ANALYSE ET CONCEPTION DE BASE DE DONNÉES

Piste de documentation sur MCD, MLD et SQL

I. Notion Analyse et Conception

En ingénierie, une **méthode d'analyse et de conception** est un procédé qui 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. Pour ce faire, on part d'un énoncé informel (le besoin tel qu'il est exprimé par le client, complété par des recherches d'informations auprès des experts du domaine fonctionnel, comme les futurs utilisateurs d'un logiciel), ainsi que de l'analyse de l'existant éventuel (c'est-à-dire la manière dont les processus à traiter par le système se déroulent actuellement chez le client).

La phase d'analyse permet de lister les résultats attendus, en termes de fonctionnalités, de performance, de robustesse, de maintenance, de sécurité, d'extensibilité, etc.

La phase de conception permet de décrire de manière non ambiguë, le plus souvent en utilisant un langage de modélisation, le fonctionnement futur du système, afin d'en faciliter la réalisation.

- **Merise**

Est une méthode d'analyse et de conception des systèmes d'information basée sur le principe de la séparation des données et des traitements. Elle possède plusieurs modèles qui sont répartis sur 3 niveaux (Le niveau conceptuel, le niveau logique ou organisationnel, le niveau physique).

- **Uml**

Le **Langage de Modélisation Unifié**, de l'anglais *Unified Modeling* UML (en anglais Unified Modeling Language ou « langage de modélisation unifié ») est un langage de

modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la ‘‘conception orientée objet ’’. Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sortes de systèmes ne se limitant pas au domaine informatique. Il est ‘accomplissement de la fusion de précédents langages de modélisation objet tels que **Booch, OMT, OOSE** respectivement issus des travaux de Grady Booch, James Rumbaugh et Ivar Jackobson. UML est à présent un standard défini par l'Objet Management Group (OMG). La dernière version diffusée par l'OMG est UML 2.3 depuis mai 2010.

UML est avant tout un support de communication performant, qui facilite la représentation et la compréhension de solutions objet :

- Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation de solutions.
- L'aspect formel de sa notation, limite les ambiguïtés et les incompréhensions.
- Son indépendance par rapport aux langages de programmation, aux domaines d'application et aux processus, en font un langage universel. Petit aparté : La notation graphique d'UML n'est que le support du langage. La véritable force d'UML, c'est qu'il repose sur un métamodèle. En d'autres termes : la puissance et l'intérêt d'UML, c'est qu'il normalise sémantique des concepts qu'il véhicule !

Qu'une association d'héritage entre deux classes soit représentée par une flèche terminée par un triangle ou un cercle, n'a que peu d'importance par rapport au sens que cela donne à votre modèle. La notation graphique est essentiellement guidée par des considérations esthétiques, même si elle a été pensée dans ses moindres détails.

Par contre, utiliser une relation d'héritage, reflète l'intention de donner à votre modèle un sens particulier. Un "bon" langage de modélisation doit permettre à n'importe qui de déchiffrer cette intention de manière non équivoque ! Il est donc primordial de s'accorder sur la sémantique des éléments de modélisation, bien avant de s'intéresser à la manière de les représenter. Le métamodèle UML apporte une solution à ce problème fondamental.

UML est donc bien plus qu'un simple outil qui permet de "dessiner" des représentations mentales... Il permet de parler un langage commun, normalisé mais accessible, car

visuel. Il représente un juste milieu entre langage mathématique et naturel, pas trop complexe mais suffisamment rigoureux, car basé sur un métamodèle.

Etude Comparative

Niveaux d'abstraction

- L'approche Merise : Le cycle d'abstraction permet de sérier les niveaux de préoccupations lors de la description ou de l'analyse du système. Les trois niveaux retenus correspondent à des degrés de stabilité et d'invariance de moins en moins élevés. Le niveau conceptuel, le niveau logique, le niveau physique.
  - L'approche UML propose différentes notions (cas d'utilisation, paquetage, classe, composant, nœud) et différents diagrammes pour modéliser le système aux différents niveaux d'abstraction.

Approche fonctionnelle

- L'approche Merise propose une approche descendante où le système réel est décomposé en activités, elles-mêmes déclinées en fonctions. Les fonctions sont composées de règles de gestion, elles-mêmes regroupées en opérations. Ces règles de gestion au niveau conceptuel génèrent des modules décomposés en modules plus simples et ainsi de suite jusqu'à obtenir des modules élémentaires... Les limites d'une telle approche résident dans le fait que les modules sont difficilement extensibles et exploitables pour de nouveaux systèmes. L'approche UML : Les fonctions cèdent la place aux cas d'utilisation qui permettent de situer les besoins de l'utilisateur dans le contexte réel. A chaque scénario correspond des diagrammes d'interaction entre les objets du système et non pas un diagramme de fonction.

Dualité des données -traitements

L'approche Merise propose de considérer le système réel selon deux points de vue : un point de vue statique (les données), un point de vue dynamique (les traitements). Il s'agit d'avoir une vision duale du système réel pour bénéficier de l'impression de relief qui en résulte, et donc consolider et valider le système final. L'approche UML : L'approche objet associe les informations et les traitements. De cette façon, elle assure un certain niveau de cohérence.

- Voici un tableau résumant l'étude comparative entre Merise et UML ci-dessous :

Merise	UML
méthode d'analyse et de conception de système d'information	langage de représentation d'un système d'information.
méthode de modélisation de données et traitements orienté bases de données relationnelles.	système de notation orienté objet.
relationnel	objet.
Franco-français	International
schéma directeur, étude préalable, étude détaillée et la réalisation.	langage de modélisation des systèmes standard, qui utilise des diagrammes pour représenter chaque aspect d'un systèmes ie: statique, dynamique,....en s'appuyant sur la notion d'orienté objet
plus adapté à une approche théorique	plus orientée vers la conception
du "bottom up" de la base de donnée vers le code	du "top down" du modèle vers la base de donnée.

II. Concepts Analyse et Conception

A) MCD

Le modèle conceptuel des données est une représentation statique du système d'information de l'entreprise qui met en évidence sa sémantique.

Il a pour but d'écrire de façon formelle les données qui seront utilisées par le système d'information.

Il s'agit donc d'une représentation des données, facilement compréhensible. Le formalisme adopté par la méthode Merise pour réaliser cette description est basé sur les concepts « entité-association ».

- **Les entités**

Une entité est un objet abstrait ou concret de l'univers du discours. Une entité peut être :

- Une personne (CLIENT)
- Un lieu (DEPOT, BUREAU, ATELIER, …)
- Un objet documentaire (LIVRE, OUVRAGE, DOSSIER, …)

Après avoir réalisé le dictionnaire de données, il faut regrouper ces données par paquet homogène.

Ces paquets représentent les entités.

Une entité est caractérisée par :

- Un identifiant
- Une suite d'information liée à cet identifiant.

- **Les attributs**

- Un attribut ou propriété est une donnée élémentaire susceptible de prendre une valeur. C'est le plus petit élément manipulé du système d'information et qui a un sens en lui-même. Une propriété peut être élémentaire ou calculée, simple ou composée.
- La note d'un élève est une propriété élémentaire, en revanche sa moyenne est une propriété calculée.
- L'âge de l'élève est une propriété simple alors que l'adresse complète de l'élève est une propriété composée. Le nom de la propriété est inscrit à l'intérieur de l'entité.

- **Les Occurrences**

L'occurrence d'une entité ou d'une association est le nombre de fois qu'apparaît l'entité (ou l'association) ayant des propriétés à valeurs distinctes.

Dans le cas d'une équipe de foot sans remplaçants, l'entité « joueurs » aurait 11 occurrences. Cette notion d'occurrence sert à estimer la taille de la base de données.

- **Cardinalités**

- La cardinalité d'une entité par rapport à une relation s'exprime par deux nombres appelés cardinalité minimale et cardinalité maximale.
- La cardinalité minimale, égale à 0 ou 1, est le nombre de fois minimum qu'une occurrence d'une entité participe aux occurrences de la relation. Si la cardinalité minimale est égale à 0, c'est qu'il existe parmi toutes les occurrences de l'entité, au moins une occurrence qui ne participe pas à la relation. Par exemple, dans une équipe de sport, tous les membres de

l'équipe ne participent pas forcément à un match. En revanche, si la cardinalité minimale est égale à 1, cela implique que toutes les occurrences d'une entité participent à toutes les occurrences de la relation. Dans notre exemple, cela se traduit par le fait qu'un joueur joue tous les matchs.

- La cardinalité maximale, égale à 1 ou n, indique le nombre de fois maximum qu'une occurrence de l'entité participe aux occurrences de la relation (n est équivalent à infini).

B) MLD

Qu'es ce que le MLDR ?

Le Modèle Logique de Données Relationnel, MLDR pour Merise est une représentation de l'ORGANISATION DES DONNEES tenant compte d'une technologie, ici la technologie relationnelle.

- **Les données y sont organisées en :**
- RELATIONS (ou TABLES RELATIONNELLES ou simplement tables), celles-ci étant décrites en terme d'ATTRIBUTS de la relation (ou COLONNES) et
- TUPLES ou n-uplets (ou LIGNES).

L'établissement du modèle logique fournit un résultat INVARIANT (élément stable) face

- Aux logiciels qui seront développés.

Au choix d'implantation physique des données (un système de gestion de bases de données précis).

- **Modèle Relationnel**

  C'est un modèle de niveau logique composé de deux concepts

  - Relation (table)
  - Attribut (colonne)

  Il est défini par Ted Codd en 1970 et développé par IBM lab. C'est un support théorique très solide utilisé aujourd'hui par beaucoup de SGBD commerciaux (Oracle, Informix, DB2, Ingres, Sybase, dBase, Access) et SIG

- **Règles de passage du MCD au MLDR**

La phase de Modélisation Conceptuelle des Données étant réalisée (modèle MCD), il va s'agir de déterminer un choix d'organisation

de ces données en fonction de technologies existantes : actuellement le Modèle Relationnel prédomine, mais il n'y a pas si longtemps, le modèle CODASYL a eu son heure de gloire et rien ne dit que dans quelques années le modèle OBJET ne deviendra pas la référence.

La transformation du Modèle Conceptuel de Données, vers le Modèle Logique de Données Relationnel résulte de l'application mécanique de 4 règles principales.

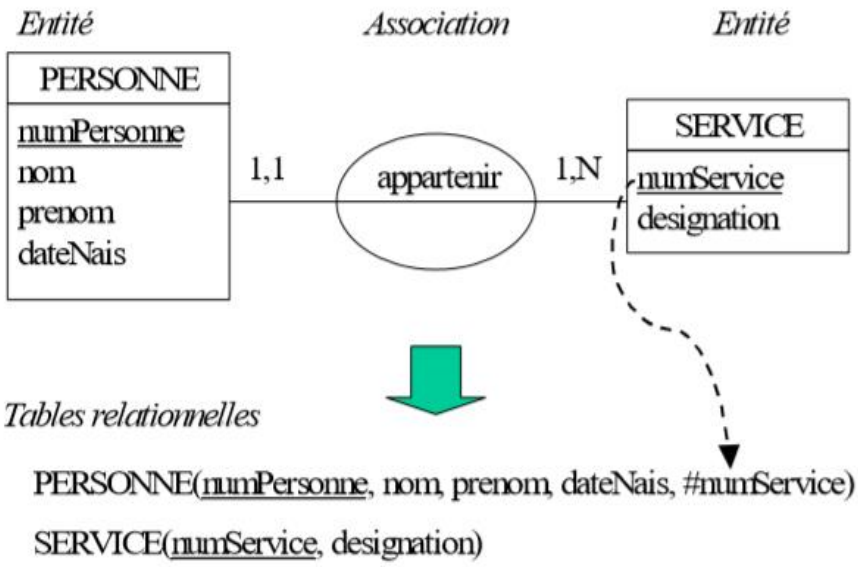
Règle 1 : chaque entité se transforme en table

- Chaque entité se transforme en table relationnelle.
- Chaque propriété se transforme en colonne de la table.

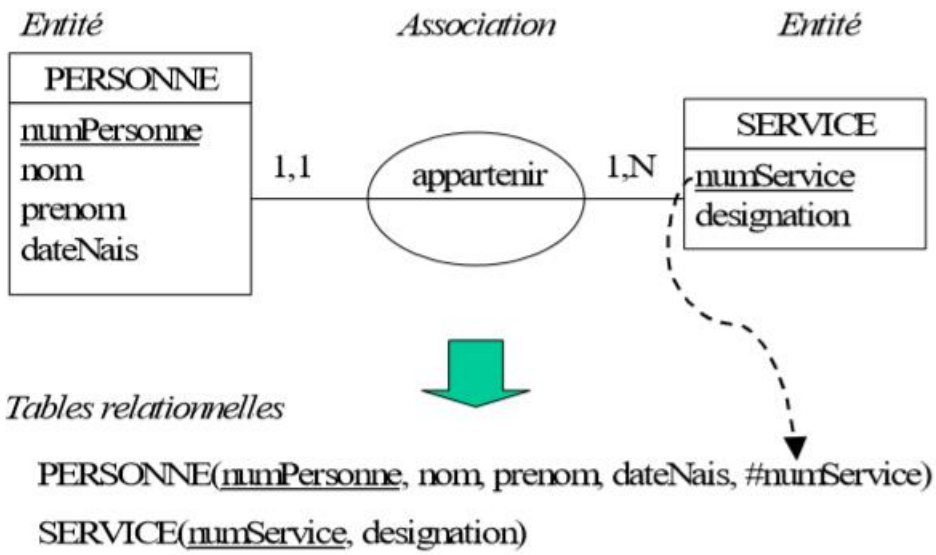
Règle 2 : association binaires 1,1 - 1,N ou 1,1 - 0,N

Chaque association binaire non porteuse de données ayant les cardinalités (1,1-1,N ou 1,1 - 0,N) se traduit par une redondance de l'identifiant de l'entité

- ayant la cardinalité 1,N ou 0,N dans la table issue de l'entité ayant la cardinalité 1,1.
- L'identifiant de l'entité devient la clef de la table.



- Si l'association est porteuse de propriétés (elle ne devrait pas l'être), celles-ci se retrouvent comme colonnes dans la table issue de l'entité participant avec la cardinalité 1,1.
- de l'entité participant avec la cardinalité 1,1.



Dans la table PERSONNE :

numService est « clef étrangère », et fait référence à numService « clef primaire » dans la table SERVICE.

Règle 3 : associations N-aires

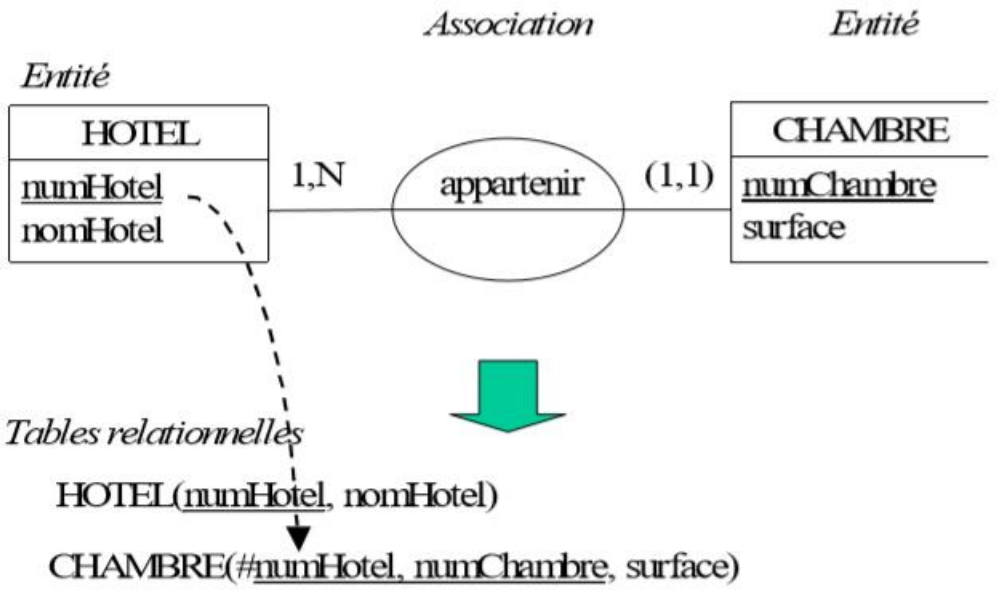
- Une association N-aire, porteuse ou non de propriétés, ayant les cardinalités 0,N ou 1,N, se transforme en une table ayant comme clef composite les colonnes issues des identifiants des entités participant à l'association

Dans la table STOCKER :

- refProduit est « clef étrangère » et fait référence à refProduit « clef primaire » de la table PRODUIT
- codeDepot est « clef étrangère » et fait référence à codeDepot « clef primaire » de la table DEPOT
- date est « clef étrangère » et fait référence à date « clef primaire » de la table DATE_STOCK

Règle 4 : entité faible

- Dans le cas d'une entité faible, la clef primaire de la table relationnelle est constituée par l'identifiant de l'entité forte et par l'identifiant de l'entité faible.



Dans la table CHAMBRE

numHotel fait partie de la clef primaire composité et est « clef étrangère » faisant référence à numHotel « clef primaire » de la table HOTEL.

III. SQL

✓ Langage de définition des données (LDD) :

Un **langage de définition de données** (**LDD** ; en anglais *data definition language*, DDL) est un langage de programmation et un sous-ensemble de SQL pour manipuler les structures de données d'une base de données, et non les données elles-mêmes.

Il permet de définir le domaine des données, c'est-à-dire l'ensemble des valeurs que peut prendre une donnée : nombre, chaîne de caractères, date, booléen. Il permet aussi de regrouper les données ayant un lien conceptuel au sein d'une même entité. Il permet également de définir les liens entre plusieurs entités de nature différente. Il permet enfin d'ajouter des contraintes de valeur sur les données.

- **CREATE :** création d'une structure de données ;
- **DROP :** suppression d'une structure de données ;
- **ALTER :** modification d'une structure de données ;
- **RENAME :** renommage d'une structure de données ;
- **TRUNCATE** :

✓ Langage de manipulation de données (LMD) :

Un **langage de manipulation de données** (**LMD** ; en anglais *data manipulation language*, DML) est un langage de programmation et un sous-ensemble de SQL pour manipuler les données d'une base de données.

Ces commandes de manipulation de données doivent être validées à l'issue d'une transaction pour être prises en compte.

- **INSERT :** insertion de données dans une table ;
- **UPDATE :** mise à jour de données d'une table.
- **DELETE :** suppression de données d'une table ;

✓ Langage d'interrogation de données (LID) : SELECT ;

Le langage d'interrogation de données (LID) permet d'**établir une combinaison d'opérations** portant sur des tables (relation). Le résultat de cette combinaison d'opérations est lui-même une table dont l'existence ne dure qu'un temps.

- **Jointure**

  C'est une opération permettant de ramener sur une même ligne des données venant de plusieurs tables. Une jointure s'effectue grâce à un produit cartésien de plusieurs tables (256 au maximum) et l'application de sélections.

Projection

La projection est une opération qui consiste à ne sélectionner que certaines données pour l'affichage. La syntaxe est la suivante :

Sous requête

- **EXISTS**

Dans le langage SQL, la commande EXISTS s'utilise dans une clause conditionnelle pour savoir s'il y a une présence ou non de lignes lors de l'utilisation d'une sous-requête.

- **ALL**

Dans le langage SQL, la commande ALL permet de comparer une valeur dans l'ensemble de valeurs d'une sous-requête. En d'autres mots, cette commande permet de s'assurer qu'une condition est "égale", "différente", "supérieure", "inférieure", "supérieure ou égale" ou "inférieure ou égale" pour **tous** les résultats retourné par une sous-requête.

- **ANY / SOME**

Dans le langage SQL, la commande ANY (ou SOME) permet de comparer une valeur avec le résultat d'une sous-requête. Il est ainsi possible de vérifier si une valeur est "égale", "différente", "supérieur", "supérieur ou égale", "inférieur" ou "inférieur ou égale" pour **au moins une des valeurs** de la sous-requête.
A noter : le mot-clé SOME est un alias d'ANY, l'un et l'autre des termes peut être utilisé.

Alias (AS)

Dans le langage SQL il est possible d'utiliser des **alias** pour renommer temporairement une colonne ou une table dans une requête. Cette astuce est particulièrement utile pour faciliter la lecture des requêtes.

Commande

Le **SELECT** est la commande de base du SQL destinée à extraire des données d'une base ou calculer de nouvelles données à partir d'existantes. La syntaxe est la suivante :

- **UNION** : permet de mettre bout-à-bout les résultats de plusieurs requêtes.

- **GROUP BY :** [GROUP BY ordre des groupes] ; regroupement de résultat d'opérations d'agrégat

- **ORDER BY :** critères de tri ou liste de colonnes

- **HAVING :** [HAVING condition] ; restriction sur l'affichage des résultats d'opérations d'agrégat

○ Fonction

- DISTINCT

L'utilisation de la commande SELECT en SQL permet de lire toutes les données d'une ou plusieurs colonnes. Cette commande peut potentiellement afficher des lignes en doubles. Pour éviter des redondances dans les résultats il faut simplement ajouter DISTINCT après le mot SELECT.

- **COUNT :** renvoie le **nombre d'enregistrements** de la table.

- LIMIT

La clause LIMIT est à utiliser dans une requête SQL pour spécifier le nombre maximum de résultats que l'on souhaite obtenir. Cette clause est souvent associé à un OFFSET, c'est-à-dire effectuer un décalage sur le jeu de résultat. Ces 2 clauses permettent par exemple d'effectuer des systèmes de pagination (exemple : récupérer les 10 articles de la page 4).

- LIKE %% (B%A)

L'opérateur LIKE est utilisé dans la clause WHERE des requêtes SQL. Ce mot-clé permet d'effectuer une recherche sur un modèle particulier. Il est par exemple possible de rechercher les enregistrements dont la valeur d'une colonne commence par telle ou telle lettre. Les modèles de recherches sont multiple.

- IN & NOT IN

L'opérateur logique IN dans SQL s'utilise avec la commande WHERE pour vérifier si une colonne est égale à une des valeurs comprise dans set de valeurs déterminés. 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.

- BETWEEN

L'opérateur BETWEEN est utilisé dans une requête SQL pour sélectionner un intervalle de données dans une requête utilisant WHERE. L'intervalle peut être constitué de chaînes de caractères, de nombres ou de dates. L'exemple le plus concret consiste par exemple à récupérer uniquement les enregistrements entre 2 dates définies.

- AVG : renvoie la **moyenne** d'un champ (valeurs de données de type numérique ou date/heure).

- MIN : renvoie la **valeur minimale** d'un champ (valeurs de données de type numérique, date et texte).

- MAX : renvoie la **valeur maximale** d'un champ (valeurs de données de type numérique, date et texte)

- SUM : renvoie à la **somme** d'un champ (valeurs de données de type numérique ou date/heure).