiliser de clause conditionnelle.

```
DELETE FROM utilisateur;
```

IV. 3. Langage de Récupération des Données (LRD)

1. Récupération de données :

- **SELECT** récupère des enregistrements dans un tableau. Cette commande peut sélectionner une ou plusieurs colonnes d'une table.
- L'utilisation basique de cette commande s'effectue de la manière suivante :

```
SELECT nom_du_champ1, nom_du_champ2, … FROM nom_du_tableau;
```

- Pour obtenir les prénoms et les noms des clients il faut utiliser la requête suivante.

```
SELECT prenom, nom FROM client;
```

IV. 3. Langage de Récupération des Données (LRD)

1. Récupération de données :

- **Exemple :**

```
SELECT *
FROM     dept;
```

```
   DEPTNO DNAME          LOC
--------- -------------- -------------
       10 ACCOUNTING     NEW YORK
       20 RESEARCH       DALLAS
       30 SALES          CHICAGO
       40 OPERATIONS     BOSTON
```

```
SELECT deptno, loc
FROM   dept;
```

```
   DEPTNO LOC
--------- -------------
       10 NEW YORK
       20 DALLAS
       30 CHICAGO
       40 BOSTON
```

IV. 3. Langage de Récupération des Données (LRD)

1. Récupération de données :

- Pour éviter des redondances dans les résultats il faut simplement ajouter **`DISTINCT`** après le mot **`SELECT`**.
- L'utilisation basique de cette commande consiste alors à effectuer la requête suivante :

```
SELECT DISTINCT ma_colonne
FROM nom_du_tableau;
```

- Cette requête sélectionne le champ **« ma_colonne »** de la table **« nom_du_tableau »** en évitant de retourner des doublons.

IV. 3. Langage de Récupération des Données (LRD)

1. Récupération de données :

- **Exemple :**

```
SELECT *
FROM   dept;
```

```
   DEPTNO DNAME          LOC
--------- -------------- -------------
       10 ACCOUNTING     NEW YORK
       20 RESEARCH       DALLAS
       30 SALES          CHICAGO
       40 OPERATIONS     BOSTON
```

```
SELECT deptno
FROM   emp;
```

```
   DEPTNO
---------
       10
       30
       10
       20
...
14 rows selected.
```

```
SELECT DISTINCT deptno
FROM   emp;
```

```
    DEPTNO
 ---------
        10
        20
        30
```

IV. 3. Langage de Récupération des Données (LRD)

2. L'Alias de Colonne :

- En SQL, il est possible d'utiliser des alias. Il s'agit de renommer temporairement une colonne ou une table dans une requête.
- Cette fonctionnalité est très pratique pour faciliter la lecture des résultats d'une requête SQL.
- Au niveau des tables, l'intérêt d'utiliser des alias se vérifie lorsqu'il y a des jointures.
- Cela permet notamment d'obtenir des noms plus courts pour les tables.
- Permet de renommer un en - tête de colonne (champs)
- Doit obligatoirement être inclus entre guillemets (") s'il contient des espaces

IV. 3. Langage de Récupération des Données (LRD)

2. L'Alias de Colonne :

- On peut utiliser la commande **AS** ou non.
- Syntaxe de l'alias pour un champ :

```
SELECT champ1 AS nom_alias FROM table1 ;
```

- Syntaxe de l'alias pour une table :

```
SELECT champ1 FROM table1 AS nom_alias ;
```

- **Exemple :**

```
SELECT ename AS name, salary sal
FRM emp;
```

IV. 3. Langage de Récupération des Données (LRD)

Exercice (1) :

Soit la table Etudiant (CIN, Nom, Prénom, Age, Adresse, Tel).

1. Afficher tous les lignes de la relation Etudiant.
2. Afficher les noms, les prénoms et les âges de tous les étudiants.
3. Afficher les CIN des étudiants dans une colonne nommée Code
4. Afficher le nom et le prénom dans une seule colonne qui porte le nom Nom&Prénom

IV. 3. Langage de Récupération des Données (LRD)

3. Opérateurs arithmétiques :

- Utiliser pour effectuer les opérations sur les date et les nombre.

Opérateur	Description
*	Multiplier
/	Diviser
+	Additionner
-	Soustraire

IV. 3. Langage de Récupération des Données (LRD)

3. Opérateurs arithmétiques :

- Pour afficher le prix TTC d'un produit on peut utiliser la requête suivante:

```
SELECT nomP, prix + tax as « prix TTC »
FROM produit;
```

- Pour afficher le prix d'un produit après réduction de 20%, on utilise la requête :

```
SELECT nomP, (prix + tax)*0,8 as « prix TTC après reduction »
FROM produit;
```

IV. 3. Langage de Récupération des Données (LRD)

4. La commande WHERE :

- La commande **WHERE** s'utilise en complément à une requête utilisant **SELECT**. La façon la plus simple de l'utiliser est la suivante :

```
SELECT nom_colonnes
FROM nom_table WHERE condition;
```

- Pour obtenir seulement la liste des clients qui habitent à Paris, il faut effectuer la requête suivante :

```
SELECT * FROM client WHERE ville = 'paris';
```

IV. 3. Langage de Récupération des Données (LRD)

5. Opérateurs de comparaisons :

- La liste ci-jointe présente quelques uns des opérateurs les plus couramment utilisés.

Opérateur	Description
=	Égale
<>	Pas égale
!=	Pas égale
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égale à
<=	Inférieur ou égale à
IN	Liste de plusieurs valeurs possibles
BETWEEN	Valeur comprise dans un intervalle donnée (utile pour les nombres ou dates)
LIKE	Recherche en spécifiant le début, milieu ou fin d'un mot.
IS NULL	Valeur est nulle
IS NOT NULL	Valeur n'est pas nulle

IV. 3. Langage de Récupération des Données (LRD)

5. Opérateurs de comparaisons :

- Pour chercher toutes les lignes où la colonne « **nom_colonne** » est égale à 'valeur 1' OU 'valeur 2' ou 'valeur 3', il est possible d'utiliser la syntaxe suivante:

```
SELECT nom_colonne
FROM table
WHERE nom_colonne IN ( valeur1, valeur2, valeur3, ... );
```

- L'utilisation de la commande BETWEEN s'effectue de la manière suivante :

```
SELECT *
FROM table
WHERE nom_colonne BETWEEN 'valeur1' AND 'valeur2';
```

IV. 3. Langage de Récupération des Données (LRD)

5. Opérateurs de comparaisons :

- La syntaxe à utiliser pour utiliser l'opérateur LIKE est la suivante :

```
SELECT *
FROM table
WHERE colonne LIKE modele;
```

- L'expression contient les symboles :

Symbole	Description
%	Représente une série de zéros ou de caractères
_	Représente un caractère quelconque

IV. 3. Langage de Récupération des Données (LRD)

5. Opérateurs de comparaisons :

- Si l'ont souhaite obtenir uniquement les clients des villes qui commencent par un « N », il est possible d'utiliser la requête suivante :

```
SELECT *
FROM client
WHERE ville LIKE 'N%';
```

- Pour obtenir les clients des villes qui terminer par « e », on utilise :

```
SELECT *
FROM client
WHERE ville LIKE '%e';
```

IV. 3. Langage de Récupération des Données (LRD)

5. Opérateurs de comparaisons :

- Pour filtrer les résultats où les champs d'une colonne à **NULL** il convient d'utiliser la syntaxe suivante :

```
SELECT *
FROM table
WHERE nom_colonne IS NULL;
```

- A l'inverse, pour filtrer les résultats et obtenir uniquement les enregistrements qui ne sont pas nul, il convient d'utiliser la syntaxe suivante :

```
SELECT *
FROM table
WHERE nom_colonne IS NOT NULL;
```

IV. 3. Langage de Récupération des Données (LRD)

5. Opérateurs de comparaisons :

- Autre opérateurs :

```
... WHERE ...   NOT  BETWEEN ... AND ...
... WHERE ...   IS  NOT  IN ...
... WHERE ...   NOT LIKE ...
... WHERE ...   IS  NOT  NULL
```

```
SELECT ename, job
FROM   emp
WHERE  job NOT IN ('CLERK','MANAGER','ANALYST');
```

```
ENAME      JOB
---------- ---------
KING       PRESIDENT
MARTIN     SALESMAN
ALLEN      SALESMAN
TURNER     SALESMAN
WARD       SALESMAN
```

IV. 3. Langage de Récupération des Données (LRD)

Exercice (2) :

- Soit la table Employé (Id, nom, prénom, salaire)
  1. Afficher les noms, prénoms et salaires+500 dans une colonne « salaire augmenté » de l'ensemble des employés
  2. Afficher les noms, prénoms des employés qui ont le salaire entre 3000 et 5000

- Soit la table Etudiant (Id, nom, prenom, age, ville, tel).
  1. Quels sont les étudiants âgés de 20 ans ou plus ?
  2. Quels sont les étudiants âgés de 19 à 23 ans ?
  3. Quels sont les étudiants ayant un numéro de tel commençant par 0535?
  4. Quels sont tous les étudiants dont la ville est inconnue/connue ?

IV. 3. Langage de Récupération des Données (LRD)

6. Opérateurs de concaténation :

- La syntaxe à utiliser pour utiliser les opérateurs de concaténation est la suivante :

```
SELECT concat(Attribut_1,Attribut_2, …) FROM table;
```

- **Exemple :**

```
SELECT concat(ename,' is a ', job) AS « Employee Details »
FROM emp
```

```
Employee Details
-------------------------
KING is a PRESIDENT
BLAKE is a MANAGER
CLARK is a MANAGER
JONES is a MANAGER
MARTIN is a SALESMAN
...
14 rows selected.
```

IV. 3. Langage de Récupération des Données (LRD)

Exercice (3) :

- Soit la table Etudiant (CNE, nom, prenom, age, ville). Ecrire les requêtes SQL qui permet de :
  - Afficher les noms et prénoms des étudiants dans une seule colonne et nommer la « nom_prenom »
  - Afficher les villes des étudiants sans doublon.
  - Afficher les étudiants des villes qui ne commence pas par « T »
  - Afficher les étudiants qui ont un prénom soit « Ahmed » soit « Said » soit « karim »
  - Afficher les étudiants qui ont des villes inconnu
  - Afficher la liste de tous les villes des étudiants sans répétition .

IV. 3. Langage de Récupération des Données (LRD)

7. Ajout d'enregistrement dans une table à partir d'une selection :

- On peut aussi insérer des ligne à partir d'une sélection en utilisant le syntaxe suivant :

Syntaxe 2:

```
INSERT INTO     Nomtable [(champs [, champs ...])]
                (select …);
```

- Les champs de la clause INSERT doit correspondre à ceux de la sous-interrogation :

Exemple:

```
INSERT INTO SALSES (mat, nom, sal, hiredate, deptno)
        SELECT   empno, ename, hiredate,  sal+ comm, deptno
        FROM   emp
        WHERE  job like 'SALSES%';
```

IV. 3. Langage de Récupération des Données (LRD)

6. Ordre SELECT (sélection) :

Ordonner la clause SELECT

- Afficher des enregistrements sélectionnés dans l'ordre croissant ou décroissant.
- Syntaxe:

```
SELECT * FROM table WHERE condition(s) ORDER BY {colonne | expression}
[ASC | DESC];
```

- **Exemple :**

```
SELECT nom,prenom,age FROM etudiant ORDER BY age;
```

L'ordre par défaut est croissant.

IV. 3. Langage de Récupération des Données (LRD)

6. Ordre SELECT (sélection) :

Ordonner la clause SELECT

- Trie sur plusieurs colonnes.
- Syntaxe:

```
SELECT col1, col2, col3 FROM table WHERE condition(s) ORDER BY col1[ASC
|DESC], col3[ASC | DESC];
```

- **Exemple :**

```
SELECT nom, prénom, age FROM etudiant ORDER BY prénom, age DESC;
```

IV. 3. Langage de Récupération des Données (LRD)

7. Fonctions de chaînes de caractères :

Fonction	Définition	Exemples	Résultat
INITCAP(col | expr)	Convertit la première lettre de chaque mot d'une chaîne de caractères en majuscule et les autres lettres en minuscule.	**INITCAP**('Cours de SQL')	Cours De Sql
LOWER(col | expr)	Convertit une chaîne de caractères en minuscule.	**LOWER**('Cours de SQL')	cours de sql
UPPER(col | expr)	Convertit une chaîne de caractères en majuscule.	**UPPER** ('Cours de SQL')	COURS DE SQL
LENGHT(col | expr)	Permet de récupérer le nombre de caractères d'une chaîne. LENGTH retourne une valeur de type NUMBER.	**LENGTH** ('Bonjour')	7
SUBSTR(col | expr, m[,n])	Permet d'extraire une chaîne de caractères de la chaîne de caractère col (ou issue de expr) sur une longueur n à partir de la position m.	**SUBSTR**('Bonjour',1,3)	Bon
CONCAT(col1 | expr1, col2 | expr2)	Permet de concaténer la valeur de la première chaîne à la valeur de la seconde chaîne. (équivalent à l'opérateur "||").	**CONCAT** ('Bon','jour')	Bonjour

IV. 3. Langage de Récupération des Données (LRD)

8. Fonctions de groupe :

Fonction	Description
SUM([**DISTINCT** | **ALL**] **n**)	Retourne la somme de toutes les valeurs du groupe **n**
MIN([**DISTINCT** | **ALL**] **expr**)	Retourne la plus petite valeur du groupe **expr**
MAX([**DISTINCT** | **ALL**] **expr**)	Retourne la plus grande valeur du groupe **expr**
COUNT({ * | [**DISTINCT** | **ALL**] **expr** })	-Retourne le nombre d'enregistrements contenus dans le groupe **expr.**
AVG([**DISTINCT** | **ALL**] **n**)	Retourne la moyenne des valeurs du groupe **n**
VARIANCE([**DISTINCT** | **ALL**] **x**)	Retourne la variance de **x**

IV. 3. Langage de Récupération des Données (LRD)

8. Fonctions de groupe :

Remarques :

- Les fonctions de groupe ignorent les valeurs nulles sauf la fonction **`COUNT(*)`**.
- Le type de données des arguments peuvent être **`CHAR, VARCHAR2, NUMBER, DATE`** sauf pour les fonctions **`AVG, SUM, VARIANCE`** qui ne peuvent être utilisées qu'avec des données de type numérique.
- Pour substituer les valeurs nulles dans un groupe, il faut utiliser la fonction « **`IFNULL`** »

IV. 3. Langage de Récupération des Données (LRD)

8. Fonctions de groupe :

Utilisation des fonctions de regroupement :

```
SELECT AVG(sal), MAX(sal), MIN(sal), SUM(sal)
FROM emp
WHERE         job LIKE 'SALES%';
```

```
AVG(SAL)  MAX(SAL)  MIN(SAL)  SUM(SAL)
--------- --------- --------- ---------
    1400      1600      1250      5600
```

```
SELECT        COUNT(*)
FROM emp
WHERE         deptno = 30;
```

```
 COUNT(*)
---------
         6
```

IV. 3. Langage de Récupération des Données (LRD)

8. Fonctions de groupe :

Utilisation des fonctions de regroupement :

```
SELECT AVG(comm)
FROM   emp;
```

```
 AVG(COMM)
---------
      550
```

```
AVG(IFNULL(COMM,0))
FROM   emp;
```

```
AVG(IFNULL(COMM,0))
----------------
       157.14286
```

IV. 3. Langage de Récupération des Données (LRD)

Exercice (4) :

- Ecrire la requête qui retourne la moyenne et la somme des notes des étudiants dont le nom commence 'M'.

IV. 3. Langage de Récupération des Données (LRD)

Exercice (4) :

- Ecrire la requête qui retourne la moyenne et la somme des notes des étudiants dont le nom commence 'M'.

```
SELECT AVG(note), SUM(note)
FROM Etudiant
WHERE nom LIKE 'M%';
```

IV. 3. Langage de Récupération des Données (LRD)

Exercice (5) :

- Ecrire la requête qui retourne la date d'embauche la plus ancienne et la plus récente.

IV. 3. Langage de Récupération des Données (LRD)

Exercice (5) :

- Ecrire la requête qui retourne la date d'embauche la plus ancienne et la plus récente.

```
SELECT MIN(date_emb), MAX(date_emb)
FROM employe;
```

IV. 3. Langage de Récupération des Données (LRD)

Exercice (6) :

- Q1: Ecrire la requête qui renvoie le nombre de départements (doublons inclus) de la table employe.
- Q2: Ecrire la requête qui retourne le nombre d'enregistrements dans la table EMP dont la colonne N_dep a pour valeur 30.

IV. 3. Langage de Récupération des Données (LRD)

Exercice (6) :

- Q1: Ecrire la requête qui renvoie le nombre de départements (doublons inclus) de la table employe.
- Q2: Ecrire la requête qui retourne le nombre d'enregistrements dans la table EMP dont la colonne N_dep pour la valeur 30.

```
Q1 :
SELECT COUNT(N_dep) FROM employe;

Q2 :
SELECT COUNT(*) FROM employe
WHERE N_dep = 30;
```

IV. 3. Langage de Récupération des Données (LRD)

Exercice (7) :

- Ecrire la requête qui renvoie la moyenne des commissions obtenues par les employés.

```
SELECT AVG(comm) FROM employe;
```

Le calcul de la moyenne ne tient pas compte des valeurs invalides telles que les valeurs nulles.

```
SELECT AVG(ifnull(comm,0))
FROM employe;
```

IV. 3. Langage de Récupération des Données (LRD)

9. Création de Groupes de Données :

Clause `GROUP BY`

- Permet de diviser les enregistrements d'une table en groupes.
- Les fonctions de groupe peuvent être utilisées pour retourner les informations relatives à chaque groupe.
- Syntaxe:

```
SELECT [colonne,] fonction_groupe(argument)
FROM table
[ WHERE condition(s) ]
[ GROUP BY colonne]
[ORDER BY fontion_group(argument)];
```

IV. 3. Langage de Récupération des Données (LRD)

9. Création de Groupes de Données :

Clause `GROUP BY`

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

2916.6667

2175

1566.6667

"salaire moyen pour chaque département de la table EMP"

DEPTNO	AVG(SAL)
10	2916.6667
20	2175
30	1566.6667

```
SELECT   deptno, AVG(sal)
FROM     emp
GROUP BY deptno;
```

IV. 3. Langage de Récupération des Données (LRD)

9. Création de Groupes de Données :

Remarques:

- L'argument peut-être un nom de colonne, une expression ou une constante.
- Les fonctions de groupe ne peuvent pas être utilisées dans les clauses **`FROM`**, **`WHERE`** et **`GROUP BY`**

IV. 3. Langage de Récupération des Données (LRD)

9. Création de Groupes de Données :

Exemple de création de groupes de données :

```
SELECT   deptno, job, sum(sal)
FROM     emp
GROUP BY deptno, job;
```

```
   DEPTNO JOB         SUM(SAL)
--------- --------- ---------
       10 CLERK          1300
       10 MANAGER        2450
       10 PRESIDENT      5000
       20 ANALYST        6000
       20 CLERK          1900
...
9 rows selected.
```

IV. 3. Langage de Récupération des Données (LRD)

Exercice (8) :

- Ecrire la requête qui renvoie la moyenne des salaires pour chaque département présent dans la table employé.

IV. 3. Langage de Récupération des Données (LRD)

Exercice (8) :

- Ecrire la requête qui renvoie la moyenne des salaires pour chaque département présent dans la table employé.

```
SELECT deptno, AVG(sal) FROM employe
GROUP BY deptno;
```

IV. 3. Langage de Récupération des Données (LRD)

Exercice (9) :

- Ecrire la requête qui renvoie la somme des salaires de la table employé pour chaque fonction, groupé par département.

Deptno	Fonction	SUM(sal)
10	Ingénieur	40000
10	Technicien	20000
20	Ingénieur	36500
20	Technicien	10000
30	Ingénieur	76500
30	Technicien	15600

IV. 3. Langage de Récupération des Données (LRD)

Exercice (9) :

- Ecrire la requête qui renvoie la somme des salaires de la table employé pour chaque fonction, groupé par département.

```
SELECT deptno, fonction, SUM(sal)
FROM emp
GROUP BY deptno, fonction;
```

Deptno	Fonction	SUM(sal)
10	Ingénieur	40000
10	Technicien	20000
20	Ingénieur	36500
20	Technicien	10000
30	Ingénieur	76500
30	Technicien	15600

IV. 3. Langage de Récupération des Données (LRD)

10. Utilisation de la clause HAVING :

- La clause WHERE n'acceptant pas les fonctions de groupe.
- la restriction du résultat des fonctions de groupe se fera dans la clause HAVING
- Syntaxe:

```
SELECT [colonne,] fonction_groupe(argument)
FROM table
[WHERE condition(s) ]
[GROUP BY colonne]
[HAVING condition_condition]
[ORDER BY fontion_group(argument)];
```

IV. 3. Langage de Récupération des Données (LRD)

10. Utilisation de la clause HAVING :

- Pour chaque département afficher son salaire maximum si il est supérieur à 2900

IV. 3. Langage de Récupération des Données (LRD)

10. Utilisation de la clause HAVING :

- Pour chaque département afficher son salaire maximum si il est supérieur à 2900

```
SELECT   deptno, max(sal)
FROM     emp
GROUP BY deptno
HAVING   max(sal)>2900;
```

```
   DEPTNO  MAX(SAL)
--------- ---------
       10      5000
       20      3000
```

IV. 3. Langage de Récupération des Données (LRD)

10. Utilisation de la clause HAVING :

- Afficher pour chaque département qui est différente à département sales sa somme de salaire a condition que la somme ne doit pas dépasser 5000
- Ordonner l'affichage des sommes ascendant.

IV. 3. Langage de Récupération des Données (LRD)

10. Utilisation de la clause HAVING :

- Afficher pour chaque département qui est différente à département sales sa somme de salaire a condition que la somme ne doit pas dépasser 5000
- Ordonner l'affichage des sommes ascendant.

```
SELECT    job, SUM(sal) PAYROLL
FROM      emp
WHERE           job NOT LIKE 'SALES%'
GROUP BY  job
HAVING    SUM(sal)>5000
ORDER BY  SUM(sal);
```

```
JOB         PAYROLL
--------- ---------
ANALYST        6000
MANAGER        8275
```

IV. 3. Langage de Récupération des Données (LRD)

10. Utilisation de la clause HAVING :

La différence entre HAVING et WHERE :

- **`WHERE`** restreint les enregistrements
- **`HAVING`** restreint les groupes d'enregistrements

IV. 3. Langage de Récupération des Données (LRD)

Exercice (10) :

- Ecrire la requête qui renvoie les numéro de départements dont le salaire maximal (à afficher aussi) est supérieur à 2900.

IV. 3. Langage de Récupération des Données (LRD)

Exercice (10) :

- Ecrire la requête qui renvoie les numéro de départements dont le salaire maximal (à afficher aussi) est supérieur à 2900.

```
SELECT deptno, MAX(sal)
FROM employe
GROUP BY deptno
HAVING MAX(sal) > 2900;
```

IV. 3. Langage de Récupération des Données (LRD)

Exercice (11) :

- Afficher la somme des salaires des employés supérieure à 5000 par fonction, et les fonctions dont les quatre premières lettres sont différentes de la chaîne de caractères « TECH». Le résultat est ordonné de façon décroissante sur les sommes des salaires.

IV. 3. Langage de Récupération des Données (LRD)

Exercice (11) :

- Afficher la somme des salaires des employés supérieure à 5000 par fonction, et les fonctions dont les quatre premières lettres sont différentes de la chaîne de caractères « TECH». Le résultat est ordonné de façon décroissante sur les sommes des salaires.

```
SELECT fonction, SUM(sal)
FROM employe
WHERE fonction NOT LIKE 'TECH%'
GROUP BY fonction
HAVING SUM(sal) > 5000
ORDER BY SUM(sal) DESC;
```

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

- La sous-interrogation (requête interne) est exécutée une fois avant la requête principale (une sous-interrogation ne doit pas contenir la clause ORDER BY).
- Le résultat de la sous-interrogation est utilisé par la requête principale (externe).
- Syntaxe:

```
SELECT select_list FROM tables WHERE expr operator
      (SELECT select_list FROM tables… );
```

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Sous-interrogation ramenant une seule valeur.

- Quels sont les employés ayant la même fonction que youness ?

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Sous-interrogation ramenant une seule valeur.

- Quels sont les employés ayant la même fonction que codd ?

```
SELECT nom
FROM emp
WHERE fonction = (SELECT fonction
                  FROM emp
                  WHERE nom ='CODD');
```

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Sous-interrogation ramenant une seule valeur.

Remarques :

- Une sous-interrogation qui ne ramène aucune ligne se termine avec un code d'erreur.
- Une sous-interrogation ramenant plusieurs lignes provoquera aussi, dans ce cas, une erreur

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Exemple de Sous-Interrogations mono-ligne :

- Afficher les noms des employés et leurs job de tout les employés qui ont le même job de Mr. ADAMS et qui ont le salaire supérieur de son salaire

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Exemple de Sous-Interrogations mono-ligne :

- Afficher les noms des employés et leurs job de tout les employés qui ont le même job de Mr. ADAMS et qui ont le salaire supérieur de son salaire

```
SELECT   ename, job
FROM     emp
WHERE    job =                          CLERK
         (SELECT     job
          FROM       emp
          WHERE      ename = 'ADAMS')
AND      sal >                          1100
         (SELECT     sal
          FROM       emp
          WHERE      ename = 'ADAMS');
```

```
ENAME      JOB
---------- ---------
MILLER     CLERK
```

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Sous-interrogation ramenant plusieurs lignes.

- Une sous-interrogation peut ramener plusieurs lignes à condition que l'opérateur de comparaison admette à sa droite un ensemble de valeurs. Les opérateurs permettant de comparer une valeur à un ensemble de valeurs sont :
  - L'opérateur **`IN`**
  - Les opérateurs obtenus en ajoutant **`ANY`** ou **`ALL`** à la suite d'un opérateur de comparaison classique **`(=, !=, >, >=, <, <=)`**

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Sous-interrogation ramenant plusieurs lignes.

- les opérateurs obtenus en ajoutant ANY ou ALL à la suite d'un opérateur de comparaison classique (=, !=, >, >=, <, <=) :
  - **ANY**: la comparaison est vraie si elle est vraie pour au moins un des éléments de l'ensemble.
  - **< ANY**: signifie inférieur à au moins une des valeurs; donc inférieur au maximum.
  - **>ANY** : signifie supérieur à au moins une des valeurs; donc supérieur au minimum.
  - **=ANY** équivalent à **IN**
  - **ALL**: la comparaison sera vraie si elle est vraie pour tous les éléments de l'ensemble.
  - **>All** : signifie supérieur à tous.

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Exemple de Sous-Interrogations multiligne :

- Afficher les noms des employés et leurs jobs qui doivent être différent au job CLERK et qui ont le salaire inferieur au-moins d'un employé parmi les employé du job CLERK.

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Exemple de Sous-Interrogations multiligne :

- Afficher les noms des employés et leurs jobs qui doivent être différent au job CLERK et qui ont le salaire inferieur au-moins d'un employé parmi les employé du job CLERK.

```
SELECT  empno, ename, job
FROM    emp
WHERE   sal < ANY
           (SELECT sal
            FROM   emp
            WHERE  job = 'CLERK')
AND         job <> ''CLERK';
```

1300
1100
800
950

```
    EMPNO ENAME      JOB
--------- ---------- ---------
     7654 MARTIN     SALESMAN
     7521 WARD       SALESMAN
```

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Sous-Interrogation mono-ligne

Opérateurs mono-ligne

Opérateur	Signification
=	Égal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
<>	Différent de

Sous-Interrogation multiligne

Opérateurs multi-ligne

Opérateur	Signification
IN	Égal à un élément quelconque de la liste
ANY	Compare la valeur à chaque valeur ramenée par la sous-interrogation
ALL	Compare la valeur à toutes les valeurs ramenées par la sous-interrogation

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Exemple :

- Quels sont les employés gagnant plus que tous les employés du département 30.

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Exemple :

- Quels sont les employés gagnant plus que tous les employés du département 30.

```
SELECT nom, salaire
FROM emp
WHERE salaire > ALL (SELECT salaire
                     FROM emp
                     WHERE n_dept = 30);
```

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Sous-interrogation ramenant plusieurs colonnes :

- Il est possible de comparer le résultat d'un **`SELECT`** ramenant plusieurs colonnes à une liste de colonnes.
- La liste de colonnes figurera entre parenthèses à gauche de l'opérateur de comparaison.

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Exemple :

- Quels sont les employés ayant même fonction et même manager que CODD ?

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Exemple :

- Quels sont les employés ayant même fonction et même manager que CODD ?

```
SELECT nom, fonction, n_sup
FROM emp
WHERE (fonction, n_sup) = (SELECT fonction,n_sup
                           FROM emp
                          WHERE nom = 'CODD');
```

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Sous-interrogation ramenant plusieurs colonnes :

- Un **SELECT** peut comporter plusieurs sous-interrogations, soit imbriquées, soit au même niveau dans différents prédicats combinés par des **AND** ou des **OR**.

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Exemple :

- Liste des employés du département 10 ayant même fonction que quelqu'un du département de DUPONT.

IV. 3. Langage de Récupération des Données (LRD)

11. Les Sous-Interrogations :

Exemple :

- Liste des employés du département 10 ayant même fonction que quelqu'un du département de Achraf.

```
SELECT nom, fonction
FROM emp
WHERE n_dept = 10
AND fonction IN (SELECT fonction
                 FROM emp
                 WHERE n_dept = (SELECT n_dept
                                 FROM emp
                                 WHERE nom = 'Achraf'));
```

IV. 3. Langage de Récupération des Données (LRD)

12. Jointures de tables :

- Les jointures en SQL permettent d'associer plusieurs tables dans une même requête. Cela permet d'exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables en même temps de manière efficace.
- Il y a trois types des jointures :
  - **Equi-Jointure**
  - **Non Equi-Jointure**
  - **Auto-Jointure**

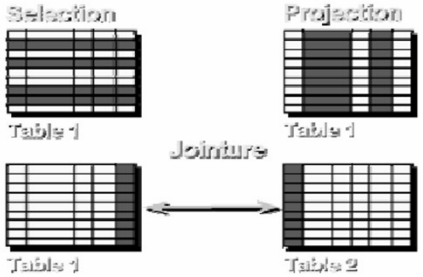
IV. 3. Langage de Récupération des Données (LRD)

12. Jointures de tables :

Ordre SELECT :

L'ordre SELECT possède trois capacités :

- Sélection : Sélection d'une ou plusieurs ligne(s).
- Projection : Sélection d'une ou plusieurs colonne(s).
- Jointure : Sélection de deux colonnes dans deux tables différentes, créant ainsi une relation entre les données des deux colonnes.



IV. 3. Langage de Récupération des Données (LRD)

12. Jointures de tables :

Equi-jointure :

- Utilisée pour afficher des données provenant de plusieurs tables lorsqu'on a un **champs en commun** entre ces tables.
- Une **équi-jointure** utilise l'opérateur d'égalité dans la clause de jointure et compare généralement des **clés primaires** avec des **clés étrangères**.

Exemple:

Id_etudiant	Nom	Id_formation
E1	Saadi	F1
E2	Mojahid	F1
E3	Salhi	F2

Id_formation	intitulé
F1	BD
F2	Math
F3	Réseau

Id_etudiant	Id_formation	intitulé
E1	F1	BD
E2	F1	BD
E3	F2	Math

IV. 3. Langage de Récupération des Données (LRD)

12. Jointures de tables :

Equi-jointure :

- **Exemple :** Afficher l'identité des pilotes et le nom de sa compagnie.

Compagnie

comp	nrue	rue	ville	nomComp
AF	124	Port Royal	Paris	Air France
SING	7	Camparols	Singapour	Singapore AL
CAST	1	G. Brassens	Blagnac	Castanet AL

Pilote

brevet	nom	nbHVol	compa	chefPil
PL-1	Pierre Lamothe	450	AF	PL-4
PL-2	Didier Linxe	900	AF	PL-4
PL-3	Christian Soutou	1000	SING	
PL-4	Henri Alquié	3400	AF	

IV. 3. Langage de Récupération des Données (LRD)

12. Jointures de tables :

Equi-jointure :

- **Exemple :** Afficher l'identité des pilotes et le nom de sa compagnie.

```
SELECT Pilote.brevet, Compagnie.nomComp
From Pilote, Compagnie
Where Compagnie.comp= Pilote.comp;
```

```
SELECT P.brevet, C.nomComp
From Pilote  P, Compagnie C
Where  C.comp=P.comp;
```

Compagnie

comp	nrue	rue	ville	nomComp
AF	124	Port Royal	Paris	Air France
SING	7	Camparols	Singapour	Singapore AL
CAST	1	G. Brassens	Blagnac	Castanet AL

Pilote

brevet	nom	nbHVol	compa	chefPil
PL-1	Pierre Lamothe	450	AF	PL-4
PL-2	Didier Linxe	900	AF	PL-4
PL-3	Christian Soutou	1000	SING	
PL-4	Henri Alquié	3400	AF	

IV. 3. Langage de Récupération des Données (LRD)

12. Jointures de tables :

Non équi-jointure :

- Une condition de non équi-jointure est utilisée lorsque deux tables n'ont pas de colonnes en commun mais elles ont des colonnes qui ont une relation entre eux.

Employé

Id_employé	Nom	salaire
E1	Rachdi	3500
E2	Maaroufi	6000
E3	Jadid	5500

Grade

id_grade	Salaire_inf	Salaire_sup
1	3000	4000
2	4100	5000
3	5100	6000

IV. 3. Langage de Récupération des Données (LRD)

12. Jointures de tables :

Non équi-jointure :

```
SELECT e.nom, g.id_grade
FROM employé e, grade g
WHERE e.salaire BETWEEN g.salaire_inf AND g.salaire_sup;
```

IV. 3. Langage de Récupération des Données (LRD)

12. Jointures de tables :

Auto-Jointure :

- Une condition d'auto-jointure permet de faire une jointure sur deux colonnes liées appartenant à la même table.
- Est un cas particulier de l'équi-jointure qui met en œuvre deux fois la même table.

• Equi-Jointure d'une table à elle-même :
  • Pourquoi ? : pour rassembler des informations venant de 2 lignes différentes
  • Exemple : noms des employés et noms de leurs managers

```
SELECT emp.nom, mgr.nom
FROM emp, emp mgr
WHERE emp.n_sup = mgr.num
```

Jointure de la table « emp » à elle-même

Besoin de renommer la table pour indiquer de quelle ligne vient la colonne citée
Ex : (emp.n_sup = mgr.num)

IV. 3. Langage de Récupération des Données (LRD)

12. Jointures de tables :

Auto-Jointure :

```
SELECT e.nom as 'nom de employé', manager.nom as 'nom du manager'
FROM emp e, emp manager
WHERE e.mgr = manager.id_emp;
```