LA MÉTHODE MERISE

Le modèle relationnel en pratique.

Contenu

Les étapes de la démarche Merise

"Pour concevoir un système d'information robuste, il faut, en premier lieu, identifier les données à traiter et les relations entre ces données."

ANALYSE	**1. Recueillir les informations** a. L'interview b. Les documents c. Les contraintes sur les données **2. Constituer le dictionnaire des données** a. Repérer les données à représenter b. Supprimer toute donnée "calculée" c. Préciser les contraintes liées à chaque donnée d. Identifier les dépendances fonctionnelles
CONCEPTION	**3. Établir le modèle conceptuel** a. Repérer et créer les entités b. Attribuer à chaque entité un identifiant et compléter le dictionnaire des données c. Placer les propriétés dans les entités d. Repérer et placer les associations 4. **Valider le modèle** a. Respect de la 1ère Forme Normale (1FN) b. Respect de la 2ème Forme Normale (2FN) c. Respect de la 3ème Forme Normale (3FN) **5. Transformer le modèle conceptuel en modèle logique** a. Respect des règles de transformation b. Les entités et associations deviennent des relations c. Les identifiants deviennent des clés primaires d. Les associations impliquent la création de clés étrangères dans les relations
RÉALISATION	**6. Implémenter le modèle physique** a. Le modèle logique devient un schéma relationnel b. Les relations deviennent des tables c. Les attributs deviennent des champs (ou colonnes) d. Les clés primaires impliquent la création de contraintes d'unicité e. Les clés étrangères impliquent la création de contraintes d'intégrité relationnelle f. Choix du Système de Gestion de Base de Données g. Le schéma est implémenté dans un langage relationnel (création des tables et des contraintes) h. Le schéma est testé et validé par un ou plusieurs jeux de données (tests d'intégrité) i. Les processus métiers et de sécurité sont implémentés (procédures stockées, déclencheurs…) j. L'ensemble est testé en conditions réelles (tests fonctionnels) k. Livraison

Le Modèle Logique de Données (MLD)

Le modèle logique de données est uniquement composé de relations issues des entités et associations du modèle conceptuel (MCD).

Ces relations permettront par la suite de créer les tables au niveau du modèle physique.

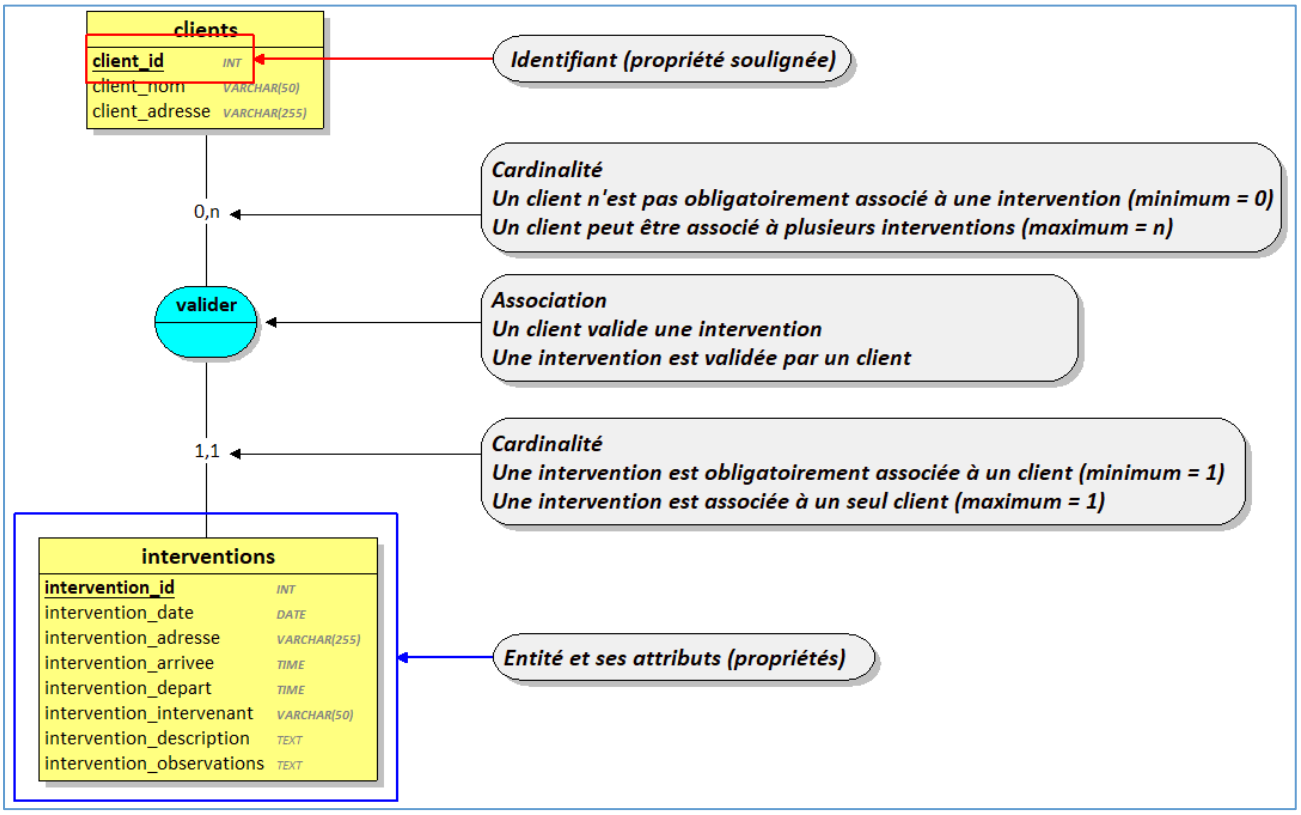
Une relation est composée d'attributs qui sont les données élémentaires issues du MCD (propriétés des différentes entités, identifiants et données de certaines associations porteuses).

Une relation possède un nom qui correspond en général à celui de l'entité ou de l'association dont elle est issue.

Elle possède aussi une clé primaire qui permet d'identifier sans ambiguïté chaque occurrence de cette relation.

La clé primaire peut être composée d'un ou plusieurs attributs, il s'agit d'une implantation de la notion d'identifiant des entités et associations qui se répercute au niveau relationnel.

Reprenons le modèle conceptuel du document précédent (informatiser les interventions)



2 entités sont représentées : **clients** et **interventions**.

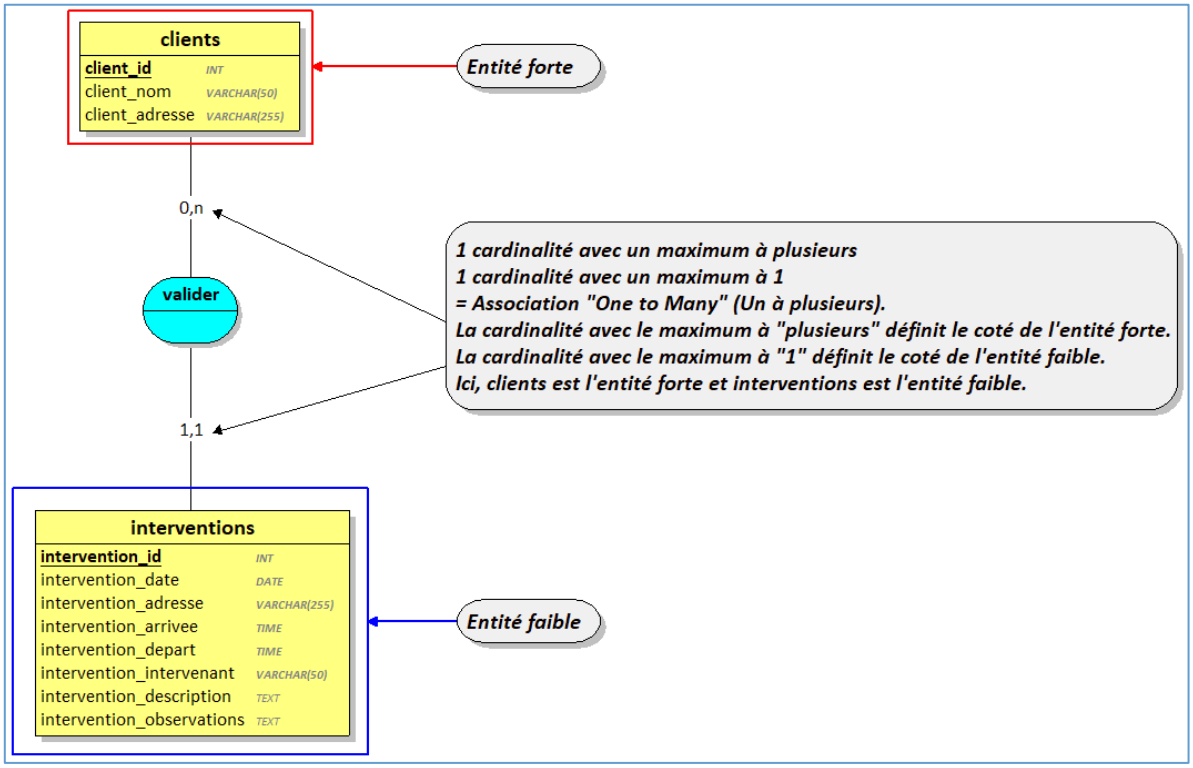
L'association **valider** représente la règle de gestion :

Un client valide 0 ou plusieurs interventions
1 intervention est validée par 1 seul client

Règles de transformations

Association « One To Many » (un à plusieurs)

L'association **valider** est une association "One to Many" :



Règles de transformation d'une association "One to Many" vers le modèle logique:

- L'entité forte devient la relation "**clients**" et conserve les mêmes propriétés. L'identifiant devient la **clé primaire**.
- L'entité faible devient la relation "**interventions**" dans laquelle on copie la clé primaire de l'entité forte (**client_id)**. cette propriété supplémentaire qu'on nomme **clé étrangère** est une référence vers la clé primaire de l'entité forte associée. Cette clé étrangère permet de s'assurer que la valeur qu'on renseignera dans la propriété **client_id** de la relation **interventions** existe dans la relation **clients**.

Le modèle logique est représenté au format texte de la manière suivante :

nom_de_la_relation (***clé primaire*, *liste*, *des*, *propriétés*,** *propriete_facultative*, ***#clé_étrangère***)

clients (***client_id*, *client_nom*, *client_adresse***)

interventions (***intervention_id***, *intervention_adresse*, ***intervention_date*, *intervention_arrivee*, *intervention_depart*, *intervention_intervenant*, *intervention_description*, *intervention_observations*, *#client_id***)

Un modèle logique de données (MLD) au format texte.

Propriété	= facultatif
Propriété en gras	= obligatoire
Propriété en gras et soulignée	= clé primaire
#Propriété commençant par un #	= clé étrangère (valeur NULL acceptée, cardinalité avec minimum à 0)
#Propriété en gras commençant par un #	= clé étrangère (valeur obligatoire, cardinalité avec minimum à 1)

Association « Many To Many » (plusieurs à plusieurs)

Dans l'entité « interventions », « intervention_intervenant » est le nom de l'intervenant principal (l'employé en charge de l'intervention). Chaque intervention nécessite le renfort d'ouvriers spécialisés.

Ajoutons quelques règles à notre système :

Un ouvrier est caractérisé par un matricule, un nom, un prénom et une spécialité.
Une intervention requiert 1 ou plusieurs ouvriers.
Un ouvrier est requis pour 0 ou plusieurs interventions.

Le dictionnaire des données doit être mis à jour :

Mnémonique	Signification	Type (longueur)	Contraintes
client_id	Numéro client	Entier	identifiant auto incrémenté
client_nom	Nom du client	Alphabétique (50)	obligatoire
client_adresse	Adresse du client	Alphanumérique (255)	obligatoire
intervention_id	Numéro d'intervention	Entier	identifiant auto incrémenté
intervention_adresse	Adresse de l'intervention (peut être différente de l'adresse du client)	Alphanumérique (255)	facultatif (adresse du client si vide)
intervention_date	Date de l'intervention	Date	obligatoire
intervention_arrivee	Heure d'arrivée	Heure	obligatoire
intervention_depart	Heure de départ	Heure	obligatoire & > à arrivée
intervention_intervenant	Nom de l'intervenant	Alphabétique (50)	obligatoire
intervention_description	Description de l'intervention	Alphanumérique (65535)	obligatoire
intervention_observations	observations de l'intervenant	Alphanumérique (65535)	facultatif
ouvrier_matricule	Numéro interne de l'ouvrier	Numérique (11)	identifiant
ouvrier_nom	Nom de l'ouvrier	Alphabétique (50)	obligatoire
ouvrier_prenom	Prénom de l'ouvrier	Alphabétique (50)	facultatif
ouvrier_specialite	Spécialité de l'ouvrier	Alphabétique (50)	obligatoire

Le dictionnaire des données mis à jour

Ces nouvelles données nous permettent de repérer une nouvelle dépendance fonctionnelle :

ouvrier_matricule → ouvrier_nom, ouvrier_prenom, ouvrier_specialite

Cette nouvelle dépendance fonctionnelle impliquera la création d'une nouvelle entité dont l'identifiant est « ouvrier_matricule » nous appellerons cette entité « ouvriers ».

Cette nouvelle entité repérée, reprenons la nouvelle règle de gestion précédemment citée :

Une intervention requiert 0 ou plusieurs ouvriers.
Un ouvrier est requis pour 0 ou plusieurs interventions.

Qu'on pourrait reformuler :

Plusieurs ouvriers peuvent être associés à une même intervention.
Plusieurs interventions peuvent être associées à un même ouvrier.

Une règle de gestion impliquant plusieurs occurrences de chaque entité (ici, ouvriers et interventions) fait ressortir une **association « plusieurs à plusieurs »**.

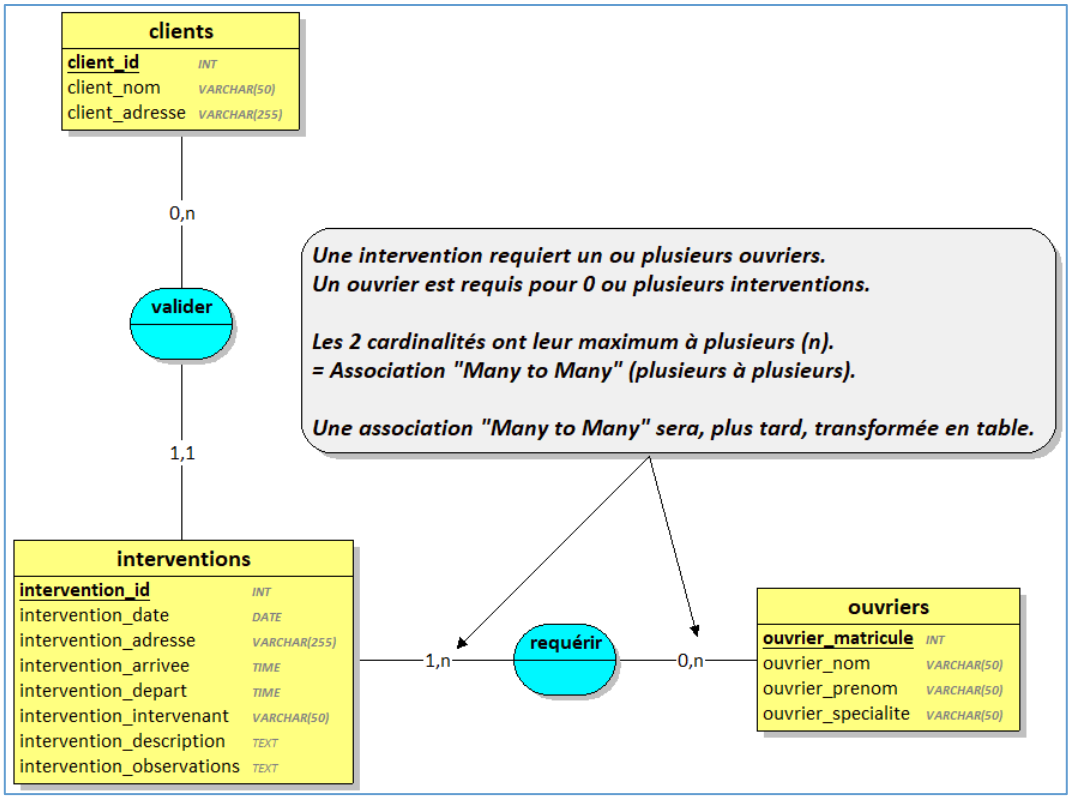
Plus concrètement,
À partir d'un ouvrier, je peux connaitre toutes les interventions qui lui ont été attribuées.
À partir d'une intervention, je peux connaitre tous les ouvriers attribués à ladite intervention.
Pour identifier un ouvrier unique pour une intervention unique, je dois connaitre les identifiants des 2 entités.
À partir d'une occurrence de chacun des 2 identifiants, je peux accéder à une occurrence unique de chaque entité :

intervention_id, ouvrier_matricule → intervention_adresse, intervention_date, intervention_arrivee, intervention_depart, intervention_intervenant, intervention_description, intervention_observations, ouvrier_nom, ouvrier_prenom, ouvrier_specialite

2 ou plusieurs propriétés permettent d'identifier une occurrence d'autres données
=
Dépendance fonctionnelle composée.

Une dépendance fonctionnelle composée relie les identifiants référencés à gauche de la flèche. Si des données présentes à droite de la flèche ne font partie d'aucune entité, cela indique que l'association sera porteuse de données. Ces associations seront plus tard transformées en tables.

Mise à jour du modèle conceptuel :



Règles de transformation d'une association "Many to Many" vers le modèle logique:

- L'association est transformée en une nouvelle relation (nouvelle table).
- Les identifiants des entités reliées (**intervention_id, ouvrier_matricule)** sont copiés dans cette nouvelle relation. Ces propriétés supplémentaires sont toutes 2 des **clés étrangères** et forment ensemble la **clé primaire** composée de cette nouvelle relation.

Pour notre cas, nous appellerons cette nouvelle relation « interventions_ouvriers »

Mise à jour du modèle logique :

clients (***client_id***, ***client_nom***, ***client_adresse***)

interventions (***intervention_id***, *intervention_adresse*, ***intervention_date***, ***intervention_arrivee***, ***intervention_depart***, ***intervention_intervenant***, ***intervention_description***, ***intervention_observations***, ***#client_id***)

ouvriers (**ouvrier_matricule**, **ouvrier_nom**, **ouvrier_prenom**, **ouvrier_specialite**) ← Nouvelle entité ouvriers

interventions_ouvriers (**#intervention_id**, **#ouvrirer_matricule**) ← Association many to many

Le modèle logique de données (MLD) mis à jour.

#Propriété soulignée en gras commençant par un # = clé étrangère et clé primaire

Une fois le MLD terminé, il servira de base à la modélisation du **modèle physique des données** qui sera implémenté avec le langage **SQL**.

Le modèle physique de Données (MPD)

Le modèle physique des données consiste à implémenter le modèle logique au sein d'un Système de Gestion de Bases de Données Relationnelles (SGBDR).

Langage de requêtes structurées

Le langage utilisé au niveau du modèle physique est généralement le langage SQL (Structured Query Language).

Le langage SQL, bien que standardisé depuis de nombreuses années, peut différer d'un SGBD à un autre.

A cette étape, il est donc primordial de bien sélectionner le SGBD qui sera adapté au système cible.
Ce choix se fera en fonction de plusieurs critères tels que le volume de données à traiter par transaction, le niveau de sécurité attendu, les performances ou le budget (le prix d'un SGBD varie de gratuit à plusieurs dizaines de milliers d'euros par licence).

Les SGBDR les plus connus sont :

- Oracle Database : https://www.oracle.com/
- Microsoft SQL Server : https://www.microsoft.com/fr-fr/sql-server/
- MySQL : https://www.mysql.com/fr/
- MariaDB : https://mariadb.org/ (Fork de MySQL 100% compatible)
- PostgreSQL : https://www.postgresql.org/

D'autres systèmes de gestion de bases de données :
https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_gestion_de_base_de_donn%C3%A9es#Quelques_SGBD

```
CREATE TABLE Cheval
(
	ChevalId INT PRIMARY KEY,
	ChevalNom VARCHAR(50) NOT NULL,
	DateNaissance DATE NOT NULL
);

INSERT INTO Cheval (ChevalId, ChevalNom, DateNaissance) VALUES (1, 'Blanche Neige du Brésil', '2017-02-25');

SELECT * FROM Cheval WHERE ChevalId = 1;
```

Du code SQL (une instruction se termine par un point-virgule)

--- FIN DU DOCUMENT ---
http://www.arfp.asso.fr