MCD,MLD et SQL

Par Ndeye Saly Dione, apprenante à la Sonatel Academy

I) Introduction

Le **système d'information (SI) est un élément central** d'une entreprise ou d'une organisation. Il permet aux différents acteurs de véhiculer des informations et de communiquer grâce à un ensemble de ressources matérielles, humaines et logicielles. Un SI permet de créer, collecter, stocker, traiter, modifier des informations sous divers formats.

L'**objectif d'un SI** est de restituer une information à la bonne personne et au bon moment sous le format approprié.

Les fonctions d'un système d'information

Il existe donc 4 fonctions principales d'un SI :

- **Collecter** : c'est à partir de là que naît la donnée, qu'on acquière les informations provenant de l'environnement interne ou externe à l'entreprise.
- **Stocker** : dès que l'information est acquise, le système d'information la conserve. Elle doit pouvoir être disponible et doit pouvoir être conservée dans le temps.
- **Transformer/traiter** : cette phase permet de transformer l'information et choisir le support adapté pour traiter l'information. Ici ont construit de nouvelles informations en modifiant le fond ou la forme.
- **Diffuser :** le SI transmet ensuite l'information dans son environnement interne ou externe.

Maintenant pour pouvoir stocker les données et effectuer tout le travail énuméré précédemment l'entreprise a besoin d'une base de données. Cette dernière est un ensemble structuré de données élémentaires enregistrées sur des supports accessibles par l'ordinateur. Pour pouvoir stocker les informations dans la base de données, l'on doit pouvoir la conceptualiser, de représenter les données d'une fonction compréhensible par l'humain, de tisser les liens existants entre elles. Pour ce fait nous allons parler de méthodes Merise, UML et faire la comparaison entre ces deux méthodes.

II) Notion Analyse et Conception

Un projet informatique nécessite une phase d'analyse et une phase de conception.

Une méthode d'analyse et de conception a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client.

Dans **la phase d'analyse**, on cherche d'abord à bien comprendre et à décrire de façon précise les besoins des utilisateurs ou des clients. Que souhaitent-ils faire avec le logiciel ? Quelles fonctionnalités veulent-ils ? Pour quel usage ? Comment l'action devrait-elle fonctionner ? C'est ce qu'on appelle « **l'analyse des besoins** ». Après validation de notre compréhension du besoin, nous imaginons la solution. C'est la partie **analyse de la solution.**

1) Merise

Merise (Méthode D'Etude et de Réalisation Informatique pour les Systèmes d'Entreprises) est une méthode qui permet de traduire un modèle en une base de données. Cela permet de mieux structurer notre base de données avant de la créer. Dans cette méthode nous avons trois modèles : Modèle Conceptuel de Données (MCD), Modèle Logique de Données (MLD) et le Modèle Physique de Données (MPD).

2) UML

UML (Unified Modeling Language) qui se traduit en français langage de modalisation unifié. La notation UML est un **langage visuel** constitué d'un ensemble de schémas, appelés des **diagrammes**, qui donnent chacun une vision différente du projet à traiter. UML nous fournit donc des diagrammes pour **représenter** le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc.

Réaliser ces diagrammes revient donc à **modéliser les besoins** du logiciel à développer. Parmi ces diagrammes, nous avons le diagramme de Contexte, le diagramme de cas d'utilisation, le diagramme d'activité…

Dans **la phase de conception**, on apporte plus de détails à la solution et on cherche à clarifier des aspects techniques, tels que l'installation des différentes parties logicielles à installer sur du matériel.

Pour mener à bien ces deux aspects : analyse et conception nous avons les méthodes Merise et UML qui sont toutes deux des méthodes d'analyses et de conception de projet.

3) Différence entre Merise et UML

La différence qu'il existe entre ces deux méthodes réside dans le fait que Merise est une méthode pour la modélisation de base de données rationnelle. Une base de données relationnelle est une base de données dans laquelle les données sont organisées dans de tableaux appelés relations ou tables.

Merise	UML
Méthode d'analyse et de conception de système d'information	Langage de représentation d'un système d'information
Méthode de modélisation des données orientée bases de données rationnelles	Système de notation orientée objet
Rationnel	Objet
Franco-français	International
Schéma directeur, étude préalable, étude détaillée et la réalisation	Langage de modélisation des systèmes standard, qui utilise de diagrammes pour représenter chaque aspect d'un système : statique, dynamique… en s'appuyant sur la notion d'oriente objets
Plus adaptée à une approche théorique	Plus orientée vers la conception
Du « bottom up » de la base de données ver le code	Du « up down » du modèle vers la base de données

III) ***Concepts Analyse et Conception***

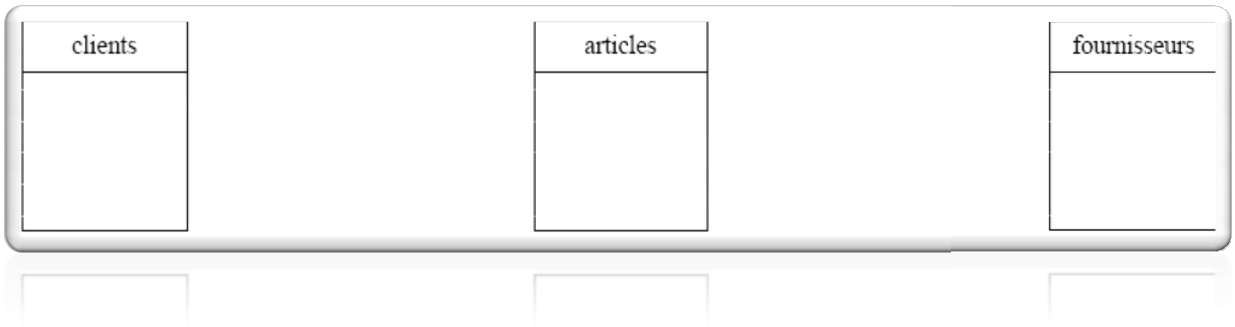
A. MCD

La modélisation est une étape fondamentale de la conception de la BD dans la mesure où, d'une part, on y détermine le contenu de la BD et, d'autre part, on y définit la nature des relations entre les concepts principaux.

Le modèle conceptuel de données est une représentation des données sous la forme de tables appelée entités.

1. Les entités

Une **Entité** est une population d'individus homogènes. Par exemple, les produits ou les articles vendus par une entreprise peuvent être regroupés dans une même entité articles, car d'un article a l'autre, les informations ne changent pas de nature. Dans l'exemple ci-dessous on a trois entités : clients, articles, fournisseurs.

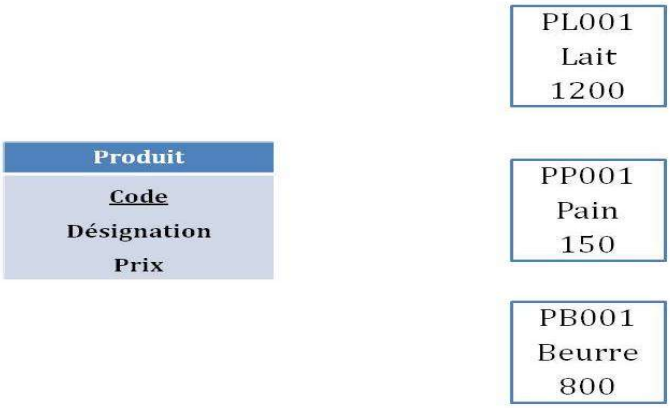


2. Les attributs

L'attribut ou la propriété est une information élémentaire, c'est-à-dire non déductible d'autres informations et qui présente un intérêt pour le domaine étudié. Par exemple pour l'entité articles on peut avoir les attributs idProduit, prixUnitaire…

3. Les occurrences

Créer une *occurrence* d'une entité consiste à donner une valeur réelle à chaque propriété de l'entité. Par exemple pour l'entité Produit on peut avoir ces trois occurrences suivantes (image ci-dessous). Dans cet exemple l'entité produit a trois attributs : Code, Désignation, Prix.



4. Les Cardinalités

Les cardinalités permettent de caractériser le lien qui existe entre une entité et l'association à laquelle elle est reliée.

La cardinalité est composée d'un couple comportant une borne minimale et une borne maximale :

- La borne minimale (généralement 0 ou 1) décrit le nombre minimum de fois qu'une entité peut participer à une association.
- la borne maximale (généralement 1 ou n) décrit le nombre maximum de fois qu'une entité peut participer à une association.

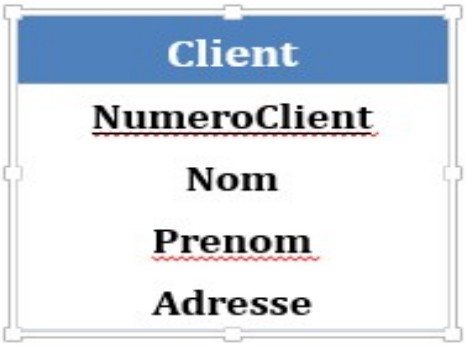
Une association (appelée aussi parfois relation) est un lien sémantique entre des entités. Dans l'exemple ci-dessous dans une commande on peut avoir 1 à plusieurs produits. Un produit peut être contenu dans aucun ou plusieurs commandes. L'association qui relie les deux entités Commande et produit est Contenir.

B. MLD

Le **modèle logique** des **données** consiste à décrire la structure de **données** utilisée sans faire référence à un langage de programmation. Il découle du modèle conceptuel de données. Dans ce modèle on a :

- Les tables qui représentaient les entités dans le MCD
- Les champs qui représentaient les attributs dans le MCD
- Les clés primaires et étrangères
- Des flèches qui matérialisent les relations entre les tables

Dans l'exemple ci-dessous on a la table Client avec ses champs NumeroClient, Nom, Prenom, Adresse.



Exemple de contenu dans la table Client :

NumeroClient	Nom	Prenom	Adresse
CL001	Diouf	Matar	PA U18 Villa ZZZ DK
CL002	Diallo	Yacine	7 Rue BD St-Louis
...	...	...	...

- **Les clés primaires**

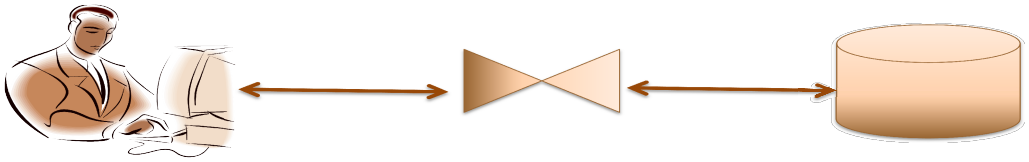
Les lignes d'une table sont uniques, i.e. qu'il existe au moins une ou un ensemble de colonnes qui sert à identifier les lignes : il s'agit de la clé primaire de la table. Les champs de la table dans le MLD qui constituent la clé primaire sont soulignés et correspondent aux identifiants des entités ou associations dans le MCD.

- **Les clés étrangères**

Une entité est **faible** si sa cardinalité maximale est de 1.
Une entité est **forte** si sa cardinalité maximale est de N.
Lorsqu'une entité est faible, elle absorbe l'identifiant de l'entité forte et dans ce cas, une nouvelle propriété apparaît dans l'entité faible que nous appellerons **clé étrangère.** On sait qu'un client peut passer une à plusieurs commandes donc (1,n) coté client mais une commande ne peut être passer que par un et un seul client donc (1,1) coté commande.

IV) SQL

Un Système de Gestion de Bases de Données (SGBD) est un logiciel qui permet de gérer les données stockées dans une base de données.



USER SGBD Base de données

Un seul langage standard utilisé par la plupart des SGBDR (système de gestion de base de données relationnelle) Structured Query Language (SQL).

B. Langage de définition de données

- **CREATE**

La commande CREATE permet de créer une base de données ou une table. Dans notre exemple nous allons créer une base de données qui s'appelle restaurant avec la commande :

CREATE DATABASE restaurant.

Ensuite nous allons créer une table dans cette base de données une table « plat » avec les champs « idPlat » qui va représenter la clé primaire de la table, « nomPlat », « tarif ».

```
CREATE TABLE plat(
        id VARCHAR(255) NOT NULL PRIMARY KEY,
        nomPlat VARCHAR(255) NOT NULL,
        tarif INT(255) NOT NULL
)
```

VARCHAR (chaine de caractères) et INT (entier) représentent les types des champs. Les 255 représentent le nombre maximum de caractères.La valeur NOT NULL sert à indiquer l'attribut ne peut pas avoir de valeur nulle.

- **DROP**

Pour supprimer notre base de données « restaurant » nous allons utiliser la ligne de commande **DROP DATABASE restaurant** ou pour supprimer notre table la ligne de commande **DROP TABLE plat**

- **ALTER**

  - Cette commande permet de d'ajouter un champ à une table. Nous allons ajouter à notre table « plat » le champ « type » avec la ligne de commande

    ALTER TABLE plat ADD type VARCHAR(255) NOT NULL

  - Supprimer un champ

    Nous allons supprimer le champ « tarif » de notre table «plat» avec la ligne de commande :

    ALTER TABLE plat DROP tarif

- **RENAME**

Cette commande permet de changer le nom d'un champ. Nous allons changer le nom de champs « nomPlat » en « nom » avec les lignes de commande :

ALTER TABLE plat

RENAME COLUMN nomplat TO nom

- **TRUNCATE**

Cette commande permet de supprimer et de réinitialiser les lignes d'une table. Si nous voulions supprimer notre table « plat » nous aurions utilisé la ligne de commande suivante :

TRUNCATE TABLE plat

C. Langage de manipulation de données (LMD)

- **INSERT**

Cette commande permet d'insérer des valeurs dans une table correspondantes à chaque champ de cette table. Nous créer le plat Thiebou-Dieun. Pour l'id du plat on lui donne la valeur de 1P, nom « Thiebou-Dieun ». Nous fixons le tarif à 1000 FCFA. Pour ce faire nous allons exécuter la ligne de commande :

INSERT INTO plat

VALUES("1P", "Thiebou-Dieun",1000)

- **UPDATE**

Permet de faire des modifications dans une table suivant une condition ou pas. Dans notre exemple nous allons modifier le plat Thiebou-Dieun en Yassa et le prix à 1500 FCFA suivant l'id avec la ligne de commande suivante :

UPDATE plat SET nom= « Yassa », tarif=1500 WHERE id=« 1P »

➢ Langage d'interrogation de données (LID)

○ SELECT

Cette commande permet de faire une sélection. Dans notre exemple nous allons sélectionner tous les plats dont le tarif est supérieur à 2000 FCFA avec la ligne de commande :

SELECT nom FROM plat WHERE tarif>2000

○ Jointure

Cela permet de joindre deux tables. Soient les tables R1 et R2 :

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R2

#X	Y
10	Echo
20	Fox
30	Golf

La ligne de commande ci-dessous :

SELECT * FROM R1 INNER JOIN R2 ON R1.C = R2.X

Va joindre les deux tables colonnes par colonnes si et seulement si la valeur C de R1 est égale à la valeur X de R2 dans les deux colonnes. Le résultat donne :

R

A	B	C	X	Y
1	Alpha	10	10	Echo
2	Bravo	10	10	Echo
3	Charlie	20	20	Fox

○ Projection

La projection sélectionne certaines colonnes d'un tableau.

Soit la table R1 suivante :

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

La projection (R1, A, C) qui se traduit par la ligne de commande :

SELECT A, C FROM R1

Donne le résultat suivant :

A	C
1	10
2	10
3	20
4	

- **Alias (AS)**

Il est possible de redéfinir le nom des propriétés de la relation résultat. Par exemple :

SELECT nom as nm, tarif as prix FROM plat

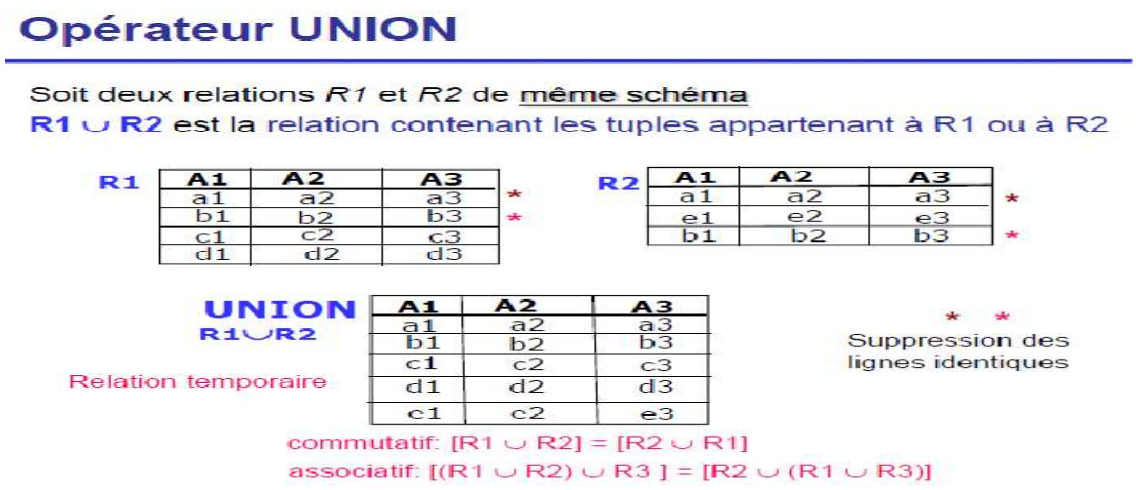
- **Commande**
  - **Union**

C'est une commande qui permet de concaténer les résultats de 2 requêtes ou plus. Par exemple :

SELET * FROM table1

UNION

SELECT * FROM table2



- **GROUP BY**

Cette commande donne la possibilité de regrouper plusieurs lignes d'une même requête. La syntaxe est la suivante :

SELECT list-exp1

FROM nomTable

GROUP BY list-exp2

Les expressions de list-exp1 doivent être des expressions formées uniquement d'expressions de list-exp2.

- **ORDER BY**

Cette commande permet de faire le tri dans tableau suivant un ordre quelconque. Dans notre base de données nous allons trier nos plats par :

Ordre croissant des prix:

SELECT nom FROM plat ORDER BY prix ASC

Ordre décroissant des prix:

SELECT nom FROM plat ORDER BY prix DESC

- **HAVING**

Cette commande permet d'exprimer une condition sur l'ensemble des lignes retournées.

Exemple 2 : Villes dont l'âge moyen des Fournisseurs est inférieur à 40 ans :

```
SELECT   VilleF , AVG(Age)  AS  AgeMoy
FROM     Fournisseur
GROUP BY VilleF
HAVING   AVG (Age) < 40:
```

- **Fonction**
  - **DISTINCT**

Cette commande permet d'éviter les doublons. Par exemple dans notre base de données nous avons la table Fournisseur et nous voulons savoir les différentes villes où ils sont repartis

SELECT DISTINCT(villeF) FROM Fournisseur

- **COUNT**

Cette commande permet de connaitre le nombre total d'éléments qu'on a dans une table. Par exemple, le nombre total de plats :

SELECT COUNT(*) FROM plat

- **LIMIT**

La clause LIMIT est à utiliser dans une requête SQL pour spécifier le nombre maximum de résultats que l'ont souhaite obtenir. Par exemple on l'on veut afficher 15 plats de notre table plat :

SELECT * FROM plat LIMIT 15

- **LIKE**

- La sélection simple
  - Opérateur de comparaison approximative **LIKE** : <chaine> LIKE '<motif>'
  - où **<motif>** est une suite de caractères contenant éventuellement les symboles **%** (parfois *) et _
  - **%** (ou *) remplace n'importe quelle suite de caractères
  - _ remplace exactement un caractère

Par exemple on veut sélectionner les plats dont le nom commence par P

SELECT nom

FROM plat

WHERE nom LIKE 'D%'

- **IN & NOT IN**

C'est une méthode simple pour vérifier si une colonne est égale à une valeur OU une autre valeur OU une autre valeur et ainsi de suite, sans avoir à utiliser de multiple fois l'opérateur OR.

Par exemple on peut avoir les plats dont le tarif est soit 1000, 1500 ou 2000 FCFA :

SELECT * FROM plat WHERE tarif in (1000, 1500, 2000)

Si on veut sélectionner les plats dont le tarif n'est pas compris n'est ni 1000 ni 1500 ni 2000 FCFA.

SELECT * FROM plat WHERE tarif NOT IN (1000, 1500, 2000)

- **BETWEEN**

L'opérateur BETWEEN est utilisé dans une requête SQL pour sélectionner un intervalle de données dans une requête utilisant WHERE.

Par exemple on veut sélectionner les plats dont le tarif est compris entre 700 et 3000FCFA

SELECT * FROM plat WHERE tarif BETWEEN 700 AND 3000

- **AVG**

Cette fonction permet de calculer la valeur moyenne.

Exemple : La table « achat » représente toutes les ventes sur un site d'e-commerce, dans laquelle on enregistre le montant de l'achat, la date et le nom du client.

id	client	tarif	date
1	Pierre	102	2012-10-23
2	Simon	47	2012-10-27
3	Marie	18	2012-11-05
4	Marie	20	2012-11-14
5	Pierre	160	2012-12-03

Pour connaitre le montant moyen effectué par chaque client, il est possible d'utiliser une requête qui va utiliser :

GROUP BY pour regrouper les ventes des mêmes clients

La fonction AVG() pour calculer la moyenne des enregistrements

La requête sera :

SELECT client, AVG(tarif)

FROM achat

GROUP BY client

- **MIN**

Cette fonction permet de sélectionner le minimum parmi un ensemble de valeurs. Par exemple on veut savoir le plat le moins cher dans notre base de données restaurant :

SELECT MIN(tarif) FROM plat

- **MAX**

Cette fonction permet de sélectionner le maximum parmi un ensemble de valeurs. Par exemple on veut savoir le plat le plus cher dans notre base de données restaurant :

SELECT MAX(tarif) FROM plat

- **SUM**

Cette fonction permet de calculer la somme totale d'une colonne contenant des valeurs numériques. Cette fonction ne fonction que sur des colonnes de types numériques (INT, FLOAT …) et n'additionne pas les valeurs NULL.

Par exemple on veut connaitre la somme totale des plats de Yassa qui ont l'id 5P.

SELECT SUM(tarif) FROM plat WHERE id= « 5P »