**MERISE - E/R – Programme CDA**

# Introduction

**MERISE** est une **méthode d’analyse et de conception** (données et traitements notamment) **et de gestion de projet informatique**.

On peut retenir sur le plan historique, que la **méthode** spécifiquement **française** date du début des années 1980, que dans son ensemble elle était pertinente dans un contexte d’**informatisation massive des organisations** (entreprises et administrations). Enfin sur le plan de la conception des bases de données, elle n’était **pas uniquement adaptée aux bases de données relationnelles**.

Plutôt tombée en désuétude, les concepts associés sont toujours d’**actualité** dans la **conception de bases de données relationnelles**.

Dans le cadre du titre **CDA**, seules **l’analyse et la conception des données sont concernées dans un contexte de bases relationnelles**. *Exit* les traitements et la gestion de projet au sens MERISE.

Dans les grandes lignes, MERISE propose **trois niveaux** d’analyse et de conception des données. Le niveau **conceptuel**, **logique** et **physique**.

Au commencement de toute analyse et conception d’une application, se trouve l’**organisation** elle-même (entreprise ou administration) avec ses activités, ses métiers, ses collaborateurs, ses clients et fournisseurs, ses **processus**, ses **informations** (données) qu’elle manipule…

Bien souvent le concepteur développeur dispose d’un **cahier des charges** ou d’une **expression de besoins** plus ou moins détaillés dans lesquels on retrouve ces informations. Dans certains contextes, des modèles issus d’une analyse MERISE se retrouvent dans le cahier des charges, **spécifications** fonctionnelles et/ou techniques.

Analyser et concevoir, c’est extraire les informations essentielles de ce contexte, le domaine étudié, sous la forme d’**entités**, de **propriétés** d’entités et d’**associations** entre ces mêmes entités.

Concernant l’analyse et la conception des données, **la finalité est de disposer d’une base de données** permettant de **stocker** et **manipuler** les **informations de l’organisation** en respectant ses besoins en matière de données.

Pour atteindre ce but, MERISE propose **trois étapes**, une à chacun des niveaux. Chaque étape est une **description**, à un certain niveau, des données de l’organisation, et un **processus de transformation** (ou traduction) est à l’œuvre **entre chacune de ces étapes**. Les transformations s’appuient sur un ensemble de **règles de normalisation visant notamment à limiter la redondance**.

Chaque étape conduit à la production d’un **modèle de données**. Entendons par modèle, **pas nécessairement un modèle graphique** tel un schéma ou un diagramme. Ce modèle peut être représenté de manière textuelle ou même en langage informatique (SQL) comme a la dernière étape avec le modèle physique.

Parallèlement à MERISE, des auteurs ont travaillé à la **représentation graphique du modèle conceptuel des données**, c’est ce qu’on appelle le modèle ou **diagramme** **Entité/Association** (E/A), Entity-Relationship (E/R) en anglais. Ce développement est étroitement lié à l’apparition des outils [AGL](https://fr.wikipedia.org/wiki/Atelier_de_g%C3%A9nie_logiciel). Ce support utilise [Looping](https://www.looping-mcd.fr/) pour les exemples.

Il ne faut donc **pas confondre la méthode et la ou les représentations** des différents modèles issus de l’application de la méthode.

# Les trois niveaux de l’analyse MERISE

## Le niveau conceptuel

### Le niveau conceptuel, généralités

* L’analyse des données au niveau conceptuel conduit à la production d’un **MCD** (Modèle Conceptuel des Données)
* À ce stade **on n’a aucune considération technique**, **pas même le type de base de données** qui sera ou devrait être utilisé plus tard pour stocker les données. Non pas qu’on ne connait pas le type de la base de données, mais que **ça n’est pas le point d’attention à ce stade**
* On ne se préoccupe de l’organisation que d’un **point de vue métier**, on emploie le **vocabulaire** métier, dans un **langage naturel** partagé par les parties prenantes du projet. Ainsi un MCD peut tout à fait être en français, il doit parler à et être compris de tout le monde
* Le travail du concepteur est à ce stade de réaliser des ***abstractions* des objets du « monde réel »**, autrement dit les **entités que manipule l’organisation** dans ses activités et métiers (commandes, factures, clients, événements, produits, collaborateurs…). On parle aussi d’**objets métiers ou de gestion**
* **On ne se préoccupe pas des règles de gestion**. Il n’est pas question ici de savoir si une date de fin doit être supérieure à une date de début, ni même si une propriété est obligatoire ou non
* **On ne se préoccupe pas du type** des propriétés, des données. Indiquer un typage suppose de connaître non seulement le type de base de données, encore plus la base de données précise qui sera utilisée, le support physique n’est pas un sujet de préoccupation à ce stade. En tant que support de communication, un bon nommage des propriétés indique implicitement le type générique, ce qui est suffisant ici
* On se concentre sur l’**identification** et l’**organisation des informations**, des données
* Il ressort donc principalement de cette étape un **ensemble d’entités avec des propriétés, et des associations entre ces entités**. Une grande partie des abstractions provient « tout simplement » de phrases en langage naturel dites par les acteurs du système (un client passe une commande, un client est facturé, un client est une personne physique ou morale avec un nom et une adresse, un client a passé au moins une commande et peut en passer plusieurs…)

### Le niveau conceptuel, définitions et règles

* L'**entité** est un **objet métier** considéré **d'intérêt** pour représenter l'activité à modéliser, chaque occurrence de l’entité est un individu de la population du nom de l’entité. Par exemple : « France » est un individu de la population « Pays »
* Une **entité** a un **nom unique** dans le modèle, conventionnellement au pluriel (représente plusieurs)
* **Chaque entité a une ou plusieurs propriétés** (attributs) qui la qualifient. Les propriétés sont des **noms**, nécessairement **uniques** dans une entité, on peut retrouver le même nom de propriété dans différentes entités. Un pays porte généralement dans les modèles les propriétés « Code ISO » et « Nom », et pourrait avoir la propriété « Superficie » dans certains modèles en fonction de l’activité de l’organisation
* Les **propriétés** doivent être **simples** et **atomiques** (pas décomposables) comme un nom, un prix, une date…
* **Chaque entité a une propriété spécifique**, une propriété qui sert d’**identifiant** (unique et discriminant). Le nom des identifiants doit être unique dans l’ensemble du modèle, par exemple « Code ISO du pays » pour l’entité « Pays » et « Code ISO de la devise » pour l’entité « Devises ». Le code ISO est un bon candidat pour l’entité « Pays » car ISO est un standard : « FR », « US », « CN » … En l’absence d’identifiant, il faut en définir un, en s’appuyant sur une règle stable et dont les valeurs parlent aux humains (ceux qui pratiquent l’activité/le métier dans l’organisation). Les organisations ont besoin d’identifier les objets qu’elles manipulent, ça n’est pas un besoin technique. Autres exemples : numéro de commande ou de facture, immatriculation, numéro de série, ISBN…
* Les **associations** entre entités sont un **lien sémantique**, identifié par un **verbe unique dans le modèle** et **quantifié**, qui se lit dans les deux sens. Verbe à l’infinitif ou conjugué pour donner le sens principal de lecture, ainsi on peut lire à la voix active puis passive dans un sens puis dans l’autre. Un transporteur *livre* une commande *vs* une commande *est livrée* par un transporteur
* Les **associations** n’ont pas d’identifiant mais **peuvent avoir des propriétés** (simples et atomiques comme au niveau des entités). Par exemple les produits vendus sont soumis à un taux de TVA qui peut évoluer dans le temps, dans certaines organisations on pourrait avoir besoin de stocker les évolutions du taux, conceptuellement on aurait une entité « Produits » en association avec une entité « Taux de TVA », association « Appliquer » dans laquelle on peut ajouter une propriété « Date effective » indiquant à partir de quelle date un taux est appliqué sur un produit
* Le concept de *quantité* dans les associations est appelé **cardinalité**. Une cardinalité indique le nombre **minimum** (0 ou 1) et **maximum** (1 ou n) de fois où une occurrence d'une entité (un individu d’une population) peut participer à une association, et ce dans les deux sens (des deux côtés de l’association). **Un minimum de 0 indique que l’association n’est pas *obligatoire* pour que l’occurrence puisse exister**. Contre-exemple, si dans une application cela n’a pas de sens (conceptuellement, fonctionnellement) qu’un utilisateur n’ait pas de rôle alors la cardinalité minimale de « Utilisateurs » vers « Rôles » doit être de 1
* Une association peut exister entre deux entités (associations **binaires** => très fréquentes) ou plusieurs (associations **ternaires** => plutôt rares), mais également sur une seule entité (associations **réflexives** => permet généralement de modéliser une hiérarchie)
* **On évitera les associations 1/1**, c’est-à-dire là où les cardinalités maximums d’une association sont de 1 des deux côtés, c’est souvent le signe d’une mauvaise analyse, d’une incompréhension. Par exemple un utilisateur (personne physique) possède un compte et ce même compte ne peut être associé qu’à un seul utilisateur (le même), dans une application mono-compte cela ne fait pas de sens d’avoir deux entités, contrairement à une application multicomptes ou permettant les comptes partagés
* **Ne pas représenter de données calculées dans une propriété**, comme des sommes qui peuvent être déduites des autres données. Une commande est faite de ligne de commandes avec des quantités et des prix (ceux des articles), on peut obtenir le montant total de la commande en faisant les multiplications et les additions, il n’est pas conceptuellement nécessaire de se représenter le total dans une propriété. Exemple plus discutable et non mathématique, le numéro de sécurité sociale : dans un système qui dispose de toutes les informations de sexe, de date et de lieu de naissance alors on peut calculer (reconstituer) le numéro de sécurité sociale, par contre si on ne dispose pas de toutes ces informations séparément alors il est acceptable de se représenter le numéro de sécurité sociale dans une et une seule propriété

### Exemple de MCD textuel

* **Contexte** : une entreprise qui souhaite déployer une application à destination des centres de formation pour leurs permettre de suivre leurs apprenants dans la préparation à différents titres RNCP
* **Spécification des (E) entités et de leurs (P) propriétés, incluant (ID) l’identifiant (unique et discriminant)**
  + Les (E) centres de formation sont des entreprises avec un (ID) numéro RCS et une (P) dénomination sociale
  + Les (E) apprenants ont une (ID) adresse électronique, un (P) prénom, un (P) nom et une (P) date de naissance
  + Les (E) titres RNCP ont un (ID) code de registre des titres et un (P) nom
  + Un (E) niveau de diplôme n’ayant pas d’identifiant standard, nous ajouterons et garantirons un (ID) code de niveau, mais dispose d’un (P) nom de niveau français et son (P) équivalent européen
* **Spécification des (A) associations et (Q) quantification / cardinalités entre entités**
  + Chaque (E) apprenant peut (A) être le mentor de (Q) plusieurs autres, et peut (A) être mentoré ou non par (Q) plusieurs autres, (Q) un (E) apprenant « mentor » peut aussi (A) être mentoré
  + Chaque (E) apprenant (A) prépare (Q) au moins un (E) titre RNCP, (Q) un (E) titre RNCP (A) est préparé dans (Q) un (E) centre de formation, (Q) une préparation se fait d’une (P) date de début à une (P) date de fin
  + A chaque (E) titre RNCP (A) correspond (Q) un et un seul (E) niveau de diplôme, (Q) un (E) niveau de diplôme peut (Q) ne pas (A) être en correspondance avec (Q) un (E) titre RNCP et peut (A) être en correspondance avec (Q) plusieurs (E) titres RNCP
  + Précision : un titre RNCP, comme un centre de formation, peut ne pas avoir d’apprenant en préparation

## Le niveau logique

### Généralités

* L’analyse des données au niveau logique conduit à la production d’un **MLD** (Modèle Logique des Données), le MLD est la **traduction du MCD**
* Cela suppose qu’on a fait le **choix**, du moins que l’on tient compte, **du type de SGBD** (Système de Gestion de Base de Données) que l’on va utiliser. Le type informe sur la **manière dont les données vont être représentées, organisées** (tables, réseaux, hiérarchies, documents…). Pour le type relationnel on parle de **SGBD-R** et **les données sont organisées sous la forme de tables**
* **On ne tient pas compte de l’implémentation**, à savoir de l’éditeur, du SGBD-R cible (par exemple MySQL ou encore PostgreSQL). Non pas qu’on ne le connait pas mais que ça n’est pas le point d’attention à ce stade. En conséquence, comme au niveau conceptuel, **on ne peut pas se soucier du typage** des données car les SGBD-R se supportent pas tous les mêmes types et pour un type proche, voire le même, ils n’auront pas forcément les mêmes noms. Pour résumer, il ne peut y avoir comme niveau de détails dans le modèle logique que des **concepts communs à tous les SGBD-R**
* Quand le **type** est une base de données **relationnelle**, on parle de **modèle** (logique) **relationnel**, en raccourcis **MLD-R**. On peut se représenter le MLD-R sous un format **textuel ou graphique**, on parle alors de **schéma relationnel** (pas standard). Attention, ce schéma relationnel n’est **pas un schéma E/A** (ou E/R en anglais) car le MLD-R est justement la traduction du modèle/schéma conceptuel (E/A) ! Cette traduction suit un ensemble de règles
* On comprend à ce stade que **la conception est un processus itératif**, on ne trouve pas LE modèle en un claquement de doigts. D’autres règles et vérifications s’appliquent au niveau logique et on peut s’apercevoir uniquement à ce stade d’une erreur de conception, donc de devoir revoir le MCD puis revenir sur le modèle logique. C’est la raison pour laquelle il est vivement **recommandé de s’appuyer sur un logiciel de modélisation** qui permet de générer le modèle logique à partir du MCD **et qui respecte les règles de traduction**
* *Last but not least*, il faut le plus tôt et le plus régulièrement possible **confronter sa modélisation émergente à de vraies données**. Le meilleur ami du concepteur de base de données relationnelles est un tableur. On peut facilement se représenter un modèle avec des tableaux et des données dedans. On voit plus aisément et rapidement les défauts de conception en voyant des données !

### Définitions et règles

* Les règles de transformation (traduction du MCD en MLD-R) peuvent tenir en une phrase-clef : les entités deviennent des **tables**, les propriétés deviennent des **colonnes**, les identifiants deviennent des **clefs primaires** et les associations deviennent des **clefs étrangères**
* Les tables et les colonnes (dont la clef primaire) ont bien entendu des noms, noms qui respectent les mêmes règles pour l’unicité qu’au niveau conceptuel. Cependant, le niveau logique est un niveau « technique », des **conventions de nommage** s’appliquent sur les différents « objets » d’une base de données, dont les tables et les colonnes. En lieu et place des noms conceptuels, à ce stade **les tables et les colonnes ont des noms dits logiques**. Même si ces noms sont des identifiants techniques ils doivent **s’inspirer du vocabulaire métier donc des noms conceptuels**. Les conventions de nommage peuvent dépendre des pratiques de l’organisation ou encore du SGBD-R cible, le principal est de définir une convention et de s’y tenir
* **Toute table a une clef primaire**, si l’identifiant de l’entité correspondante est composé alors la clef primaire de la table l’est également (même si rare ou plutôt déconseillé)
* Les subtilités de traduction du MCD en MLD-R tiennent dans la **transformation des associations en clefs étrangères**
* **Note** : certaines caractéristiques des clefs étrangères, qui n’ont rien à voir avec la traduction du MCD en MLD-R, ne sont pas présentées dans ce support (notamment cascade et référencement de colonnes non primaires) et de plus pas forcément supportées par tous les SGBD-R
* Une **clef étrangère** est une **contrainte** d’intégrité qui garantit l’**intégrité référentielle**, généralement entre deux tables, plus exceptionnellement dans une seule table (cas des associations réflexives). L’intégrité référentielle est une des caractéristiques fondamentales des bases de données relationnelles. Elle permet de s’assurer qu’un enregistrement existe bien dans la table référencée
* La **clef étrangère identifie** une ou plusieurs colonnes car **la clef étrangère référence la clef primaire**, qui peut être composée (sur plusieurs colonnes). La clef étrangère est déclarée dans la table qui référence
* En terminologie de modélisation, **la clef étrangère représente l’association** d’une entité avec une autre (ou elle-même), dans le modèle logique elle indique où cette information est stockée
* **Pour savoir où** est déclarée (dans le MLD-R) une clef étrangère issue d’une association, **on s’intéresse** à la cardinalité de l’association et plus précisément **aux maximums**, **cette information précise où (dans quelle table) glisse(nt) la ou les clefs primaires** (et éventuellement les propriétés de l’association), soit dans une des deux tables soit dans une nouvelle table dite de jointure (ou encore de jonction ou d’association)
* Dans le **cas d’une association** où les maximums sont d’un côté **max=1** et de l’autre **max=n** (dite association 1/n), **la clef primaire glisse dans la table du côté max=1** et ajoute une (ou plusieurs) colonne(s) de clef étrangère. Par exemple, si un produit est catégorisé par un maximum d’une (1) catégorie de produits et qu’une catégorie de produits peut catégoriser plusieurs (n) produits, nous sommes dans une association 1/n, alors la clef primaire de la table des catégories glisse vers la table des produits et devient une clef étrangère (une ou plusieurs colonnes selon la composition de la clef primaire)
* Dans le **cas d’associations réflexives 1/n**, la clef primaire glisse pareil dans la table du côté max=1, **c’est-à-dire la table elle-même** (ajout de la clef étrangère dans cette même table)
* Dans le **cas d’une association où les maximums sont n des deux côtés** (dite association n/n) alors **les deux clefs primaires glissent dans une nouvelle table de jointure** dans laquelle **elles deviennent deux clefs étrangères vers leur table respective**, et comme toute table doit avoir une clef primaire, **la combinaison des clefs étrangères devient la clef primaire de la table de jointure**
* En ce qui concerne les **associations avec des propriétés**, dans le cas d’une **association 1/n les propriétés de l’association migrent dans la même table que les clefs étrangères**, dans le cas d’une **association n/n les propriétés migrent dans la table de jointure**
* Pour les **associations ternaires** (plus de deux entités), la **traduction** en MLD-R **est la même que pour les associations binaires n/n** où toutes les clefs primaires deviennent des clefs étrangères dans la nouvelle table de jointure et leur combinaison la clef primaire de la table de jointure, ainsi que toutes les propriétés de l’association le cas échéant
* **Note sur le nommage de la table de jointure** : on reprend le **nom logique de l’association** qui peut être basé **soit sur le verbe soit sur la concaténation des noms logiques** des entités (tables) concernées par l’associations. Par exemple, « products\_categories » vs « categorize », la première étant plus fréquente mais pas nécessairement la meilleure convention, en effet il peut exister plusieurs associations entre mêmes entités et il faut bien lever l’ambiguïté… Cette convention peut donner des tables de jointures avec le même nom, ce qui n’est pas permis ! Aussi des noms à rallonge !

### Exemple de MLD-R textuel

Avant l’apparition des outils graphiques (AGL) on représentait le modèle de données au format textuel (idem au niveau conceptuel). Le formalisme ci-dessous que l’on retrouve dans Looping se lit comme suit :

* + En gras, le nom de la table suivi du signe « = » puis des colonnes de la table entre parenthèses
  + La ou les colonnes (clef composée) soulignées représentent la clef primaire de la table
  + Une colonne précédée d’un « # » représente une clef étrangère

Le MDL-R :

**learners**=(email, firstname, lastname, birthdate);

**training\_centers**=(company\_number, company\_name);

**diploma\_levels**=(level\_code, french\_level, european\_level);

**certificates**=(certificate\_code, certificate\_name, #level\_code);

**preparation**=(#email, #company\_number, #certificate\_code, start\_date, end\_date);

**mentoring**=(#email\_mentor, #email\_mentoree);

## Le niveau physique

* L’analyse des données au niveau logique conduit à la production d’un **MPD** (Modèle Physique des Données), le MPD est la **traduction du MLD-R**
* **On tient compte de l’implémentation**, du SGBD-R cible, on prend en considération le système de stockage. Et pour cause, **le MPD est la traduction en langage de la base de données**. Dans un contexte **relationnel**, il s’agit d’**instructions SQL** pour créer les tables avec leurs définitions de colonnes, de clefs primaires et étrangères. C’est un modèle qui **permet de construire physiquement la base de données**
* On reprend généralement les **noms logiques** pour les tables et les colonnes mais le SGBD-R cible peut « imposer » ses propres **conventions** pour les tables, colonnes, indexes, clefs et autres objets. Les SGBD-R disposent de syntaxes « courtes » (des raccourcis) pour définir les clefs et autres contraintes, cependant en utilisant les raccourcis on ne spécifie pas le nom qui est généré automatiquement par le SGBD-R avec sa propre **stratégie de nommage**. La **bonne pratique** est d’**utiliser la syntaxe « longue »** permettant d’appliquer ses propres conventions
* Connaissant la cible, on peut et même on doit **spécifier les types de données** pour chaque colonne, selon les types leurs **précisions** (longueurs de chaînes, tailles pour les entiers et parties décimales…), les **contraintes** (non nullité, unicité). On peut également se préoccuper des performances et ajouter des indexes et mêmes des colonnes calculées (pour stocker une bonne fois pour toute un total par exemple, ce qui évite de le calculer à chaque besoin)
* L’**ajout et** la **suppression de colonnes** (rare) sont motivés par l’**optimisation des performances** (mémoire et/ou temps de traitement) et représentent des actions de **dénormalisation** (certaines règles énoncées dans les bonnes manières et pratiques de conception sont « violées »). Dans tous les cas il faut veiller à répondre aux besoins, respecter les données utiles à l’organisation pour exercer ses activités, ses métiers
* **Le MPD** est donc **à la fois la dernière étape de conception** des données **et** le contenu du ou des **script(s) SQL à exécuter pour mettre en place** (ou à jour) le schéma d’une base de données relationnelle. Le script qui ne contient que des instructions de création et/ou modification de la structure est un script [DDL](https://fr.wikipedia.org/wiki/Langage_de_d%C3%A9finition_de_donn%C3%A9es) (Data Definition [or Description] Language), sous-ensemble de SQL pour spécifier le type d’instructions que contient le script
* Une pratique courante en bases de données relationnelles, pour des raisons de **simplicité** (pour écrire des requêtes), d’**uniformité** (même type et nom dans toutes les tables) et de **performance** (notamment pour les jointures), est d’avoir des **clefs primaires numériques sur une seule colonne**, auto-incrémentées ou basées sur des séquences
* Ceci peut être modifié, **ajouté au niveau du MPD**. Cependant les **identifiants (MCD)** puis **clefs primaires (MLD-R)** sont des informations nécessaires à l’organisation et **ne peuvent donc pas être supprimées**. En conséquence de l’introduction d’une clef primaire purement « technique », elles ne peuvent pas être conservées comme des clefs primaires, il ne peut y avoir qu’une seule clef primaire dans une table
* **L’impact est assez important s’il n’est pas anticipé**, impact dans le MPD mais également dans le mapping côté applicatif le cas échéant (ORM). En effet il faut ajouter la définition de colonne pour la nouvelle clef primaire, ne pas déclarer comme clef primaire l’identifiant d’origine de l’entité mais cependant lui déclarer une contrainte d’unicité, car conceptuellement (fonctionnellement) il faut conserver cette notion de discriminant *métier* unique (pour rappel, des exemples : ISBN, numéro de facture, code ISO…). **L’identifiant fonctionnel, déclaré unique, devient une clef secondaire**. Et pour finir les clefs étrangères ne référencent plus les mêmes clefs primaires

### Exemples de MPD

#### MySQL

CREATE TABLE `learners` (

`email` varchar(255),

`firstname` varchar(100) NOT NULL,

`lastname` varchar(100) NOT NULL,

`birthdate` date NOT NULL,

**PRIMARY KEY** (`**email**`)

);

#### PostgresSQL

CREATE TABLE public."learners" (

"email" varchar(255),

"firstname" varchar(100) NOT NULL,

"lastname" varchar(100) NOT NULL,

"birthdate" date NOT NULL,

**PRIMARY KEY** ("**email**")

);

#### MySQL après dénormalisation

CREATE TABLE `learners` (

**`id` int AUTO\_INCREMENT**,

`email` varchar(255) **NOT NULL**,

`firstname` varchar(100) NOT NULL,

`lastname` varchar(100) NOT NULL,

`birthdate` date NOT NULL,

PRIMARY KEY (`**id**`),

**UNIQUE KEY `uk\_learners\_email` (`email`)**

);

#### PostgreSQL après dénormalisation

CREATE TABLE public."learners" (

**"id" SERIAL**,

"email" varchar(255) **NOT NULL**,

"firstname" varchar(100) NOT NULL,

"lastname" varchar(100) NOT NULL,

"birthdate" date NOT NULL,

CONSTRAINT "pk\_learners" PRIMARY KEY **("id")**,

**CONSTRAINT "uk\_learners\_email" UNIQUE ("email")**

);

## Avertissements et références

* Un peu comme UML, **MERISE**, et d’une manière générale la conception et la modélisation de bases de données sont un progressivement en **état d’abandon dans le monde de l’entreprise**. En conséquence, il y a des **interprétations** et des **confusions**, qui parfois surgissent lors d’une soutenance, il faut s’y préparer en **maîtrisant le sujet**, avoir des **références solides et cohérentes**
* **Premier sujet de discorde (**bien que plus rare que le deuxième) est que **pour certains les types de données doivent être présents dans le MLD-R et même parfois dans le MCD** ! Si on comprend et retient que la méthode **MERISE dit qu’on ne connait pas la base de données cible aux étapes conceptuelle et logique** alors **il est juste impossible de préciser des types**, aussi cela n’apporte rien en matière de conception.
* L’**intérêt** de se préoccuper du typage n’apparait qu’à l’**étape physique**. Cependant, connaissant le type de SGBD à l’étape logique on peut tout de même préciser des types génériques (numeric, text, yes/no, date…). « Yes/no » et pas « boolean » : au niveau logique on ne tient pas compte du SGBD cible, tous les SGBD-R ne supportent pas le type « boolean ». C’est aussi un argument qui vient défendre le peu d’intérêt d’indiquer le typage à ce stade logique. Quant au MCD on est uniquement concentré sur l’identification des données (pas leurs définitions) et leur organisation
* **Rappel** : un **bon nommage** d’attributs donne une indication du type générique aux concepteurs et aux lecteur des modèles
* Si on utilise un AGL qui peut générer le MPD à partir du MLD-R (graphique) alors on doit pouvoir indiquer tous les types (et précisions, contraintes, valeurs par défaut…) et tout ce qui concerne les performances (indexes…) dans le processus de traduction en précisant la base de données cible incluant sa version
* **Deuxième sujet de discorde** et plus fréquent que le premier, le **MPD en personne** ! **Pour certains le MPD est un schéma donc un diagramme donc une représentation graphique** ! C’est une mauvaise interprétation du terme *schéma* dans ce contexte. Comment une image peut-elle bien créer physiquement des tables dans un SGBD-R ?! Peut-être un jour avec l’intelligence artificielle…
* La réponse est probablement dans l’existence des outils graphiques de modélisation, les fameux AGL. MERISE est une méthode qui n’impose aucun formalisme et qui est née avant les AGL, pourtant MERISE décrit déjà ce qu’est un MPD en précisant que c’est le modèle dans le langage de la base de données cible. Il est donc impossible historiquement que le MPD soit un diagramme
* Autre source de confusion possible, les schémas E/A. Ce sont bien des représentations graphiques mais uniquement du MCD, pas du MLD(-R) et encore moins du MPD
* Autre source de confusion possible, les AGL particulier, à savoir les clients SQL avec interface graphique type DBeaver ou encore MySQL Workbench qui permettent de créer un *schéma (graphique) relationnel au niveau physique* (une sorte de fusion entre le MLD-R et le MPD) directement, puis de générer le script SQL de création de la base, et inversement, de partir d’un schéma physique et par rétroconception (*reverse engineering*) d’obtenir une sorte de « représentation graphique du MPD ». Ils ne représentent pas des diagrammes E/A (ou E/R en anglais) mais des EER (*Enhanced Entity-Relationship*), c’est du E/R étendu
* Ces outils de modélisation dans les clients SQL et les diagrammes de plus « haut niveau » que sont les EER sont nés pour avoir une et une seule représentation d’un modèle. Simplifier et faciliter la modélisation d’une part et réduire le coût de maintenance des modèles dans le monde de l’entreprise
* En effet ces outils permettent de faire de la conception et de la rétroconception, de générer les scripts SQL, offrent une représentation graphique avec les tables, les relations, les colonnes et toutes les informations sur le modèle physique (clefs primaires et étrangères, contraintes, indexes…). Tout cela en un seul diagramme, c’est pratique mais ça ne respecte toujours pas la démarche MERISE et c’est autre chose que les MCD et MLD-R. On a une sorte de « tout en un »
* Point d’attention : même si très efficace en apparence, concevoir directement avec un tel outil n’est vraiment pas une bonne idée car il y a (con)fusion des problématiques métiers et techniques. Dans la pratique aujourd’hui, la conception a tendance à se faire en équipe dans une salle avec tableaux blancs où on va se représenter, sans formalisme particulier, les entités, leurs propriétés et leurs associations. Prendre des photos, prendre du recul, discuter entre collègues. Puis éventuellement initialiser et maintenir un EER correspondant
* Dernier point, quand on arrive sur un projet sans documentation et encore moins de modèles de données (MCD et/ou MLD-R) on est bien content d’avoir ce type d’outils pour obtenir en quelques clics une représentation graphique plus lisible et informative du modèle de données qu’on doit appréhender ☺

### Références et extraits

**Définition** [**Wikipédia**](https://en.wikipedia.org/wiki/Database_schema) **du terme *schéma* dans un contexte de base de données (anglais)**

« The database schema is the **structure** of a database described in a **formal language** supported typically by a relational database management system (RDBMS). The term "schema" refers to the **organization** of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases). »

**Définition** [**Wikipédia**](https://fr.wikipedia.org/wiki/Mod%C3%A8le_physique_des_donn%C3%A9es) **du MPD**

« Dans la méthode Merise, le modèle physique des données (MPD) consiste à **implanter** une base de données dans un SGBDR. Le langage utilisé pour ce type d'opération est le **SQL**. »

**Définition de Cyril GRUAU du MPD (**[**Conception d’une base de données**](https://cyril-gruau.developpez.com/merise/)**)**

« La traduction d'un MLD conduit à un MPD qui précise notamment le **stockage de chaque donnée** à travers son **type** et sa **taille** [...]. La traduction d'un MLD relationnel en un modèle physique est la création (par des requêtes **SQL** [...]) d'une base de données hébergée par un SGBD relationnel particulier. »

**Définition de Dominique NANCI et Bernard ESPINASSE (**[**Ingénierie des systèmes d‘information : MERISE deuxième génération**](https://www.lirmm.fr/~laurent/POLYTECH/IG4/RAPPELS-MERISE/LivreMerisePDF-total-12sept14.pdf) **– *Vuibert 2001 4ème édition PDF rendue gratuite*)**

« Au niveau physique, le modèle physique de données (MPD) est une description de la ou des bases de données ou de l'ensemble des fichiers, exprimée dans la **syntaxe du système de gestion de bases de données** (SGBD) ou système de gestion de fichiers (SGF) adoptés. »