

Bases de données

1

ENSAM-Meknès

A.AHMADI

2021/2022

Partie 1 : Conception d'une base de données

2

ENSAM-Meknès

A.AHMADI

2021/2022

Chap 1

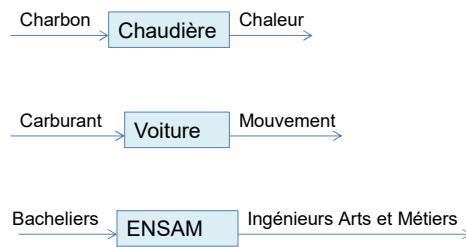
Introduction générale

I- Définitions

1-Système

Un ensemble d'éléments matériels et immatériels (hommes, machines, méthodes, règles, programmes, etc.) en interaction qui transforme, par un processus des éléments d'entrée (Input) en d'autres éléments de sortie (Output).

Exemples :



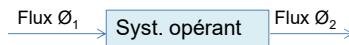
Chap 1

Introduction générale

I- Définitions

2- Système opérant

- Le système opérant est l'ensemble des moyens humains, matériels, organisationnels qui exécutent les ordres du système de pilotage.
- C'est un système transformant un flux physique en un autre flux physique. Il englobe toutes les fonctions liées à l'activité propre de l'organisme.



3- Système de pilotage

- Chaque système opérant est géré par un système de pilotage (de commande). Il décide des actions à **conduire** sur le système opérant en fonction des objectifs et des politiques de l'entreprise.

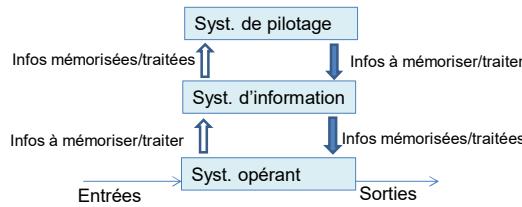
2- Système d'information (SI)

C'est une interface entre le système opérant et le système de pilotage. Il permet de stocker, d'extraire et de traiter les informations du *système opérant* pour les mettre à la disposition du *système de pilotage*.

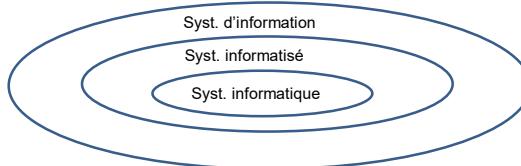
Chap 1

Introduction générale

I- Définitions



- Le **SI** est la mémoire de l'organisation. Il contient 2 aspects : **statique** (Données) et **dynamique** (Traitements).



Chap 1

Introduction générale

II- Méthode MERISE

- C'est une méthode de conception et de développement des systèmes d'information ;
- Elle a été conçue entre 1978 et 1979 par le CTI (Centre Technique d'Informatique), et le CETE (Centre d'Etudes Techniques de l'Equipement), en France.
- Elle procède par 3 niveaux d'abstraction :

Niveau	Données	Traitements
Conceptuel	Modèle Conceptuel de Données	Modèle Conceptuel de Traitements
Logique	Modèle Logique de Données	Modèle Organisationnel de Traitements
Physique	Modèle Physique de Données	Modèle Opérationnel de Traitements

Chap 1 Introduction générale

II- Méthode MERISE

1- Niveau conceptuel

On fait abstraction de tout concept lié à l'organisation tant de point de vue Données que celui des traitements.

QUOI ? FAIRE QUOI ? AVEC QUELLES DONNEES ?

2- Niveau organisationnel

On intègre à l'analyse les concepts liés à l'organisation (Notion de lieu, de temps d'action, etc).

QUI ? OÙ ? QUAND ?

3- Niveau opérationnel

Consiste à apporter des solutions techniques au problème (choisir les structures de données, les méthodes de stockage, découpage en programmes, etc.).

COMMENT ?

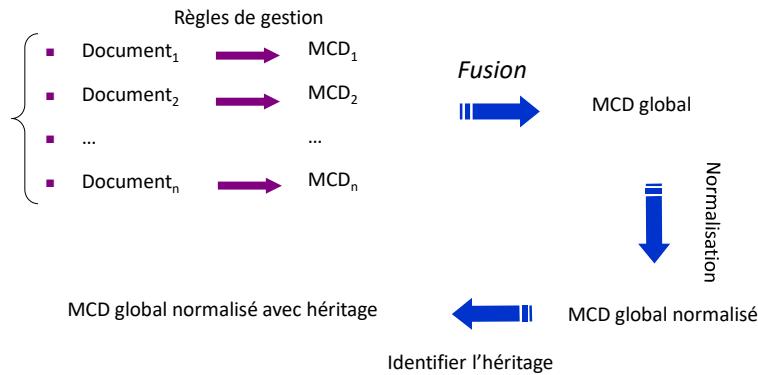
Chap 1 Introduction générale

III- Bases de Données

Historiquement l'informatisation a connu 2 approches : approche **Fichiers** et approche **Bases de Données** :

- Approche Fichiers : les fichiers dépendent des programmes qui les utilisent → **Redondance** et **incohérence** des données.
- Approche Bases de données : intégrer les données, aussi peu redondantes que possible et les gérer par un Système de Gestion de Bases de Données (**SGBD**).
 - Un SGBD permet de créer, modifier, supprimer, ajouter et consulter les données de la base tout en assurant les fonctionnalités suivantes :
 - + Intégrité des données
 - + Partage des données
 - + Sécurité des données
 - + Récupération des données en cas de panne
 - + ...

Chap 2 Modèle Conceptuel de Données



Chap 2 Modèle Conceptuel de Données

I- Modèle Entité/Association

- **Propriété** : donnée que l'on perçoit sur une entité ou association

➡ 3 types de propriétés

- **Calculée** → Calculée en fonction des autres propriétés : Moyenne_générale = $\sum \text{note} * \text{Coef} / \sum \text{Coef}$
- **Concaténée** → Composée de plusieurs propriétés : Étudiant = (nom , prénom)
- **Élémentaire** → Ni calculée, ni composée : nom , prénom , note , coefficient.

- **Occurrence** : chaque valeur prise par une propriété.

- ➡ Les propriétés calculées doivent être éliminées du MCD pour réduire l'espace mémoire de stockage
- ➡ Les propriétés concaténées doivent être éliminées du MCD pour réduire le temps de recherche des informations

Chap 2 Modèle Conceptuel de Données

I- Modèle Entité/Association

- Entité : ensemble de propriétés caractérisant un objet.

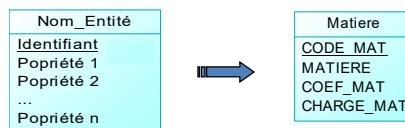
➡ Matière = (CODE_MAT, MATIERE, COEF_MAT, CHARGE_MAT)



- Identifiant d'une entité : Groupe minimum de propriétés identifiant le reste des propriétés.

Exemple : l'identifiant de l'entité Matière est CODE_MAT

Formalisme d'une entité :



Chap 2 Modèle Conceptuel de Données

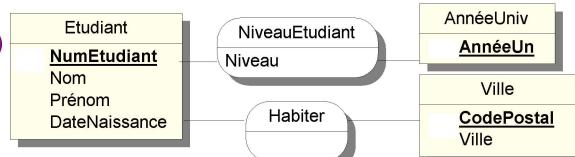
I- Modèle Entité/Association

- Association : liaison entre 2 ou plusieurs entités (binaire, ternaire, quaternaire, ... n-aire)

➡ Une association peut contenir ou non des propriétés

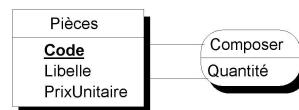


Exemple (Modèle Entité/Association)



Cas particulier :

Association (réflexive) qui relie une entité à elle-même



Chap 2 Modèle Conceptuel de Données

I- Modèle Entité/Association

- **Identifiant (Clé primaire) d'une Association :** Composition des identifiants des entités formant cette association.

Exemple :

- L'identifiant de l'association **NiveauEtudiant** est ([NumEtudiant](#) , [AnnéeUn](#))
- L'association **Habiter** n'a pas de identifiant.

Chap 2 Modèle Conceptuel de Données

II- Dépendances Fonctionnelles

- **Dépendance fonctionnelle :**

Propriété1 \xrightarrow{DF} *Propriété2* si une valeur de la 1ère correspond au plus à une valeur de la seconde.

Exemples : *NumEtud* , *Nom* \xrightarrow{DF} *Prénom* et *NumEtud* \xrightarrow{DF} *Nom* , *Prénom*

- **Dépendance Fonctionnelle Élémentaire :**

P₁ \xrightarrow{DFE} *P₂* si $P_1 \rightarrow P_2$ et aucune partie **stricte** de *P₁* n'entraîne *P₂*

<u>Exemples :</u>	<i>NumEtud</i> , <i>Nom</i> \xrightarrow{DF} <i>Prénom</i>	Dépendance Fonctionnelle non Élémentaire
	<i>NumEtud</i> \xrightarrow{DFE} <i>Nom</i> , <i>Prénom</i>	Dépendance Fonctionnelle Élémentaire

Chap 2 Modèle Conceptuel de Données

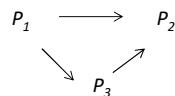
II- Dépendances Fonctionnelles

- Dépendance Fonctionnelle Élémentaire Directe :

$P_1 \xrightarrow{DFED} P_2$ si :

$P_1 \xrightarrow{DFE} P_2$ et si elle n'existe aucune propriété P_3 telle que $P_1 \rightarrow P_3$ et $P_3 \rightarrow P_2$

➡ Pas de transitivité entre deux propriétés



Exemple : Les deux dépendances $NumEtud \rightarrow Prénom$ et $NumEtud \rightarrow Nom$ sont DFED.

Par contre, ni $Nom \rightarrow Prénom$, ni $NumEtud \rightarrow Ville$ ne sont des DFED. En effet, la 1^{ère} n'est pas une DF et la 2^{ème} n'est pas directe : $NumEtud \rightarrow CodePostal$ et $CodePostal \rightarrow Ville$.

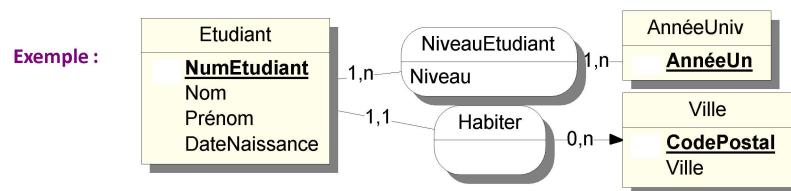
➡ La conception d'une BD dont les dépendances entre les propriétés reliées par une DFED permet de réduire l'espace mémoire de stockage.

Chap 2 Modèle Conceptuel de Données

III- Cardinalités

- Cardinalités : La Cardinalité d'une entité à travers une association est le nombre d'occurrences de cette association correspondant à une occurrence de l'entité.

➡ { Cardinalités maximales : le nombre maximum d'occurrences de la relation.
Cardinalités minimales : le nombre minimum d'occurrences de la relation.



➡ $0 \leq \text{Cardinalités minimales} \leq 1$ et : $1 \leq \text{Cardinalités maximales} \leq n$

Chap 2 Modèle Conceptuel de Données

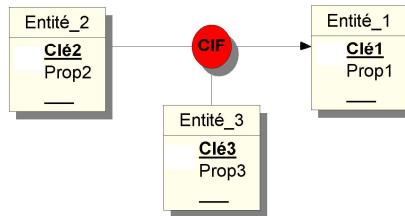
IV- Contrainte d'Intégrité Fonctionnelle

- **CIF** : La Contrainte d'Intégrité Fonctionnelle formalise une DF entre une ou plusieurs entités dites *origines* et une entité dite *cible*.

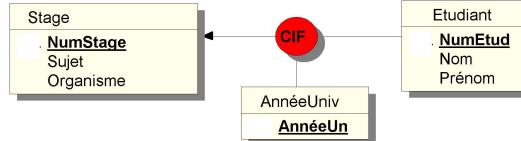
→ Pour toutes occurrences des entités origines correspond au plus une occurrence de l'entité cible

Formalisme d'une CIF :

→ {
Entité_1 : Cible
Entité_2 et _3 : origines



Exemple :



Chap 2 Modèle Conceptuel de Données

V- Etapes de construction d'un MCD

- **Étape 1 : Établissement de la liste des propriétés.**

➤ Établir la liste à partir de chaque document recueilli.

→ La propriété peut apparaître sous deux formes dans un document :

Valeur ⇒ interpréter la valeur pour identifier la propriété.
Exemple : Emploi du temps 4ème année ⇒ Niveau

Propriété implicite

Propriété : valeur
Exemple : Étudiant : X Y ⇒ Étudiant

Propriété explicite

➤ Éliminer les synonymes et régler les polysèmes.

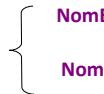
Chap 2 Modèle Conceptuel de Données

V- Etapes de construction d'un MCD

- **Synonymes** : deux ou plusieurs propriétés ayant la même signification
- **Polysème** : Une propriété qui se trouve dans le même document ou dans d'autres avec plusieurs sens (significations).

Exemple :

 2 synonymes : on en garde un seul.

 On précise le sens de la propriété polysème

Chap 2 Modèle Conceptuel de Données

V- Etapes de construction d'un MCD

- **Étape 2 : Établissement du dictionnaire de données**

Nom abrégé	Nom détaillé	Nature	Type	Taille (en octet)	Remarques
....



- Nom abrégé de la propriété
- Nom détaillé de la propriété
- Sa nature : E (Elémentaire), CA (Calculée) et CO (Concaténée)
- Type : N (Numérique), A (Alphabétique), AN (AlphaNum.), Date, Image, OLE, ...
- Remarque : si la propriété est calculée, on écrit l'expression du calcul; si elle est concaténée, on écrit l'expression de décomposition.

Chap 2 Modèle Conceptuel de Données

V- Etapes de construction d'un MCD

- **Étape 3 : Établissement du Graphe de Dép. Fonct. Élém. (GDF)**

→ **2 Méthodes**

Établir le GDF d'un document et fusionner les MCDs par la suite
Établir le GDF global pour établir un MCD global directement

→ Il faut éliminer les transitivités du GDF pour rendre les dépendances fonctionnelles élémentaires des DFED.

Etude de cas N°1 : Gestion d'une entreprise commerciale

- Collecte de 2 documents : **Fiche du client** et **Bon de commande**
- TVA fixe = **20%**

Document client

N° : 1712
Nom : Alami
Prénom : Youssef
Tél : 06 11 11 11 11
Adresse : 13, Avenue Med V, Meknès
Mail : y.alami@gmail.com

Bon de commande

N° : 357892	Date : 22/03/2020
Client : 1712	Nom : Alami Prénom : Youssef

Référence	Désignation	PU	Quantité	Montant (HT)
.....

Total Hors Taxes (HT) :
Net (TTC) :

Etape 1 : Liste des propriétés

Dans cet exemple, toutes les propriétés sont explicites :

N°, Nom, Prénom, Tél, Adresse, Mail, N°, Date, Client, Référence, Désignation, PU, Quantité, Montant, Total_HT, Net.

i- Propriétés synonymes :

$$\begin{matrix} N° \\ \text{Client} \end{matrix} \rightarrow NCLIENT$$

ii- Propriétés polysèmes :

$$N° \rightarrow \begin{cases} NCOMMANDE \\ NCLIENT \end{cases}$$

iii- Propriétés Composées :

Dans les règles de gestion, on souhaite faire des traitements (statistiques) sur des parties de l'adresse (exple : nombre de clients habitant une rue ou une ville). On décompose alors la propriété en des sous propriétés utiles :

Adresse = (RUE, VILLE) :

On peut ne pas considérer le N° du logement s'il ne sera pas utile.

iv- Propriétés calculées :

- **MONTANT = PU * QUANTITE**
- **TOTALHT = $\sum PU * QUANTITE$**
- **NET = $(1 + TVA) * (\sum PU * QUANTITE) = 1.2 * \sum PU * QUANTITE$**

Etape 2 : Dictionnaire de données

Nom abrégé	Nom détaillé	Nature	Type	Longueur	Remarques
ADRESSE	Adresse du client	CO	AN	30	(RUECLI,VILLECLI)
DATECDE	Date de commande	E	Date	8	
DESIGN	Désignation du produit	E	AN	20	
MONTANT	Montant du produit commandé	CA	N	8	PU * QTE
MAILCLI	Mail du client	E	T (Texte)	20	
NCLI	Numéro du client	E	N	5	
NDE	Numéro de commande	E	N	8	
NET	Net à payer	CA	N	12	$1.2 * \sum PU * QTE$
NOMCLI	Nom du client	E	A	15	
PRENCLI	Prénom du client	E	A	15	
PU	Prix unitaire du produit	E	N	6	(Format: xxx.xx)
QTE	Quantité du produit commandé	E	N	3	
REF	Référence du produit commandé	E	AN	15	
RUECLI	Rue du Client	E	AN	20	
TELCLI	Téléphone du Client	E	Texte	10	
TOTALHT	Total hors taxes	CA	N	10	$\sum PU * QTE$
VILLECLI	Ville du client	E	A	20	

Remarque : Les propriétés composées et calculées ne seront pas considérées dans la suite de la conception

Etape 3 : Liste des dépendances fonctionnelles

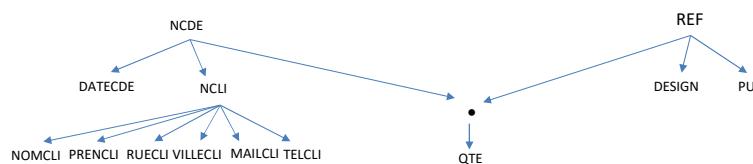
- On essaie de déterminer les *dépendances fonctionnelles élémentaires directes* qui pourraient exister en les propriétés (simples) retenues du dictionnaire de données.
- On commence par les propriétés simples qui entraînent d'autres propriétés. Puis, on cherche le groupement minimum de propriétés qui impliquent d'autres propriétés.

1. NCLI → NOMCLI, PRENCLI, MAILCLI, TELCLI , RUECLI, VILLECLI (DFED)
2. NCDE → DATECDE, NCLI (DFED)
3. REF → DESIGN, PU (DFED)
4. NCDE, REF → QTE

- Les propriétés se trouvant à gauche de la DF sont dites propriétés **sources**, et celles à droite on les appelle propriétés **buts** (ou cibles).
- Toutes les propriétés élémentaires du dictionnaire de données doivent apparaître dans la liste des dépendances fonctionnelles.

Etape 4 : Graphe de dépendances fonctionnelles (GDF)

- Il s'agit tout simplement, de fusionner les dépendances établies, dans un graphe (arborescence) de sorte que chaque propriété y figure une seule fois.



Etape 5 : Construction du Modèle Conceptuel de Données (MCD)

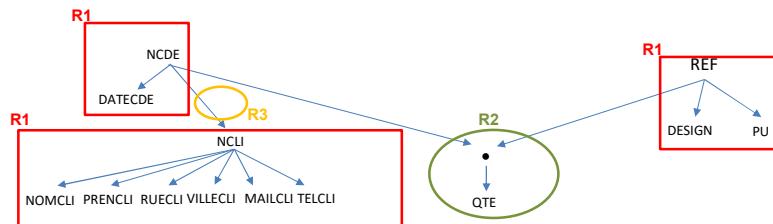
Règles de construction :

La construction se fait du bas en haut.

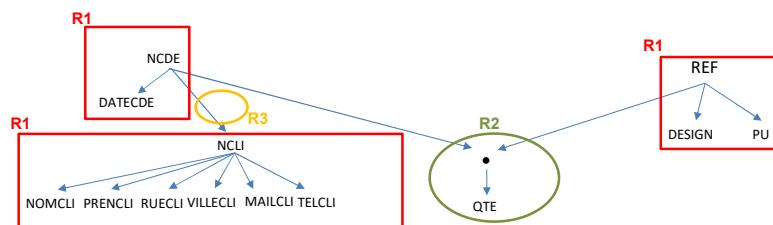
R1 : les propriétés qui dépendent d'une propriété simple, forment avec elle une **Entité** dont la clé primaire est cette propriété simple.

R2 : les propriétés qui dépendent d'une propriété composée (de plusieurs prop.), forment une **association** porteuse de ces propriétés.

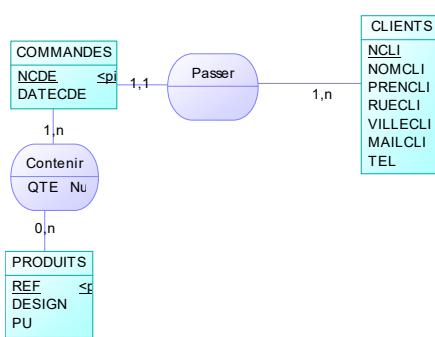
R3 : lorsqu'il y a un lien (une flèche) d'une entité vers une autre, il sera traduit par une relation de contrainte d'intégrité fonctionnelle (CIF).



Etape 5 : Construction du Modèle Conceptuel de Données (MCD)



MCD :



Etude de cas N°2 : Gestion d'une clinique

Document PATIENT

Clinique Belle Vue

Patient : Rachidi Med

Profession : Technicien

Date de naissance : 30/09/1980

Adresse : 23, Avenue Med V, Meknès

Tél : 06.34.87.90.11

Organisme mutuelle : CNOPS

Date d'entrée	Date de sortie
01/01/2014	10/01/2014
03/02/2014	07/02/2014
.....

Etude de cas N°2 : Gestion d'une clinique

Document FACTURE

Clinique Belle Vue

N° Facture : 27

Patient : Rachidi Med

Date d'entrée : 01/01/2014

Date de sortie : 10/01/2014

Durée : 10 jours

Opérations	Date	Médecin	Montant
Fibroscopie	02/01/2014	Amine Saidi	1000
Lavage d'estomac	07/01/2014	Ali Yousfi	1200
.....

Total des opérations : 2200,00 DH

Frais chambre : 3000,00 DH

Total : 5200,00 DH

Mutuelle : 4160,00 DH

Net à payer : 1040

Date : 12/01/2014

Etude de cas N°2 : Gestion d'une clinique

Règles de gestion

1. Un seul médecin est responsable d'une opération ;
2. Le pourcentage de couverture de mutuelle dépend de l'organisme de mutuelle et de l'opération effectuée
3. Un patient peut être hospitalisé zéro ou plusieurs fois dans la clinique ;
4. Une simple consultation est considérée aussi, comme opération ;
5. Durant une hospitalisation, un patient peut effectuer une ou plusieurs opérations ;
6. Un patient peut effectuer une opération sans qu'il soit hospitalisé ;
7. Le montant de la mutuelle est récupéré directement de l'organisme de mutuelle, si le patient est mutualiste. On suppose aussi que la mutuelle ne couvre pas les frais de chambre.
8. On ne s'intéresse pas à la gestion des médecins ;
9. Le tarif (par jour) d'une chambre dépend de sa catégorie ;
10. Le montant d'une opération est le même pour tous les patients ;
11. Aucun traitement ne sera fait sur une partie de l'adresse ni sur la date de naissance.

Etape 1 : Liste des propriétés

On considérera directement les noms abrégés des propriétés recensées :

*PATIENT, PROFESSION, DATE_NAISS, ADRESSE, TEL, ORG_MUT, DATE_ENT, DATE_SORT,
NFACT, DUREE, OPERATION, DATE, MEDECIN, MONTANT_OPER, TOTAL_OPER,
FRAIS_CHAMB, TOTAL, MUTUELLE, PRCNT_MUT, NET, DATE.*

i- Ajout de nouvelles propriétés

- **NUM_PAT** : Numéro du patient (*son identifiant*)
- **CODE_OPER** : Code de l'opération (*son identifiant*)
- **NUM_CHMAB** : Numéro d'une chambre
- **CATEG_CHAMB** : Code d'une catégorie chambres
- **TARIF_CATEG** : Tarif par nuit des chambres d'une catégorie.
- **CODE_MED** : Code du médecin
- **SPEC_MED** : Spécialité du médecin
- **TEL_MED** : Téléphone du médecin

ii- Synonymes

Pas de synonymes

iii- Polysèmes



Etape 1 : Liste des propriétés

iii- Propriétés composées

- Parmi les propriétés qui peuvent être décomposées, ici, il y a l'Adresse et les Dates. Dans les règles de gestion, il est mentionné qu'il n'y aura pas de traitement utilisant des parties de ces propriétés. Ils seront alors considérées comme élémentaires (simples).
- $PATIENT = (NOM_PAT, PREN_PAT)$: Nom et prénom du patient.
- $MEDECIN = (NOM_MED, PREN_MED)$: Nom et prénom du médecin.

v- Propriétés calculées

- $- DUREE = (DATE_SORT - DATE_ENT) + 1$
- $- TOTAL_OPER = \sum MONTANT_OPER$
- $- FRAIS_CHAMB = (DATE_SORT - DATE_ENT + 1) * TARIF_CATEG$
- $- TOTAL = \sum MONTANT_OPER + (DATE_SORT - DATE_ENT + 1) * TARIF_CATEG$
- $- MUTUELLE = \sum (MONTANT_OPER * PRCENT_MUT)$
- $- NET = TOTAL - MUTUELLE$
 $= [\sum MONTANT_OPER + (DATE_SORT - DATE_ENT + 1) * TARIF_CATEG] * (1 - PRCENT_MUT)$

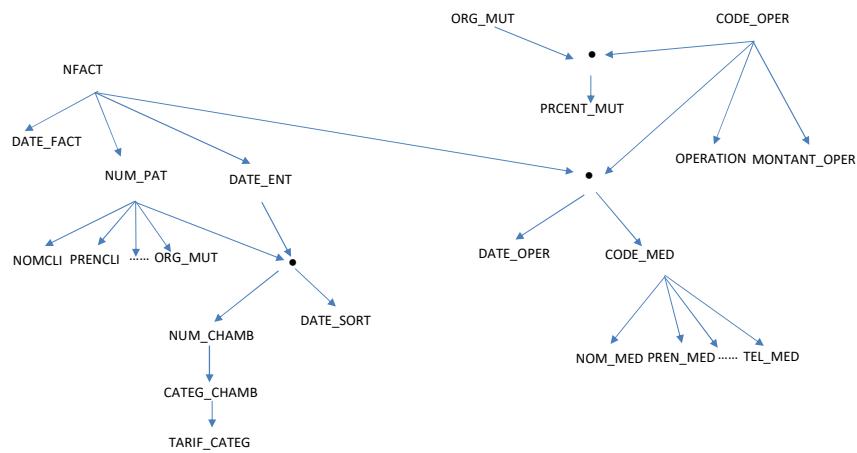
Etape 2 : Dictionnaire de données

De la même manière que l'étude de cas N°1, on dresse un tableau contenant les caractéristiques des propriétés déterminées.

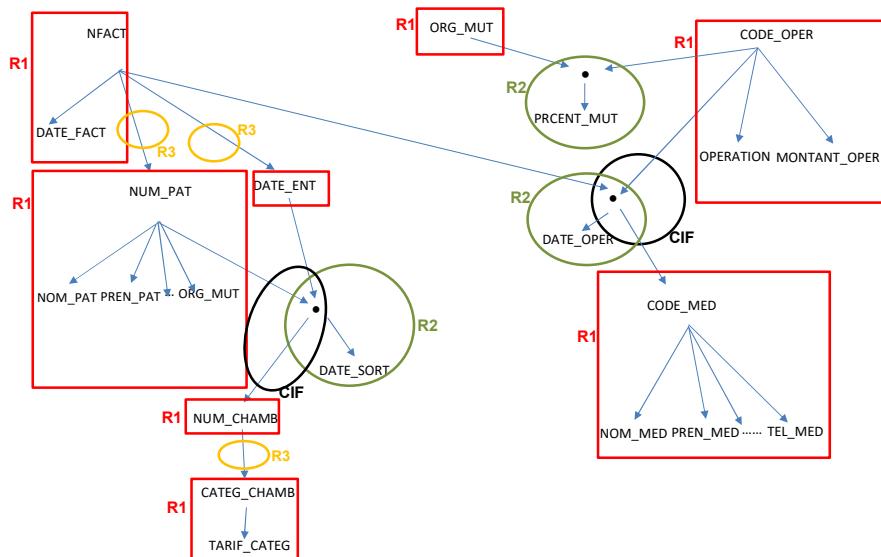
Etape 3 : Liste des dépendances fonctionnelles

- $NUM_PAT \rightarrow NOM_PAT, PREN_PAT, PROFESSION, DATE_NAISS, ADRESSE, TEL, ORG_MUT$ (DFED)
- $NFACT \rightarrow NUM_PAT, DATE_FACT, DATE_ENT, DATE_SORT, NUM_CHAMB$: (DFED)
- $CODE_OPER \rightarrow OPERATION, MONTANT_OPER$: (DFED)
- $CODE_MED \rightarrow NOM_MED, PREN_MED, SPEC_MED, TEL_MED$: (DFED)
- $NUM_CHAMB \rightarrow CATEG_CHAMB$: (DFED)
- $CATEG_CHAMB \rightarrow TARIF_CATEG$: (DFED)
- $NUM_PAT, DATE_ENT \rightarrow DATE_SORT, NUM_CHAMB$: (DFED)
- $NFACT, CODE_OPER \rightarrow CODE_MED$: (DFED)
- $CODE_OPER, ORG_MUT \rightarrow PRCENT_MUT$: (DFED)

Etape 4 : Graphe de dépendances fonctionnelles (GDF)

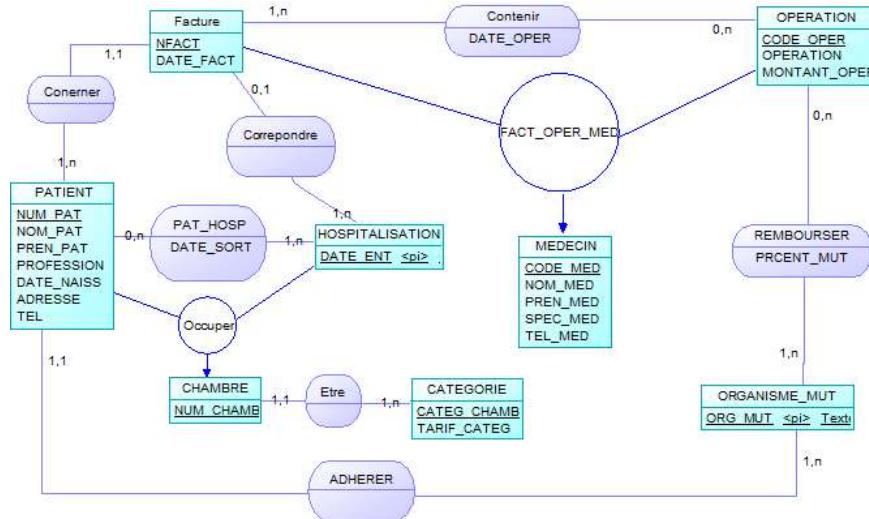


Etape 5 : Construction du Modèle Conceptuel de Données (MCD)



Etape 5 : Construction du Modèle Conceptuel de Données (MCD)

MCD (avec relation CIF) :



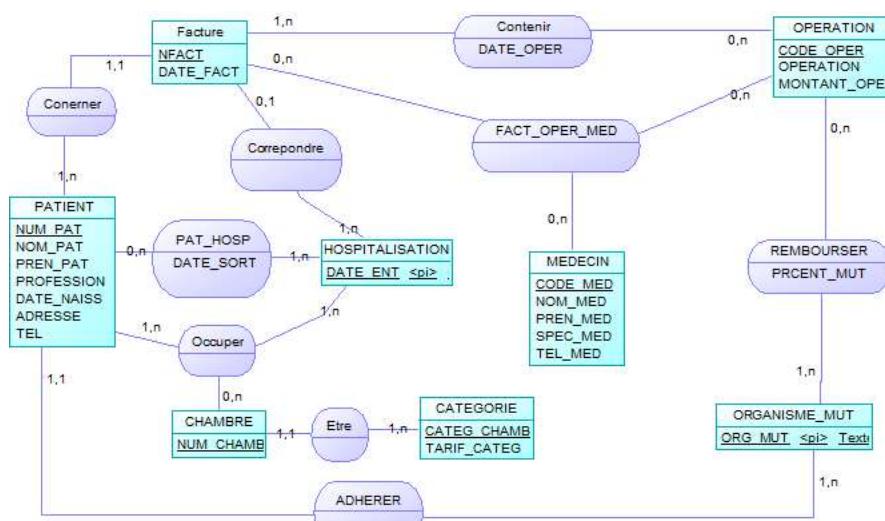
Bases de Données

37/96

A. Ahmadi

Etape 5 : Construction du Modèle Conceptuel de Données (MCD)

MCD (Sans relation CIF) :



Bases de Données

38/96

A. Ahmadi

Chap 2 Modèle Conceptuel de Données

VI- Normalisation

- Si le MCD a été construit en suivant l'approche se basant sur les dépendances fonctionnelles, il sera normalisé.
- Si le MCD a été élaboré de manière intuitive, il faudrait vérifier s'il est normalisé. i.e., il est en troisième forme normale (3FN) :
 - Une entité ou association est en première forme normale (respectivement deuxième et troisième FN) si toutes ses propriétés sont reliées avec sa clé (identifiant) par une DF (respectivement DFE et DFED).
 - Le MCD est en première FN (respectivement 2FN, 3FN) si toutes ses entités et associations sont en 1FN (respectivement 2FN, 3FN).

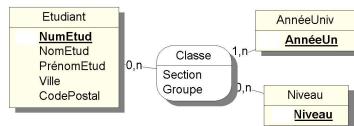
Remarques :

- Si une entité E1, ayant une clé élémentaire, est en 1FN alors elle est en 2FN.
- Si une entité E1 ou une association R1 est en 2FN et contiennent une seule propriété qui est la clé, alors elles sont en 3FN.

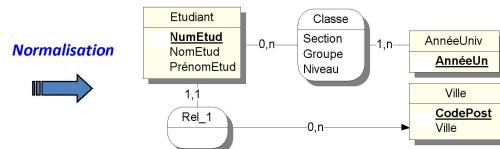
Chap 2 Modèle Conceptuel de Données

VI- Normalisation

Exemple : Normaliser le MCD suivant :



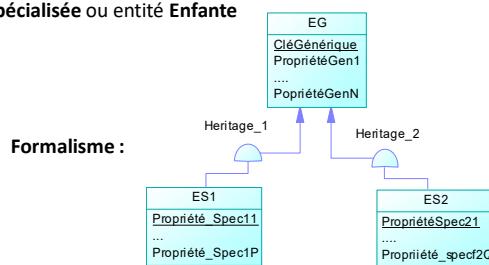
- L'entité **Etudiant** est en **2FN** mais elle n'est pas en **3FN** (**la ville** ne dépend pas **directement** de **NumEtud**) : on a : **NumEtud → CodePostal** et **CodePostal → Ville** (il y a une transitivité)
- Les deux entités **AnnéeUniv** et **Niveau** sont en **3FN** (elles contiennent uniquement la clé).
- L'association **Classe** est en **1FN** mais pas en **2FN** (**la section et le groupe** dépendent **seulement** de **NumEtud et AnnéeUn**)
- ⇒ Le MCD est en **1FN** : il faut rendre **Étudiant** et **Classe** en **3FN**



Chap 2 Modèle Conceptuel de Données

VII- Héritage

- Une entité **ES** hérite d'une deuxième entité **EG** si ES hérite des propriétés (propriétés génériques) de EG en plus de ses propres propriétés (propriétés spécifiques).
- ⇒ **EG** est appelée entité **Générique** ou entité **Parente**
- ⇒ **ES** est appelée entité **Spécialisée** ou entité **Enfante**

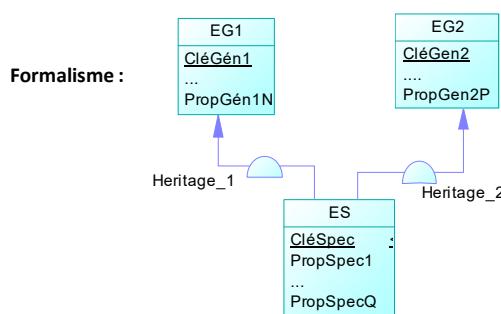


- Les deux entités filles héritent de l'entité Parente, les propriétés **propriété_Gen1**, ... **prop_GenN**.
- En plus des propriétés héritées, chacune des entités **ES1**, (resp. **ES2**) a ses propres propriétés spécifiques **PropriétéSpec11**, ..., **PropriétéSpec1P** (resp. **PropriétéSpec21**, ..., **PropriétéSpec2Q**).

Chap 2 Modèle Conceptuel de Données

VII- Héritage

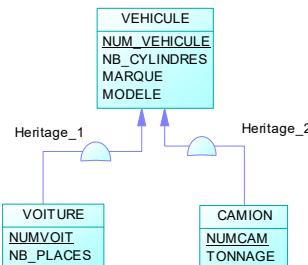
- Une entité **ES** peut hériter de 2 ou de plusieurs entités génériques :



Chap 2 Modèle Conceptuel de Données

VII- Héritage

Exemple :

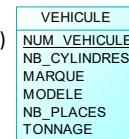


Apport de l'héritage : optimisation de l'espace mémoire.

En effet : Absence de l'héritage \Rightarrow Éliminer les entités spécialisées et garder une seule entité (l'entité générique)

\Rightarrow s'il s'agit d'un camion, le *NB_PLACES* ne sera pas saisi et s'il s'agit d'une voiture le *TONNAGE* ne le sera pas également.

\Rightarrow Espace mémoire perdu (cases vides : réservées mais non remplies)



Chap 3 Modèle Logique de Données (MLD)

- *MCD* : Représentation fidèle des données du SI de l'univers à informatiser.
- Cette représentation n'est pas utilisable directement sur une machine.
- Passer par un niveau intermédiaire (niveau logique) en transformant le *MCD* en **MLD** (Modèle Logique de Donnée) proche de la réalité d'un système informatique (Ordinateur, Station de travail, Serveur, etc.).
- Le *MLD* permet de modéliser la structure selon laquelle les données seront stockées dans la future base de données, en fonction du système de Gestion de Bases de Données (SGBD) à utiliser. On a plusieurs possibilités :
 - Un *MLD* sous la forme de fichiers ;
 - Un *MLD* sous la forme d'une base de données hiérarchique type XML ;
 - Un *MLD* sous la forme d'une base de données relationnelle
 - Un *MLD* sous la forme d'une base de données objet.
- Chaque modèle adopte des techniques d'organisation des données adaptées aux logiciels qui vont les exploiter.
- Ce cours traitera le *MLD* relationnel, exploitable par les SGBDR (SGBD Relationnel) tels que : *Oracle*, *SQL Server*, *MySQL*, *ACCESS*, etc.

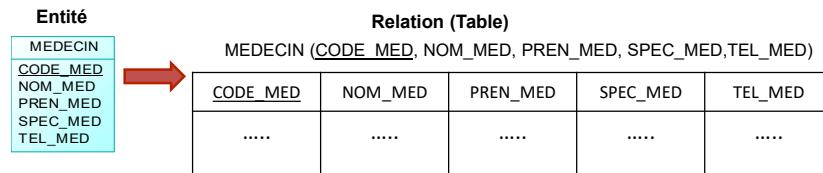
Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

- Le modèle **Entité/Association** exprime les relations entre les entités.
- Le modèle **Relationnel** est un ensemble de relations (tables liées).
- Une table est en liaison avec une autre si elles ont un attribut commun. Elles peuvent, aussi, être liées via deux attributs de même type.

Règle 1 :

- Chaque entité devient une **relation** (table) et chaque attribut de l'entité devient un **champ** (colonne) de cette table.
- L'identifiant de l'entité devient la **clé primaire** de la relation (table).
- La ligne d'une table est dite **enregistrement** de la table (record).

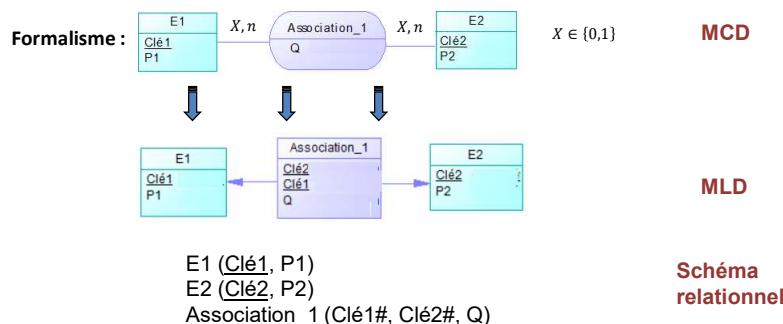


Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Une association soit elle se transforme en table, soit elle disparaît, avec migration de sa clé dans une autre table.

Règle 2 : Cardinalités Maximales égales **n** du côté E1 et du côté E2 : $(x,n) - (x,n)$



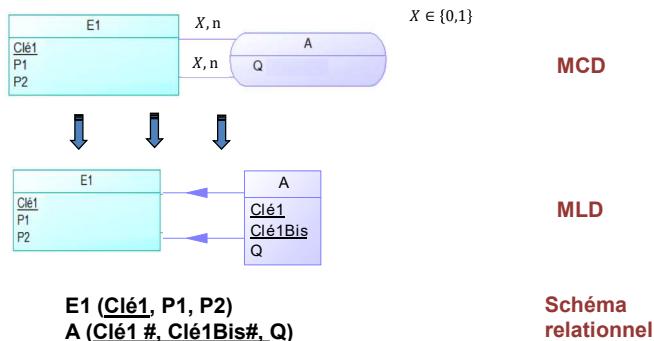
L'association se transforme, elle aussi en une table dont la clé est la concaténation des clés des entités qui y participent

Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Règle 2 : Cardinalités Maximales égales n du côté E1 et du côté E2 : : (x,n) – (x,n)

Cas particulier : Association réflexive



Remarque : Ici la clé **clé1** a migré 2 fois dans A. Or, une table ne peut contenir deux propriétés portant le même nom, la 2^{ème} a été surnommée en **Clé1Bis**.

Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Règle 2 : Cardinalités Maximales égales n du côté E1 et du côté E2 : : (x,n) – (x,n)

Cas particulier : Association réflexive

Exemple1 :

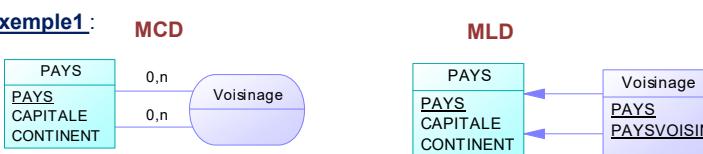


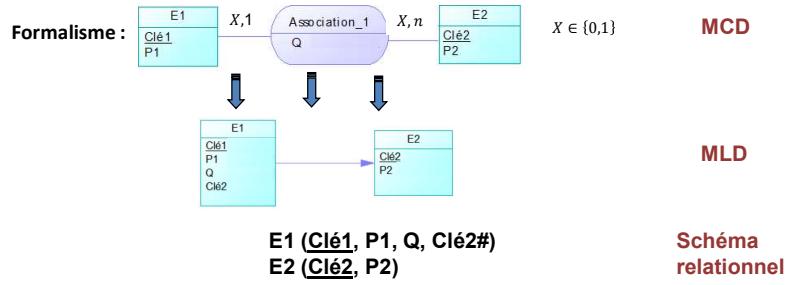
Schéma relationnel

PAYS (PAYS, CAPITALE, CONTINENT)
Voisinage (PAYS, PAYSVOISIN)

Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Règle 3 : Cas d'une association binaire hiérarchique $(x,1) - (x,n)$: (CIF)



- L'association disparaît dans le MLD, mais il y a migration de la clé primaire **Clé2** de l'entité du côté de cardinalité (x,n) dans l'autre entité du côté de cardinalité $(x,1)$ pour devenir clé étrangère.
- La clé étrangère **Clé2** dans **E1** est un attribut (colonne) normal (non souligné). On lui ajoute souvent, un # à la fin pour mentionner que c'est une **clé étrangère**.
- C'est cette clé étrangère (colonne commune) qui assure, en fait, la liaison entre E1 et E2.

Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Règle 3 : Cas d'une association hiérarchique $(x,1) - (x,n)$

Cas particulier : Association réflexive (unaire)

Exemple : chaque employé a, au plus, un supérieur hiérarchique direct. Un employé peut ou non avoir des subordonnés

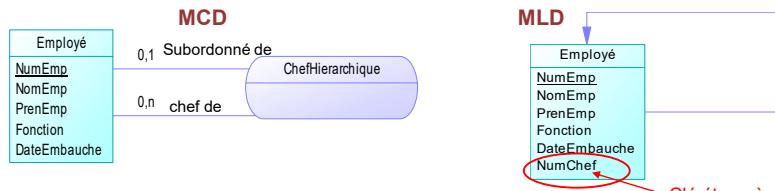


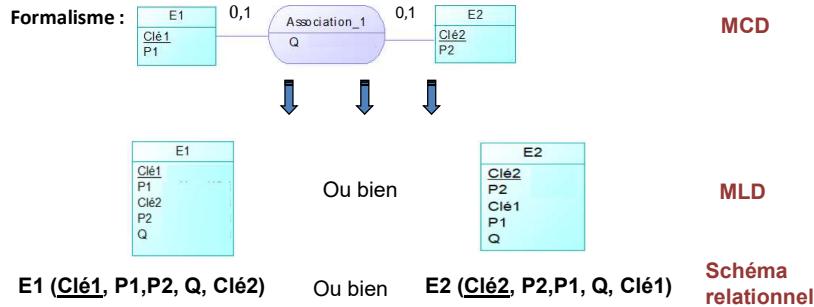
Schéma relationnel :

Employé (NumEmp, NomEmp, PrenEmp, Fonction, DateEmbauche, NumChef#)

Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Règle 4 : Cas d'une association binaire (0,1) - (0,1) :



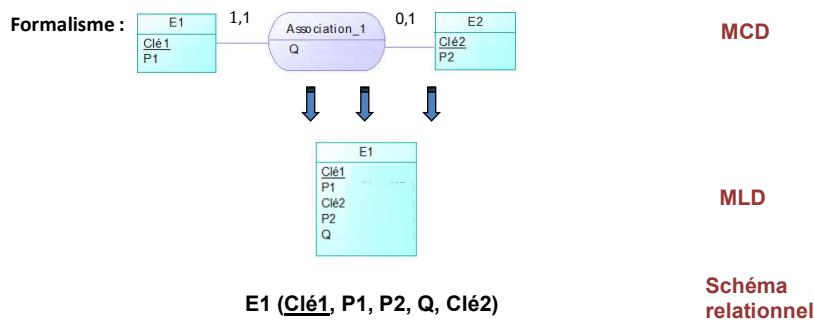
Remarques :

- 1- On opte pour la solution qui laisserait moins de lignes ayant des colonnes vides ((clé2,P2)/(clé1,P1) non renseignées) lors du remplissage de la table.
- 2- Si les entités E1 et E2 sont liées à d'autres entités, le MLD contiendra les 2 tables E1 et E1 avec migration de clé1 dans E2 et clé2 dans E1. On peut aussi avoir 3 entités dans le MLD : E1, E2 et Association_1. Cette dernière aura comme clé clé1,clé2 et contiendra l'attribut Q.

Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Cas d'une association binaire (1,1) - (0,1) :



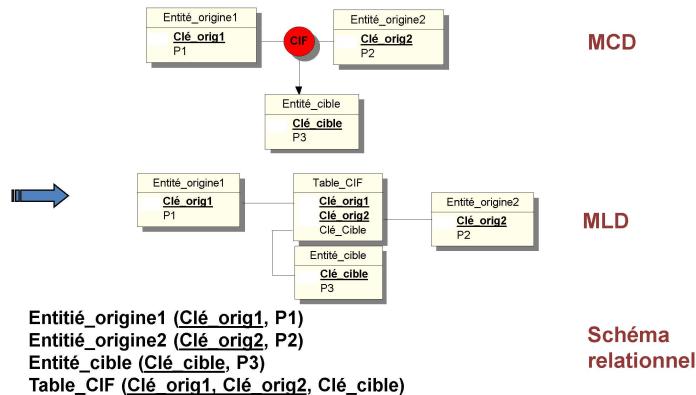
Remarque :

Ici, il vaut mieux considérer E1 (du coté cardinalité (1,1)), car à chaque occurrence de E1, correspond une (et une seule) occurrence de E2. On ne risque pas alors, d'avoir une ligne de E1 pour laquelle, les attributs Clé2 et P2 restent vides.

Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Règle 5 : Cas d'une relation CIF entre plusieurs entités :



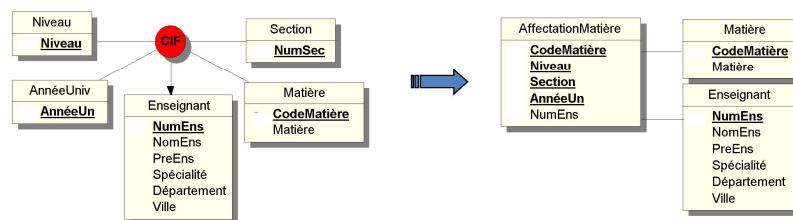
- La clé_Cible est **primaire** dans la table *Entité_Cible* et **étrangère** dans la table *Table_CIF*
- La clé de la table *Table_CIF* est composée de **Clé_orig1** et **Clé_orig2**.

Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Règle 5 : Cas d'une relation CIF entre plusieurs entités :

■ Exemple :



Remarque :

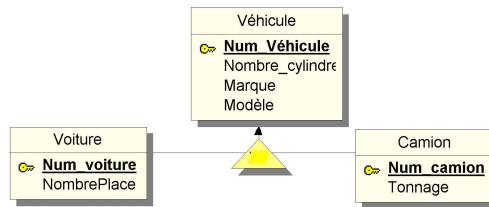
Les tables qui ne contiennent que la clé primaire peuvent être éliminées du MLD. En effet, elles ne donneront aucune information utile, puisque leurs clés primaires sont déjà migrées dans les autres tables.

Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Règle 6 : Cas d'un héritage:

Exemple :



- ➡ Deux cas : { - Favoriser la généralisation : Éliminer les entités enfantées (spécialisées)
- Favoriser la spécialisation : garder les 3 entités

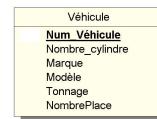
Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Règle 6 : Cas d'un héritage:

- Favoriser la généralisation :

⇒ Garder l'entité générique et éliminer les entités spécialisées ➡



- Favoriser la spécialisation :

⇒ Garder à la fois l'entité générique et les entités spécifiques :

- ➡ Deux cas : { - Valeurs des clés spécialisées sont héritées de la clé générique
- Valeurs des clés spécialisées ne sont pas héritées de la clé générique

Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Règle 6 : Cas d'un héritage:

- Valeurs des clés spécialisées sont héritées de la clé générique

The diagram illustrates inheritance in a relational model. At the top is a generic table 'Véhicule' with columns: Num_Véhicule, Nombre_cylindre, Marque, and Modèle. It contains four rows with values 1, 2, 3, and 4 respectively. Red arrows point from these rows to two specialized tables below: 'Voiture' and 'Camion'. The 'Voiture' table has columns Num_Voiture and NombrePlace, with rows 1 and 3. The 'Camion' table has columns Num_Camion and Tonnage (t), with rows 2 and 4. Blue arrows indicate the inheritance of the primary key Num_Véhicule from the generic table to the specialized tables.

Num_Véhicule	Nombre_cylindre	Marque	Modèle
1	4	Renault	R19
2	6	SCANIA	R144 LA 4X2 NA
3	5	Peugeot	506
4	6	SCANIA	R 124 LA 4X2 NA

Num_Voiture	NombrePlace
1	5
3	6

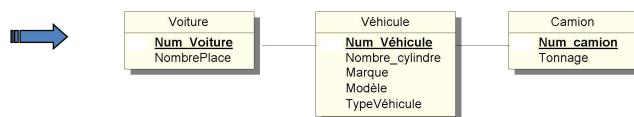
Num_Camion	Tonnage (t)
2	26
4	19

Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Règle 6 : Cas d'un héritage:

- Valeurs des clés spécialisées sont héritées de la clé générique



- On relie les tables spécialisées avec la table générique via les clés
- On rajoute une propriété *TypeVéhicule* à la table générique pour accéder directement à la bonne table spécialisée : Réduire le temps de recherche des informations spécialisées

Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

Règle 6 : Cas d'un héritage:

- Valeurs des clés spécialisées ne sont pas héritées de la clé générique

Num_Véhicule	Nombre_cylindre	Marque	Modèle
1	4	Renault	R19
2	6	SCANIA	R144 LA 4X2 NA
3	5	Peugeot	506
4	6	SCANIA	R 124 LA 4X2 NA

Num_Voiture	NombrePlace	Num_Camion	Tonnage (t)
1	5	1	26
2	6	2	19

- Deux cas : {
- introduire la clé spécialisée dans la table générique
 - introduire la clé générique dans chaque table spécialisée

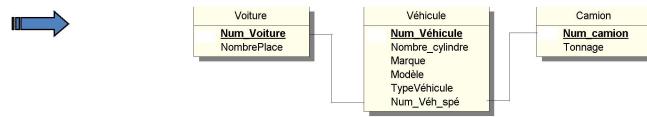
Chap 3 Modèle Logique de Données (MLD)

Règles de passage du MCD au MLD (relationnel)

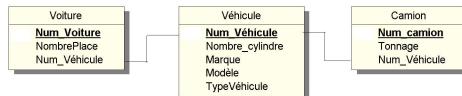
Règle 6 : Cas d'un héritage:

- introduire la clé spécialisée dans la table générique

⇒ introduire une propriété **TypeVéhicule** pour distinguer le type de la clé spécialisée introduite dans la table générique.



- introduire la clé générique dans toutes les tables spécialisées



Partie 2 : Langage SQL

61

ENSAM-Meknès

A.AHMADI

2016/2017

Chap. 1 Langage de définition de données

I- Création d'une base de données

62

```

DROP table Detail;
DROP table Facture;
DROP table Produit;
DROP table Client;

CREATE TABLE Client (cocli CHAR(4) CONSTRAINT pk_Client Primary Key,
                    Nomcli VARCHAR(30) CONSTRAINT ck_Client_nomcli CHECK (nomcli=UPPER(nomcli)),
                    Ville VARCHAR(15) CONSTRAINT nn_ville NOT NULL
                    );
CREATE TABLE Produit (copro CHAR(3) CONSTRAINT pk_produit PRIMARY KEY,
                     libelle VARCHAR(15),
                     PU NUMBER(5,2)
                     );
CREATE TABLE Facture (nufact NUMBER(5) CONSTRAINT pk_facture PRIMARY KEY,
                     Datefact DATE default sysdate,
                     cocli CHAR(4) CONSTRAINT fk_facture_client REFERENCES Client(cocli) ON DELETE CASCADE,
                     Montant NUMBER(8,2) CONSTRAINT ck_montant CHECK (montant>0)
                     );

```

A. AHMADI

Langage SQL

Chap. 1**Langage de définition de données****I- Crédation d'une base de données**

(63)

```
CREATE TABLE Detail (nufact NUMBER(5),
    Copro CHAR(3),
    Qte NUMBER,
    CONSTRAINT fk_detail_produit FOREIGN KEY (copro) REFERENCES Produit(copro),
    CONSTRAINT fk_detail_fact FOREIGN KEY (nufact) REFERENCES Facture(nufact),
    CONSTRAINT pk_detail PRIMARY KEY (nufact,copro)
);
```

A. AHMADI

Langage SQL

Chap. 1**Langage de définition de données****II- Contraintes d'intégrité**

(64)

1- Ajout d'une contrainteExple1 :

```
ALTER TABLE Client
ADD CONSTRAINT un_client_NomCli UNIQUE (NomCli) ;
```

Exple2 :

```
ALTER TABLE Client
ADD CONSTRAINT ck_client_Ville CHECK (Ville IN('Rabat','Casa')) ;
```

2- Suppression d'une contrainteExple1 :

```
ALTER TABLE Client
DROP CONSTRAINT un_client_NomCli ;
```

Exple2 :

```
ALTER TABLE Produit
DROP CONSTRAINT PRIMARY KEY CASCADE ;
```

L'option CASCADE permet de supprimer les dépendances associées.

A. AHMADI

Langage SQL

Chap. 1**Langage de définition de données****II- Contraintes d'intégrité**

(65)

3- Inhibition d'une contrainteExple1 :

```
ALTER TABLE Client
DISABLE CONSTRAINT ck_Ville ;
```

Exple2 :

```
ALTER TABLE Produit
DISABLE PRIMARY KEY [CASCADE] ;
```

4- Activation d'une contrainteExple1 :

```
ALTER TABLE Client
ENABLE CONSTRAINT ck_Ville ;
```

Exple2 :

```
ALTER TABLE Produit
ENABLE PRIMARY KEY ;
```

A. AHMADI

Langage SQL

Chap. 1**Langage de définition de données****II- Contraintes d'intégrité**

(66)

4- Activation d'une contrainte

La contrainte ne sera pas effectuée si une des lignes transgresse la contrainte d'intégrité. La clause **EXCEPTIONS** permet de récupérer les lignes qui la transgressent :

```
ALTER TABLE Client
ENABLE CONSTRAINT ck_Ville EXCEPTIONS INTO Rejets ;
```

En plus du contrôle, SQL insère dans une table de rejets les tuples qui transgessent la contrainte. La structure de cette table est :

```
CREATE TABLE Rejets (row1 ROWID, owner VARCHAR2(30), table_name VARCHAR2(30),
constraint VARCHAR2(30));
```

Pour sélectionner les lignes ayant transgressé la contrainte, on exécute la requête suivante :

```
Select client.* from Client, Rejets
where Client.rowid=Rejets.row1 ;
```

ROWID étant un pointeur sur l'emplacement de chacune des lignes.

A. AHMADI

Langage SQL

Chap. 1**Langage de définition de données****III- Modification et mise jour des données**

67

1- Création et remplissage d'une table à partir d'une autreExemple : Créez une table contenant uniquement les factures du client 'c001' :

```
CREATE TABLE FactC001 AS
SELECT * from Facture WHERE cocli='c001';
```

2- Modification de la structure d'une tableExemple1 : Ajouter une nouvelle colonne **Nbfact** à la table client :

```
ALTER TABLE client
ADD ( Nbfact NUMBER(4) NULL );
```

Exemple2 : Augmenter de **2** chiffres la longueur de la colonne **Montant** :

```
ALTER TABLE Facture
MODIFY ( Montant NUMBER(10,2) );
```

Remarque : en cas de diminution de la taille, il faut que les valeurs des montants déjà saisis aient une taille inférieure à la nouvelle taille, sinon la modification de la structure ne sera pas acceptée.

A. AHMADI

Langage SQL

Chap. 1**Langage de définition de données****III- Modification et mise jour des données**

68

3- Effacement d'une table

```
DROP TABLE <nom_table> ;
```

Exemple :

```
DROP TABLE Dept ;
```

4- Renommer une table

```
RENAME <ancien_nom> TO <nouveau_nom> ;
```

Exemple : RENAME Client to Client_Bis ;**5- Suppression d'une colonne**Exemple : Enlever la colonne ville de la table client :

```
CREATE TABLE Temp AS SELECT Cocli,Nomcli FROM Client ;
DROP TABLE Client ;
RENAME Temp to Client ;
ou bien ALTER TABLE client DROP COLUMN ville ;
```

6- Renommer une colonne

```
ALTER TABLE produit RENAME COLUMN libelle to DESIGN;
```

A. AHMADI

Langage SQL

Chap. 1 Langage de définition de données

III- Modification et mise jour des données

69

6- Insertion de données

Syntaxe: `INSERT INTO <nom_table>[<colonne1,...,ColonneN>] VALUES (Val1,..., ValN) ;`

Exemple1 : `INSERT INTO client VALUES ('c009', 'SAADI', 'Rabat') ;`

Exemple2 : `INSERT INTO client(cocli,ville) VALUES ('c011', 'Meknes') ;`

Le champ Nomcli ne doit pas être obligatoire dans ce cas.

Exemple3 : `INSERT INTO client VALUES ('c009', 'SAADI', '&vil') ;`

Exemple4 : Ajouter dans Produit les produits de la table NouvProd valant plus de 100DH :

```
INSERT INTO Produit
SELECT * FROM NouvProd WHERE Npu >100 ;
```

7- Suppression de lignes

Exemple1 : Supprimer tous les produits dont le prix de vente est inférieur à 50 DH :

```
DELETE FROM Produit
WHERE Pu<50 ;
```

Exemple2 : Supprimer toutes les factures antérieures au 01/01/95 :

```
DELETE FROM Facture WHERE Datefact<'01-01-95' ;
```

A. AHMADI

Langage SQL

Chap. 1 Langage de définition de données

III- Modification et mise à jour des données

70

8- Modification de valeurs

Format1 : `UPDATE <nom_table>
SET <col1>=<expression1> [, ... , <colN>=<expressionN>]
[WHERE <condition>] ;`

Format2 : `UPDATE <nom_table>
SET (<col1>, ..., <colN>)=<sous-requête>
[WHERE <condition>] ;`

Exemple1: Augmenter les prix de 10% :

```
UPDATE Produit SET Pu=Pu*1.1 ;
```

Exemple2 :Modifier le prix unitaire du produit 'P03' en 50 DH ;

```
UPDATE Produit SET Pu=50 WHERE copro='p03' ;
```

Exemple3 : Augmenter de 5 DH, tous les prix moins de 10 DH :

```
UPDATE Produit SET Pu=Pu+5 WHERE Pu<10 ;
```

Exemple4 : Mettre le produit 'P04' au même prix que le produit 'P01' :

```
UPDATE Produit
SET Pu= (SELECT Pu FROM Produit WHERE copro='p01')
WHERE copro='p04' ;
```

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

1- Introduction

(71)

- LID Permet d'écrire des requêtes de consultation des données de la base ;
- Le résultat est fourni sous forme de tableau avec le nom des colonnes en en-tête ;
- Le LID utilise l'instruction SELECT dont la syntaxe (restreinte) est :

```
SELECT [DISTINCT/ALL] */<colonne>/<expression>
FROM <nom_table>
[WHERE <condition>]
[GROUP BY <expression>, ...]
[HAVING <condition>]
[ORDER BY <colonne>/<expression> [ASC/DESC] ...];
```

- *DISTINCT/ALL*: permet l'élimination ou la conservation des tuples (enregistrements) identiques dans le résultat ;
- *<expression>*: est une expression arithmétique, résultat de calcul ou de fonction ;
- *WHERE <condition>* : permet de sélectionner les tuples vérifiant la condition *<condition>* ;
- *GROUP BY* : Permet la constitution de groupes de tuples ayant des valeurs identiques dans un champ donné ;
- *HAVING <condition>*: Permet d'afficher uniquement les groupes de tuples vérifiant la condition *<condition>* ;
- *ORDER BY* : Permet de trier (par ordre croissant ou décroissant) les tuples selon un ou plusieurs champs ;

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

2- Affichage

(72)

- On peut présenter le résultat en changeant les noms de colonnes et/ou des expressions :

Exemple 1 : SELECT cocli Code_Client, Nomcli Nom_Client FROM Client :

Exemple 2 : SELECT Salaire+Comm Salaire_Global FROM emp;

3- Conditions

- Syntaxe des conditions : *<Opérande> <Opérateur> <Opérande>*
- **Opérande** :
 - Un nom de colonne;
 - Une valeur (numérique, alphanumérique ou date) ;
 - Une expression : (arithmétique et/ou utilisant des fonctions) ;
 - Une sous-requête (fournissant une valeur ou une liste de valeurs compatibles);
- **Opérateur** :
 - Arithmétique : + - * /
 - Concaténation de chaînes : ||
 - Comparaison : = <> != < > <= >=

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

3- Conditions

(73)

Exemples :

1- Afficher tous les attributs de tous les clients :

```
Select * from client;
```

2- le nom et la ville de tous les clients :

```
Select Nomcli, Ville from client;
```

3- Afficher tous les clients de la ville de Tanger :

```
Select * from client Where ville ='Tanger';
```

4- Afficher tous les produits dont le libellé commence par A :

```
Select * from Produit where Libelle LIKE 'A%' ; (like 'A_I%')
```

5- Afficher tous les produits dont le prix unitaire est supérieur ou égal à 150 DH ;

```
Select * from Produit Where PU >= 150 ;
```

6- Afficher tous les produits dont le libellé commence par A et le prix est inférieur à 300 DH :

```
Select * from Produits where Libelle LIKE 'A%' AND PU<300 ;
```

7- Afficher le code, le libelle et le prix unitaire de tous les produits en concaténant le code et le libellé :

```
Select Copro||' '|| Libelle "Identifiant du produit" , PU From Produit;
```

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

3- Conditions

(74)

- Opérateurs spécifiques :

- **[NOT] IN** : le 1er opérande doit (ou ne doit pas) appartenir à la liste des valeurs mentionnées en 2ème opérande.

Exemple : `Select * from client where UPPER(Ville) IN ('RABAT' , 'MEKNES' , 'TANGER') ;`

- **ANY, SOME** : la condition doit être satisfaite pour au moins une des valeurs du 2ème opérande . En général, les valeurs de ce dernier sont retournées par une sous-requête;

Exemple 1: `Select * from Produit where PU < ANY (100,335,123) ;`

Exemple 2: `Select * from Produit where PU < ANY (Select PU from Produit where LIBELLE LIKE 'A%');`

- **ALL** : La condition est vraie si elle est vérifiée pour toutes les valeurs fournies par le 2ème opérande

Exemple : `Select * from Produit where PU > ALL (Select PU from Produit where LIBELLE LIKE 'c%');`

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

3- Conditions

(75)

- **[NOT] BETWEEN ... AND ...** : La condition est vraie si le 1^{er} opérande est (n'est pas) compris entre les 2 valeurs données (bornes comprises).

Exemple : `SELECT * FROM facture where Datefact BETWEEN '01-03-14' AND '31-12-15' ;`

- **[NOT] EXISTS <sous-requêtes>** : la condition est vraie si la sous-requête retourne (ou ne retourne pas) au moins un tuple.

- **[NOT] LIKE** : La condition est vraie si le 1^{er} opérande contient (ou ne contient pas) un ensemble de caractères définis par les jokers _ et %. Le symbol "_" représente un caractère quelconque, et le symbol "%" représente une chaîne de caractères quelconque.

Exemple : afficher les clients dont le nom contient la lettre "a" à la 3^{ème} position

`Select * from Client where NomCli LIKE '_a%' ;`

- **IS [NOT] NULL** : la condition est vraie si la valeur du 1^{er} opérande est (n'est pas) NULL (non renseignée).

Exemple : `Select * from produit where LIBELLE IS NULL ;`

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

3- Conditions

(76)

- Opérateurs ensemblistes : UNION, INTERSECT, MINUS

- Afficher les codes des produits n'ayant jamais été commandés :

`(select copro from Produit) MINUS (select copro from Detail) ;`

- Afficher les codes et les noms des clients habitant Rabat ou Meknès :

`(select cocli,nomcli from Client Where UPPER(ville)='RABAT')
UNION (select cocli,nomcli from Client Where UPPER(ville)='MEKNES') ;`

Cette requête est équivalente à :

`select cocli,nomcli from Client Where UPPER(ville) IN ('RABAT', 'MEKNES');`

Remarque : pour utiliser convenablement les opérateurs ensemblistes, il faut que les 2 requêtes retournent le même nombre de colonnes, avec les mêmes types et dans le même ordre.

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

4- Fonctions SQL

(77)

Fonctions numériques :

ABS(n)	Valeur absolue : ABS(-32.5)=32.5
CEIL(n)	Plus petit entier relatif supérieur ou égal : CEIL(31.5)=32 CEIL(-31.5)=-31
FLOOR(n)	Plus grand entier relatif inférieur ou égal : FLOOR(31.5)=31 FLOOR(-31.5)=-32
MOD(m,n)	Reste de la division de m par n : MOD(35,4)=3
POWER(m,n)	m puissance n : POWER(4,3)=84
SIGN(n)	Indique le signe de n : SIGN(0)=0 SIGN(-5)=-1 SIGN(5)=1
SQRT(n)	Racine carrée de n : SQRT(9)=3 SQRT(-9)=NULL
ROUND(n)	Arrondi de n à 100 (Partie entière) : ROUND(15.3)=15 ROUND(15.5)=16
ROUND(n,m)	Arrondi de n à 10 ^{-m} : ROUND(1500,-3)=20 ROUND(1499,-3)=1000 ROUND(1.551,1)=1.6 ROUND(1.551,2)=1.55
TRUNC(n)	n tronqué à 100 (Partie entière) : TRUNC(15.3)=15
TRUNC(n,m)	n tronqué à 10 ^{-m} : TRUNC(1500,-3)=1000 TRUNC(1499,-3)=1000 TRUNC(1.551,1)=1.5 TRUNC(1.551,2)=1.5

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

4- Fonctions SQL

(78)

Fonctions de groupe:

AVG(expr)	Moyenne de toutes les valeurs de expr
COUNT(*)	Nombre d'enregistrements retournés par la sélection.
COUNT(expr)	Nombre d'enregistrements retournés par la sélection, pour lesquels expr n'a pas une valeur NULL..
MAX(expr)	Valeur maximale de toutes les valeurs de expr
MIN(expr)	Valeur minimale de toutes les valeurs de expr
STDDEV(expr)	Ecart type de toutes les valeurs de expr
SUM(expr)	Somme de toutes les valeurs de expr
VARIANCE(expr)	Variance de toutes les valeurs de expr

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

4- Fonctions SQL

79

Fonctions chaîne de caractères:

ASCII(char)	Donne la valeur ASCII ou EBCDIC du premier caractère de la chaîne de caractères char. ASCII('(parenthèses')=40 ASCII(' ')=32
INSTR(char, char1[,n,[m]])	Recherche dans la chaîne de caractères char la position de la chaîne de caractères char1. Si n est précisé, la recherche se fait à partir du rang n. Si m est précisé, la recherche donne la position de la m-ème occurrence de char2 dans char1. Si m est précisé, il est obligatoire de préciser n. INSTR('Contentement','t',1,5)=0 INSTR('Détermination','i',8)=11
LENGTH(char)	Donne la longueur de la chaîne de caractères char. LENGTH('Intégrité')=9
CHR(n)	Caractère ayant la valeur ASCII ou EBCDIC de n CHR(65) = 'A' CHR(97)='a' CHR(40)=''
INITCAP(char)	La première lettre de chaque mot de la chaîne de caractères est mise en majuscule, toutes les autres lettres sont mises en minuscules. INITCAP('mAdamE de sTaEl')='Madame De Stael'
LOWER(char)	Toutes les lettres sont mises en minuscules LOWER('ConFiAnce')='confiance'
UPPER(char)	Toutes les lettres sont mises en majuscules UPPER('ConFiAnce')='CONFIAНCE'
LPAD(char1,n,[char2])	Fait précéder la chaîne de caractères char1 de la chaîne de caractères char2 (espace si char2 non spécifié) jusqu'à la longueur n. Si n est inférieur à la longueur de char1, tronque char1 à la longueur n. LPAD('Juin', 9,'=')= '=====Juin' LPAD('Juin',2,'=')='Ju'

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

4- Fonctions SQL

80

Fonctions chaîne de caractères:

RPAD(char1,n,[char2])	Fait suivre la chaîne de caractères char1 du caractère char2 (espace si char2 non spécifié) jusqu'à la longueur n. Si n est inférieur à la longueur de char1, tronque char à la longueur n. RPAD('Juin', 9,'=')= 'Juin===== RPAD('Juin',2,'=')='Ju'
LTRIM(char1[,char2])	Supprime du début de la chaîne de caractères char1 les caractères présents dans la chaîne de caractère char2 jusqu'à ce que plus aucun caractère de char2 ne débute char1. LTRIM('=====Juin','=')= 'Juin' TRIM('Juin','Ju')='in' LTRIM('JJuuJuuin','Ju')='ain'
RTRIM(char1[,char2])	Supprime de la fin de la chaîne de caractères char1 les caractères présents dans la chaîne de caractère char2 jusqu'à ce que plus aucun caractère de char2 ne termine char1. RTRIM('Juin=====','=')= 'Juin' TRIM('Juin','in')='Ju' RTRIM('Juiniinnii','in')='Ju'
REPLACE(char, char1, char2)	Remplace dans la chaîne de caractères char la chaîne de caractères char1 par la chaîne de caractères char2. REPLACE('1.235.256.45','.')='1 235 256.45' REPLACE('ABBABAAAB','AB','C')="CBCAAC"
SUBSTR(char1,n[,m])	Extrait de la chaîne de caractères char1, les caractères situés à partir du rang n jusqu'à la longueur m, ou jusqu'à la fin si m non spécifié. SUBSTR('Respect',4)=pect' SUBSTR('Respect',4,2)=pe'
TRANSLATE(char, char1 , char2)	Remplace dans la chaîne de caractères char les caractères présents dans la chaîne de caractères char1 par les caractères de même rang présents dans la chaîne de caractères char2. REPLACE('1.235.256.45','.',',')='1 235 256.45' REPLACE('Satisfaction','sa','**')='St*fction'

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

4- Fonctions SQL

(81)

Fonctions date:

ADD_MONTHS(date, n)	Ajout de n mois à la date. ADD_MONTHS('01-DEC-93')='01-JAN-94'
LAST_DAY(date)	Indique le dernier jour du mois de date LAST_DAY('15-FEV-93')='28-FEV-93'
MONTHS_BETWEEN(date1, date2)	Nombre de mois entre date1 et date2. La partie décimale est obtenue en divisant le nombre de jours par 31. MONTHS_BETWEEN('26-JUN-90','25-DEC-93')=40,967742 MONTHS_BETWEEN('26-JUN-90','26-DEC-89')=-6
NEXT_DAY(d,j)	Date postérieure à la date d du jour j. NEXT_DAY('12-DEC-93', 'MERCREDI')='15-DEC-93'
ROUND(date[,format])	Arrondi de la date au format spécifié. Si format non spécifié, arrondi au jour le plus proche. ROUND(TO_DATE('30-JUN-93','Y'))='01-JAN-93' ROUND(TO_DATE('01-JUL-93','Y'))='01-JAN-94' ROUND(TO_DATE('15-12-93 12:00:00','DD-MM-YY HH:MI:SS')) = '16-DEC-93' ROUND(TO_DATE('15-12-93 11:59:59','DD-MM-YY HH:MI:SS')) = '15-DEC-93'
TRUNC(date[,format])	Date tronquée au format spécifié. Si format non spécifié, arrondi au jour. TRUNC(TO_DATE('01-JUL-93','Y'))='01-JAN-93' TRUNC(TO_DATE('15-12-93 12:00:00','DD-MM-YY HH:MI:SS')) ='15-DEC-93' TRUNC(TO_DATE('15-12-93','MM'))='01-DEC-93'

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

5- Exemples de requêtes

(82)

i- Fonctions de groupe

1- Afficher le prix le plus élevé de tous les produits :

`select MAX(PU) from produit;`

2- Afficher le chiffre d'affaires de chaque client :

`select cocli,SUM(Montant) from facture GROUP BY cocli;`

3- Afficher le chiffre d'affaires global de l'entreprise :

`select SUM(Montant) FROM Facture;`

4- Afficher la quantité totale vendue de chaque produit :

`select copro, sum(qte) AS Quantite_Totale From detail group by copro Order by 2 DESC;`

5- Afficher le nombre total de lignes de facture :

`select COUNT(*) FROM Facture;`

6- Afficher le nombre de produits achetés dans chaque facture :

`Select Nufact, Count(*) from Detail Group By Nufact;`

7- Afficher le nombre de factures par client :

`Select cocli,count(*) From Facture
Group by cocli
Order by 2 DESC,cocli;`

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

5- Exemples de requêtes

(83)

i- Fonctions de groupe

8- Afficher le nombre de clients ayant été facturés :

```
select COUNT(COUNT(*)) FROM Facture GROUP BY Cocli;
```

9- Afficher les clients ayant une seule facture :

```
Select cocli From Facture Group by Cocli HAVING COUNT(*)=1;
```

10- Afficher les clients ayant un chiffre d'affaires >600 DH;

```
Select Cocli, SUM(Montant) AS Chiffre_Affaires FROM Facture
GROUP BY Cocli
HAVING SUM(Montant)>600;
```

11- Afficher les clients qui ont un chiffre d'affaires supérieur à 500 DH depuis le 1er novembre 94:

```
Select Cocli, SUM(Montant) From Facture Where Datefact>='01-11-94'
GROUP BY Cocli
HAVING SUM(Montant) > 500;
```

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

5- Exemples de requêtes

(84)

ii- Requêtes imbriquées

Une condition de SELECT utilise le résultat d'une sous-requête. On distingue 2 cas :

- Requêtes *non Synchronisées* : les sous-requêtes sont exécutées en premier. On commence par la requête la plus interne.

- Requêtes *Synchronisées* : les sous-requêtes sont exécutées pour chaque valeur de la requête principale.

a- Requêtes non-synchronisées

1- Afficher le nom du client correspondant à la facture N° 3 :

```
select nomcli from client where cocli=(select cocli from facture where Nufact=3);
```

2- Afficher les noms des clients qui n'ont jamais été facturés :

```
select nomcli from client where cocli NOT IN (select cocli from Facture);
```

3- Afficher les numéros et les dates des factures du client 'Mounir' :

```
select Nufact, Datefact from facture where cocli IN (select cocli from client
where UPPER(nomcli)= 'MOUNIR') ;
```

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

5- Exemples de requêtes

85

ii- Requêtes imbriquées

a- Requêtes non-synchronisées

4- Afficher le(s) nom(s) du produit le plus cher :

```
select Libelle from produit where pu=(select MAX(pu) From Produit) ;
```

5- Afficher les codes et les noms des clients habitant la même ville que le client 'Mounir'

```
select cocli,nomcli from client
where UPPER(ville) IN (select UPPER(ville) from Client where
    UPPER(nomcli)='MOUNIR')
AND UPPER(nomcli) != 'MOUNIR' ;
```

6- Afficher les codes, les libellés et les prix des produits facturés le 10-08-94 :

```
select * from Produit
where copro IN (select DISTINCT copro From Detail
    Where Nufact IN (select Nufact From Facture
        where Datefact='10-08-94'));
```

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

5- Exemples de requêtes

86

ii- Requêtes imbriquées

b- Requêtes synchronisées

1- Nom du client correspondant à la facture numéro 3 :

```
SELECT Nomcli from Client
WHERE EXISTS (SELECT * FROM Facture WHERE Cocli=Client.Cocli AND Nufact=3);
```

2- Noms des clients qui n'ont jamais été facturés :

```
SELECT nomcli from Client
WHERE NOT EXISTS (SELECT * from Facture WHERE Cocli=Client.Cocli);
```

3- Nom du produit le plus cher :

```
SELECT Libelle FROM Produit
WHERE Pu>= ALL (SELECT Pu from Produit);
```

4- Codes et noms des clients (autres que Rachidi) qui habitent la même ville que le client "Rachidi" (supposé unique) :

```
SELECT Cocli, Nomcli From Client X
WHERE EXISTS (SELECT * FROM Client Y WHERE NomCli= 'Rachidi' AND X.ville=Y.ville)
AND Nomcli!='Rachidi'
```

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

5- Exemples de requêtes

(87)

ii- Requêtes imbriquées

b- Requêtes synchronisées

5-Codes, libellés et prix des produits facturés le 15-12-94 :

```
SELECT * FROM Produit
WHERE EXISTS (SELECT * FROM Detail WHERE Copro=Produit.Copro
              AND EXISTS (SELECT * FROM Facture WHERE Nufact=
Detail.Nufact AND Datefact='15-12-94')) ;
```

A. AHMADI

Langage SQL

Chap. 2 Langage d'interrogation de données(LID)

5- Exemples de requêtes

(88)

iii- Jointures

Les jointures permettent de ramener des données à partir de plusieurs tables (liées) en évitant les imbrications de requêtes.

1- Afficher les produits qui n'ont jamais été facturés :

```
SELECT Produit.Copro, libelle From Produit,Detail
WHERE Produit.Copro=Detail.copro (+)
AND Detail.Copro IS NULL ;
```

2- Nom du client correspondant à la facture numéro 3 :

```
SELECT Nomcli From Client,Facture
WHERE client.Cocli=Facture.Cocli
AND Nufact=3 ;
```

3- Nom des clients qui n'ont jamais été facturés :

```
SELECT Nomcli From Client, Facture
WHERE client.cocli=Facture.cocli (+)
AND Nufact IS NULL ;
```

4- Numéros et Dates des factures du client 'Rachidi'

```
SELECT Nufact,Datefact From Facture, Client
WHERE Facture.cocli=Client.cocli
AND Nomcli='Rachidi' ;
```

A. AHMADI

Langage SQL