HELHa
Haute École Louvain en Hainaut

Syllabus

Catégorie Économique

Fichiers et bases de données (UE204)

La conception du modèle de données

HELHA – Campus Mons – IRAM
Cours de Mr. Colmant
Année académique 2020-2021

1 Le recueil des données

Le Système d'Information (SI)

Voici la définition d'un système donnée par Jean-Louis Le Moigne, spécialiste de la systémique (in Algèbre relationnelle de Clouse) :

- un ensemble d'éléments identifiables avec leurs attributs,
- qui a une activité ou une fonction,
- qui est doté d'une structure,
- qui évolue dans le temps,
- qui est inséré dans un environnement et qui a une frontière ; les éléments des autres systèmes pouvant être affectés par ce système ou l'affecter eux-mêmes,
- qui a une finalité (c'est-à-dire des objectifs),
- le concept de totalité (pas juste la somme des parties mais les relations permettent d'obtenir un tout),
- un réseau d'informations et de communications,
- des entrées (input) et des sorties (output).

Un système peut être fermé (nombre fini d'états et les événements extérieurs n'ont pas d'impact sur lui) ou ouvert (s'adapte aux influences externes et possède donc un nombre infini d'états).
On distingue les données (atomiques) des informations (regroupement de données).
Une donnée est statique, peut être sauvegardée ou détruite après les traitements. Une information est obtenue en regroupant des données brutes (ex : un nom, un prix,...) et de données issues de traitement (ex : prix total d'une commande).
Dans les entreprises, le système d'information est basé sur les bases de données qui doivent donc être cohérentes et posséder des relations normalisées.

Le dictionnaire des données

Pour recueillir les données, il faut établir quelles sont les données entrantes (ex : bon de commande), sortantes (ex : bon de livraison) mais aussi internes (utiles au fonctionnement) (ex : client) et la circulation des données et la plus-value qu'elle amène (traitement du bon de commande avec articles indisponibles,...).
Il faut ensuite éliminer toute redondance : attributs ayant la même utilité (nécessite de vérifier qu'elles sont bien identiques mais n'ont pas forcément le même nom) et attributs calculables (tous les attributs qui peuvent être retrouvés à partir d'un autre attribut). Les attributs calculables pourront éventuellement être conservés s'ils sont souvent utilisés dans les traitements dans le but d'améliorer les performances (nécessite une analyse des traitements).
Il faut aussi donner des noms différents à des données homonymes qui représentent des concepts différents (ex : ville de livraison et ville de facturation).
Les données élémentaires doivent être les plus atomiques possibles (non-décomposables) mais sans aller dans un niveau de détail trop important (ex : l'adresse peut être décomposé en rue, numéro, boîte, code postal et ville mais cela n'aura pas forcément beaucoup de sens de décomposer un numéro de téléphone en préfixe et autre suite de numéros).
Les informations doivent correspondre à des objets du monde réel (on parle d'entité) et chacune de

leurs occurrences doivent pouvoir être identifiées de manière unique. Il faudra donc définir un identifiant pour chaque entité qui devra être unique.
Il est possible d'ajouter des entités supplémentaires pour faciliter les recherches futures d'information sur certaines entités (ex : ajouter une catégorie pour les produits dans un catalogue de produits).
On va donc pouvoir construire un dictionnaire de données qui contiendra pour chaque propriété/donnée (pas encore de regroupement en entité) :

- un numéro (unique) ;
- un nom interne dans le SI (mnémonique) ;
- une description de la donnée ;
- un type (numérique, alphanumérique, date, booléen,...) ;
- un format (nombre maximum de caractères) ;
- une catégorie (identifiant : I ou simple propriété : S).

Numéro	Nom	Description	Type	Format	Catégorie
1	CODECLI	Code client	Numérique	4	I
2	NOMCLI	Nom du client	Alphanumérique	50	S
3	PRENOMCLI	Prénom du client	Alphanumérique	50	S

2 Les dépendances fonctionnelles

La création de la matrice des dépendances fonctionnelles (MDF)

Après le dictionnaire des données, on va établir les dépendances fonctionnelles (entre des propriétés d'une même entité). On parle de dépendance fonctionnelle si la connaissance de la valeur d'une donnée source permet de n'identifier qu'une seule valeur pour une donnée cible. On marque cela de la manière suivante : source → cible. Exemple : le code client permet de retrouver un seul nom de client, donc : CODECLI → NOMCLI. Par contre, le nom du client ne donne pas qu'un seul code client (il peut y avoir des homonymes) donc il n'y a pas de dépendance fonctionnelle de NOMCLI vers CODECLI. On pratique de la sorte pour toutes les données identifiées dans le système d'information.
Les identifiants sont à la base des dépendances fonctionnelles.
On va pouvoir créer une matrice des dépendances fonctionnelles (MDF) en plaçant toutes les données du dictionnaire des données en tant que source dans les colonnes ainsi qu'en tant que cible dans les lignes (car toute propriété peut être la source ou la cible d'une dépendance fonctionnelle). On placera la numéro de chaque donnée avec éventuellement son nom interne. Une colonne est ajoutée à la fin de la matrice qui permettra de comptabiliser le nombre de dépendances fonctionnelles dont la donnée en ligne est la cible.

	Numéro de la source	Numéro de la source	Numéro de la source	Total
Numéro de la cible				
Numéro de la cible				
Numéro de la cible				

Le remplissage de la MDF

On placera une étoile dans chaque case de la diagonale car chaque donnée cause une dépendance fonctionnelle sur elle-même.
Ensuite, on inscrira un 1 dans chaque case dont la donnée en colonne est source de la donnée cible en ligne.

	1 CODECLI	2 NOMCLI	3 PRENOMCLI	4 NOFACT	Total
1 CODECLI	*			1	
2 NOMCLI	1	*			
3 PRENOMCLI	1		*		
4 NOFACT				*	

Dès que la matrice est complétée, on peut la simplifier en supprimant toutes les colonnes qui ne sont sources d'aucune donnée. On va également remplir la colonne total en y plaçant le total de l'addition de tous les '1' de la ligne.

	1 CODECLI	4 NOFACT	Total
1 CODECLI	*	1	1
2 NOMCLI	1		1
3 PRENOMCLI	1		1
4 NOFACT		*	0

Les manipulations dans la MDF

Si le total est égal à 0, il n'y a rien à faire si la donnée est source de dépendances fonctionnelles. Par contre, si la donnée n'est la source d'aucune dépendance fonctionnelle (ni la cible donc car le total est de 0), il faut se demander si la donnée est bien nécessaire car elle est esseulée (soit on lui trouve une dépendance fonctionnelle en approfondissant la réflexion soit on ne lui trouve pas d'utilité et on la supprime).

Il est possible qu'une donnée soit la cible non pas d'une seule donnée source mais d'un ensemble de données sources. Pour représenter cela dans notre matrice, il faudrait ajouter une nouvelle donnée (nouveau numéro avec le nom dans le SI qui contiendra les noms des différentes données concernées avec un '+' entre chaque nom).
Prenons l'exemple d'un client qui réserve une chambre pour plusieurs nuits à une certaine date. Le nombre de nuits ne dépend pas du code client car un client peut effectuer plusieurs réservations d'une chambre ni du numéro de la chambre car une chambre peut être réservée par plusieurs clients mais bien par la combinaison du code client, du numéro de la chambre et de la date de réservation (un client peut réserver plusieurs fois la même chambre). Voici ce que donne la matrice des dépendances fonctionnelles (extrait) :

	1 CODECLI	10 NUMCHAMB	11 DATERES	12 NBNUITS	13 CODECLI + NUMCHAMB + DATERES	Total
1 CODECLI	*					0
10 NUMCHAMB		*				0
11 DATERES			*			0
12 NBNUITS				*	1	1
13 CODECLI + NUMCHAMB + DATERES					*	0

Pour les données dont le total est supérieur à 1, cela signifie qu'il y a une transitivité dans les dépendances fonctionnelles et donc une redondance de l'information : A → B, B → C et A → C. Dans ce cas, on va supprimer le chemin le plus court et garder les dépendances intermédiaires (donc ici A → B et B → C). On crée ce qu'on appelle une entité paramètre qui va servir à supprimer cette redondance.
Prenons l'exemple d'un client qui habite dans une ville dont on connaît le code postal et dans un pays. Un code postal permet d'identifier le pays dans lequel on se trouve (ajout du B pour Belgique, du F pour France,...).

	1 CODECLI	2 CODEPOST	3 PAYS	Total
1 CODECLI	*			0
2 CODEPOST	1	*		1
3 PAYS	1	1	*	2

On se retrouve donc avec les dépendances suivantes : CODECLI → CODEPOST, CODEPOST → PAYS, CODECLI → PAYS. Il y a donc redondance de l'information car on peut savoir à quel pays appartient un client sur base de son code postal. On va donc créer une nouvelle entité paramètre qui regroupera la code postal et le pays et on pourra supprimer la dépendance fonctionnelle entre CODECLI et PAYS.

	1 CODECLI	2 CODEPOST	3 PAYS	Total
1 CODECLI	*			0
2 CODEPOST	1	*		1
3 PAYS		1	*	1

Les colonnes restantes dans la matrice des dépendances fonctionnelles représentent donc les identifiants des différentes entités.

3 Modèle conceptuel de données (MERISE)

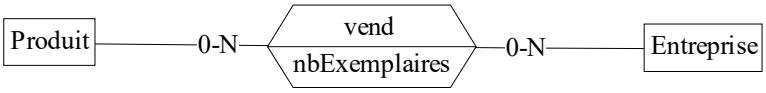
Merise propose une méthode d'analyse basée sur une séparation des données et des traitements. En effet, des modifications dans les traitements n'ont pas toujours d'impact dans les données et inversement.
Le niveau conceptuel de Merise permet de répondre à QUOI ? On ne tient pas compte ici des moyens pour arriver à une solution. On construit deux modèles : le MCD (Modèle Conceptuel de Données) et le MCT (Modèle Conceptuel des Traitements).
Le niveau organisationnel permet de répondre à COMMENT ? OÙ ? QUI ? QUAND ? On crée ici le MOD (Modèle Organisationnel des Données) et le MOT (Modèle Organisationnel des Traitements).
Le niveau logique permet de répondre à QUELS MOYENS ? Le MLD (Modèle Logique des Données) et le MLT (Modèle Logique des Traitements) sont alors conçus pour répondre à cette question.
Le niveau physique consiste en la construction de la solution par le MPD (Modèle Physique des Données) et le MPT (Modèle Physique des Traitements).
Dans ce cours, nous nous concentrerons sur les modèles de données. Il ne faut toutefois occulter le fait que les données seront aussi influencées par les traitements (notamment en terme de performance et de redondance des informations, dans le choix des index,...).

Le schéma entité-association (MCD)

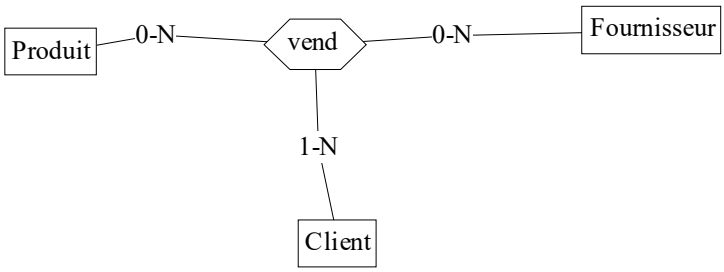
Une entité est un tout cohérent défini par des propriétés et possédant un identifiant (qui permet de rendre unique chacune de ses instances).
L'entité possède un nom et des propriétés (ou champs).
Une entité se représente par un rectangle en deux parties. La première reprend le nom (généralement un nom commun) de l'entité et la deuxième reprend les propriétés de celle-ci. L'identifiant sera souligné ou on place id(entité) = attribut_identifiant (on peut éventuellement en mettre plusieurs en les séparant par des virgules).
Il est obligatoire de préciser si un attribut est facultatif en mettant [0..1] à côté de son nom. S'il est multi-valué, on indiquera le nombre minimum et le nombre maximum d'occurrences, ex : [1..5].

Entite
id prop1 prop2 [0..1]

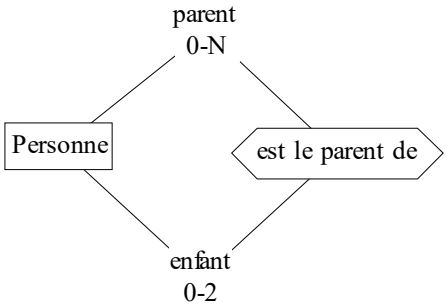
Une association relie 2 ou plusieurs entités et peut posséder des propriétés. Il est vivement conseillé de lui donner un nom (généralement un verbe).



On précise des cardinalités pour chaque entité concernée par l'association. On précise toujours une cardinalité minimale et une cardinalité maximale. Les cardinalités précisent le nombre d'occurrences (instances) de l'entité concernée par l'association. Le choix de 0 ou 1 dans la cardinalité minimale aura une influence sur le stockage des données. En effet, si la cardinalité est à 0, une occurrence de l'entité concernée pourrait n'avoir aucune association vers l'autre entité. Par contre, si la cardinalité est à 1, cela signifie que toute occurrence de l'entité devra avoir une association avec l'autre entité !
Un rôle peut également être ajouté à chaque entité dans l'association.
Une association peut également être n-aire et mettre en relation plus de 2 entités. Voici ce que donne une association ternaire :



Si une des cardinalités de l'association ternaire est de maximum 1 alors l'association ternaire doit être remplacée par une association binaire.
Une association peut être cyclique si elle ne concerne qu'une seule entité. Il faut alors préciser les rôles.



On parle de contrainte d'intégrités multiples (CIM) lorsque toutes les cardinalités maximales de l'association sont à N.

La transformation de la matrice des dépendances fonctionnelles en schéma entité-association

Les entités sont construites à partir de la matrice des dépendances fonctionnelles. On crée une entité pour chaque identifiant (colonne restante dans la MDF composée d'une seule donnée) et on y place comme propriété la donnée source de la dépendance et ses données cibles. La donnée source de la dépendance sera l'identifiant de l'entité.
Dans le cas des colonnes avec une combinaison de données, il y a deux cas possibles. Si la colonne

reprend 0 ou une colonne qui est source de dépendances fonctionnelles alors la colonne devient une entité. Si par contre la colonnes compte au moins 2 données sources de dépendances fonctionnelles alors la colonne devient une CIM entre les entités identifiées par les données sources de dépendances fonctionnelles.
Si on reprend la MDF, les données sources complexes (issues du regroupement de plusieurs données) forment des associations CIM et leurs données cibles sont les propriétés de cette association. La donnée source complexe sera l'identifiant de l'association.
Les associations simples seront alors construites sur base des demandes formulées par le MOA.

Exemple 1 : Une voiture dispose d'un seul propriétaire

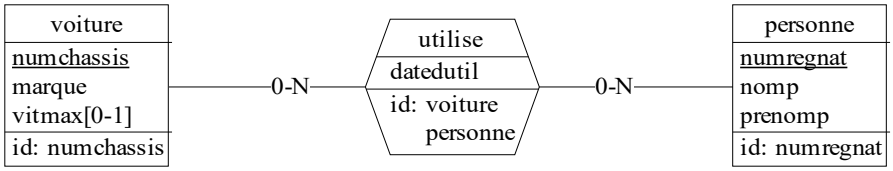
	NUMCHASSIS	NUMREGNAT	Total
NUMCHASSIS	*		0
MARQUE	1		1
VITMAX	1		1
NUMREGNAT	1	*	1
NOMP	0	1	1
PRENOMP	0	1	1

Il n'y a que 2 colonnes avec une simple donnée source donc il y aura 2 entités. Les attributs présents dans chacune des entités seront les lignes avec des 1 pour la colonne concernée **à l'exception** des lignes représentant des données sources de dépendances fonctionnelles (avec une colonne contenant 1 ou plusieurs « 1 »). La donnée en colonne doit aussi être placée dans les attributs et est l'identifiant de l'entité. Ces entités seront reliées par une association permettant de préciser quelle personne est propriétaire de quel voiture.

Exemple 2 : Une voiture est utilisée par plusieurs personnes (il faut retenir la date de dernière utilisation)

	NUMCHASSIS	NUMREGNAT	NUMCHASSIS+NUMREGNAT	Total
NUMCHASSIS	*		1	1
MARQUE	1		0	1
VITMAX	1		0	1
NUMREGNAT		*	1	1
NOMP		1	0	1
PRENOMP		1	0	1
DATEDUTIL			1	1

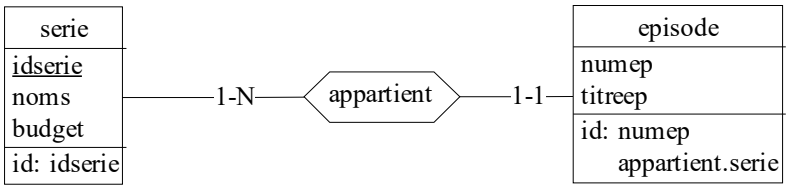
Les 2 premières colonnes sont des entités (voir exemple 1). La dernière colonne est composée de plusieurs données sources de dépendances fonctionnelles. Il faudra donc placer une association N à N entre les 2 entités représentées par les données. Seules les lignes avec des 1 qui ne sont pas sources de dépendances fonctionnelles seront présents comme attribut de l'association.



Exemple 3 : Une série télévisée compte plusieurs épisodes

	IDSERIE	IDSERIE+NUMEP	Total
IDSERIE	*	1	1
NOMS	1	0	1
BUDGET	1	0	1
NUMEP		1	1
TITREEP		1	1

Dans cet exemple-ci, la dernière colonne ne dispose que d'une seule donnée source de dépendances fonctionnelles. Cette colonne deviendra donc une nouvelle entité qui sera identifiée par l'attribut « numep » + son association avec l'entité « episode ».



Le MOD

Le MOD (modèle organisationnel des données) permet d'établir les droits d'accès de chaque utilisateur dans les différentes entités (consultation, mise à jour) et précise les données qui peuvent être modifiées par chaque utilisateur.

Le MLD

Pour le MLD (modèle logique des données), on va ici choisir d'utiliser les bases de données relationnelles (il existe aussi des datawarehouses ainsi que des bases de données orientées objet notamment).
Il va donc falloir transformer notre schéma entité-association en un schéma relationnel.
Les entités sont transformées en tables. Les propriétés deviennent des colonnes de table. Les occurrences seront appelées des lignes de la table.
L'identifiant constituera la clé primaire de la table.
Pour assurer ces transformations, il va falloir suivre quelques règles :

1. Toute entité est transformée en table dont le nom correspond à celui de l'entité, dont la clé primaire est l'identifiant et où les colonnes sont formées des autres propriétés.
2. Les associations d'intégrités multiples (CIM) doivent être transformées en tables relationnelles en mettant le nom de l'association comme nom de la table (on utilise généralement un nom plutôt qu'un verbe pour le nom d'une table), où la clé primaire sera la concaténation des clés primaires des entités concernés par l'association et où les autres propriétés formeront les colonnes de la table.
3. Pour les autres associations (de type 0,1 ou 1,1 à 0,N ou 1,N), la transformation se fera par l'ajout d'une colonne, dans la table avec la cardinalité maximale de 1, qui correspond à la clé primaire de l'autre table. On parle alors de clé étrangère.
4. Les associations où les cardinalités maximales sont toutes égales à 1 vont donner une fusion des deux entités en une seule table contenant toutes les propriétés dans les colonnes (en

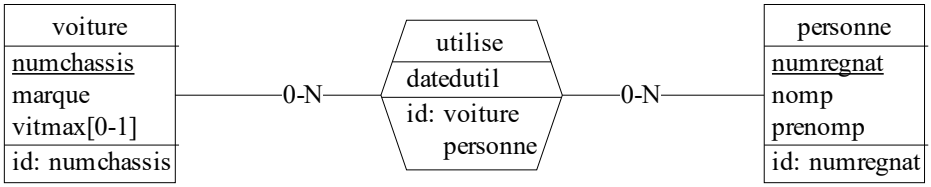
supprimant les éventuelles redondances). La clé primaire sera choisie parmi les identifiants des entités concernées.

On représente les tables de la manière suivante : Nom_table (cle_primaire, colonne1, colonne2,...).
Il faut encore également normaliser nos tables avant d'avoir notre schéma final de base de données relationnelle (chapitre suivant).

Exemple 1 : Une voiture dispose d'un seul propriétaire

Les 2 entités deviennent chacune une table. Les attributs multivalués (ici prenomp) doivent être décomposés en autant de colonnes qu'indiquées par la cardinalité maximale (ici 3). Certaines de ces colonnes sont facultatives (ici les colonnes 2 et 3). Il faut rajouter une clé étrangère dans voiture pour relier la table à personne (la clé étrangère se met toujours du côté de la cardinalité maximale à 1).
personne(numregnat, nomp, prenomp1, prenomp2[0-1], prenomp3[0-1])
voiture(numchassis, marque, vitmax[0-1], numregnat[0-1])

Exemple 2 : Une voiture est utilisée par plusieurs personnes (il faut retenir la date de dernière utilisation)



Ici, l'association N à N doit devenir une nouvelle table qui contiendra, en plus de ses attributs, les clés étrangères vers les tables qu'elle relie.
personne(numregnat, nomp, prenomp)
voiture(numchassis, marque, vitmax[0-1])
utilise(dateutil, numregnat, numchassis)

Le MPD

Le MPD (modèle physique de données) traduit le MLD en instructions SQL correspondant à l'outil utilisé (MySQL, DB2, Oracle,...). Les exemples donnés ici correspondent à MySQL, il peut donc être nécessaire de faire quelques adaptations pour un autre outil. Remarque pour les instructions SQL : les mots clés du langage sont mis en majuscules (il n'est pas nécessaire de les mettre en majuscule dans les outils) et les instructions entre [ ] sont facultatives.

La création de la base de données

La première étape est de créer la base de données dans laquelle seront placées toutes les tables identifiées dans le MLD. Pour créer la base de données, on place : CREATE DATABASE Nom_baseDeDonnees ;

Exemple : CREATE DATABASE IF NOT EXISTS 'BandesDessinees' ;
Pour créer les tables, on va également avoir besoin du dictionnaire des données car il reprend le type de chaque donnée (donc les colonnes de nos tables).

La création des tables

Pour créer une table, on utilise : CREATE TABLE [IF NOT EXISTS] Nom_table (...) ;
On place toutes les colonnes dans les parenthèses en précisant le nom, le type, la taille et également si le champ ne peut pas être vide (not null) : colonne1 type(taille) [NOT NULL]. Les colonnes sont séparées par des virgules. Si une colonne est un auto-incrément (incrémenté automatiquement à chaque nouvelle entrée), on met AUTO_INCREMENT.
La clé primaire est signalée grâce à PRIMARY KEY et les autres colonnes qui sont indexées sont marquées par KEY (le nom qui suit est le nom donné à l'index). Les clés étrangères doivent être indexées.
Exemple : CREATE TABLE IF NOT EXISTS 'album' ('id' int(8) NOT NULL AUTO_INCREMENT, 'numero' int(3) NOT NULL, 'titre' varchar(200) NOT NULL, 'id_serie' int(4) NOT NULL, PRIMARY KEY ('id'), KEY 'id_serie' ('id_serie')) ;
Les types de données qui peuvent être utilisés pour les colonnes (non exhaustif, voir http://dev.mysql.com/doc/refman/5.7/en/create-table.html et http://dev.mysql.com/doc/refman/5.7/en/data-types.html pour la description des types) :

- INT : entier long
- SMALLINT : entier court
- DECIMAL : nombre réel (fonctionne comme deux entiers séparés par une virgule) où il faut préciser la longueur et le nombre de décimales). Ex : DECIMAL(10,3)
- FLOAT : nombre réel de plus grande précision, utilisé pour les calculs scientifiques
- CHAR : chaîne de caractères de longueur fixée
- VARCHAR : chaîne de caractères de longueur maximale fixée
- DATE : stocke la date sous format AAAA-MM-JJ
- TIME : stocke l'heure sous format HH:MM:SS
- DATETIME : stocke la date sous format AAAA-MM-JJ HH:MM:SS

Il est possible de créer de nouveaux types (on parle de domaine), en utilisant CREATE DOMAIN nom_domaine AS type_existant. Exemple : CREATE DOMAIN NOMBREPRECIS AS DECIMAL(10,4). Le nouveau type peut alors être utilisé dans la définition des colonnes.

L'ajout des contraintes

Les clés étrangères s'ajoutent à la suite des tables en utilisant la syntaxe suivante : ALTER TABLE tableAvecCleEtrangere ADD CONSTRAINT nom_contrainte FOREIGN KEY (cle_etrangere) REFERENCES tableAvecId (id_table) ;
Exemple : ALTER TABLE 'album' ADD CONSTRAINT 'id_albumserie' FOREIGN KEY ('id_serie') REFERENCES 'serie' ('id') ;
L'ajout d'une clé primaire peut également se faire de la manière suivante (après la déclaration de la table) : ADD CONSTRAINT nom_contrainte PRIMARY KEY(cle_primaire) ;
Il est également possible d'ajouter des colonnes après avoir construit une table en utilisant ALTER TABLE table ADD COLUMN nom_colonne type(taille) [NOT NULL] [FIRST | AFTER nom_colonne].

Exemple 1 : Une voiture dispose d'un seul propriétaire
personne(numregnat, nomp, prenomp1, prenomp2[0-1], prenomp3[0-1])
voiture(numchassis, marque, vitmax[0-1], numregnat[0-1])
Les champs obligatoires seront déclarés NOT NULL alors que les champs facultatifs ([0-1]) ne se verront pas mettre de mention complémentaire. Les types de données devront correspondre à ceux du dictionnaire des données.
Les clés étrangères doivent être déclarées dans les champs de la table où elles sont ajoutées (le type doit être le même que celui du champ lié dans l'autre table mais le nom peut être différent) et être indexées. Si une clé étrangère est placée en premier élément dans une clé primaire, alors cette clé étrangère est automatiquement indexée. Pour indexer une colonne, il faut utiliser l'instruction KEY(nomcolonne_aindexer). Les contraintes de clés étrangères ne peuvent être ajoutées que quand toutes les tables liées par la clé étrangère ont déjà été créées.

```
CREATE TABLE personne (
numregnat CHAR(11) NOT NULL,
nomp VARCHAR(50) NOT NULL,
prenomp1 VARCHAR(50) NOT NULL,
prenomp2 VARCHAR(50),
prenomp3 VARCHAR(50),
PRIMARY KEY(numregnat)) ENGINE=INNODB ;

CREATE TABLE voiture (
numchassis CHAR(20) NOT NULL,
marque VARCHAR(60) NOT NULL,
vitmax NUMERIC(5,2),
numregnat CHAR(11),
PRIMARY KEY(numchassis),
KEY(numregnat)) ENGINE=INNODB ;

ALTER TABLE voiture ADD CONSTRAINT fkvoiturepersonne FOREIGN KEY (numregnat)
REFERENCES personne(numregnat) ;
```

Exemple 2 : Une voiture est utilisée par plusieurs personnes (il faut retenir la date de dernière utilisation)
personne(numregnat, nomp, prenomp)
voiture(numchassis, marque, vitmax[0-1])
utilise(dateutil, numregnat, numchassis)
Dans ce cas-ci, la table « utilise » comporte 2 clés étrangères. Comme la clé primaire est constituée de la combinaison de ces 2 clés étrangères, il ne faudra plus indexer que la clé étrangère qui sera placée en seconde position dans la déclaration de la clé primaire.

```
CREATE TABLE personne (
numregnat CHAR(11) NOT NULL,
nomp VARCHAR(50) NOT NULL,
prenomp1 VARCHAR(50) NOT NULL,
prenomp2 VARCHAR(50),
prenomp3 VARCHAR(50),
PRIMARY KEY(numregnat)) ENGINE=INNODB ;

CREATE TABLE voiture (
numchassis CHAR(20) NOT NULL,
```

```
marque VARCHAR(60) NOT NULL,
vitmax NUMERIC(5,2),
PRIMARY KEY(numchassis)) ENGINE=INNODB ;

CREATE TABLE utilise (
dateutil DATETIME NOT NULL,
numregnat CHAR(11) NOT NULL,
numchassis CHAR(20) NOT NULL,
PRIMARY KEY(numregnat, numchassis),
KEY(numchassis)) ENGINE=INNODB ;

ALTER TABLE utilise ADD CONSTRAINT fkutilisepersonne FOREIGN KEY (numregnat)
REFERENCES personne(numregnat) ;
ALTER TABLE utilise ADD CONSTRAINT fkutilisevoiture FOREIGN KEY (numchassis)
REFERENCES voiture(numchassis) ;
```

4 La normalisation

Les objectifs de la normalisation sont les suivants : regrouper les attributs strictement nécessaires dans les relations (sur base des dépendances fonctionnelles) et ne laisser que les redondances indispensables (clés étrangères).
On doit pouvoir considérer que les règles suivantes doivent fonctionner : la réflexivité (une donnée est la source d'elle-même), l'augmentation (si une donnée cible dépend fonctionnellement d'une donnée source alors si on ajoute une même donnée aussi bien à la source et qu'à la cible, la dépendance fonctionnelle reste vraie), la transitivité (si A1 → A2, A2 → A3 alors A1 → A3), la décomposition sans perte (si il y a décomposition d'une relation en plusieurs relations alors la jointure naturelle de toutes ces relations donne le même résultat que la relation de départ).
Toute relation doit être normalisée. Pour normaliser une relation, on passe successivement par les 6 formes normales (de la 1ère à la 5ème avec une 3ème forme normale adaptée) en s'assurant que les conditions mentionnées sont bien remplies (on ne passe pas à la forme normale suivante tant que ce n'est pas le cas). Si une relation ne répond pas aux critères d'une forme normale, elle doit être décomposée en plusieurs relations (qui devront chacune vérifier les 6 formes normales). Dans ce cours, nous aborderons les 3 première formes normales ainsi que celle de Boyce-Codd. Les 2 dernières sont données ici à titre d'information.

La première forme normale (1FN)

Une relation est en première forme normale si, pour chaque tuple de la relation, chaque attribut contient une valeur atomique (autrement dit, il ne peut y avoir d'attribut multi-valué ni d'attribut composé). Exemple pour l'attribut multi-valué : plusieurs noms d'auteurs pour un livre, pour l'attribut composé : une adresse (avec rue, numéro, code postal, ville). Pour pouvoir parler d'un attribut composé, il faut que chaque élément de la composition pris individuellement ait un sens (ex : pour une date, chacune des parties ne donne pas une information). Les attributs composés sont alors décomposés afin d'obtenir des attributs atomiques.
Pour être en 1FN, il faut également que tous les attributs non-clés de la relation soient dépendants fonctionnellement de l'attribut clé de la relation.

La deuxième forme normale (2FN)

Une relation est en deuxième forme normale seulement si elle est en première forme normale et que tout attribut non-clé de la relation ne dépend pas que d'une partie de la clé de la relation. Il ne faudra vérifier cette forme normale que si le relation possède comme clé une composition d'attributs. Les relations dont la clé ne possède qu'un seul attribut sont en 2FN si elles sont en 1FN.
Si une relation n'est pas en 2FN, cela signifie qu'au moins une nouvelle relation devra être créée avec les attributs qui ne dépendent que d'une partie de la clé.

La troisième forme normale (3FN)

Une relation est en troisième forme normale si elle est en 2FN et que tout attribut non-clé de la relation ne dépend pas d'un attribut non-clé de la relation. Si une relation ne possède qu'un attribut non-clé alors elle est 3FN si elle est en 2FN. Il faut donc faire la vérification seulement pour les relations comportant au moins 2 attributs non-clés.

La forme normale de Boyce-Codd (FNBC)

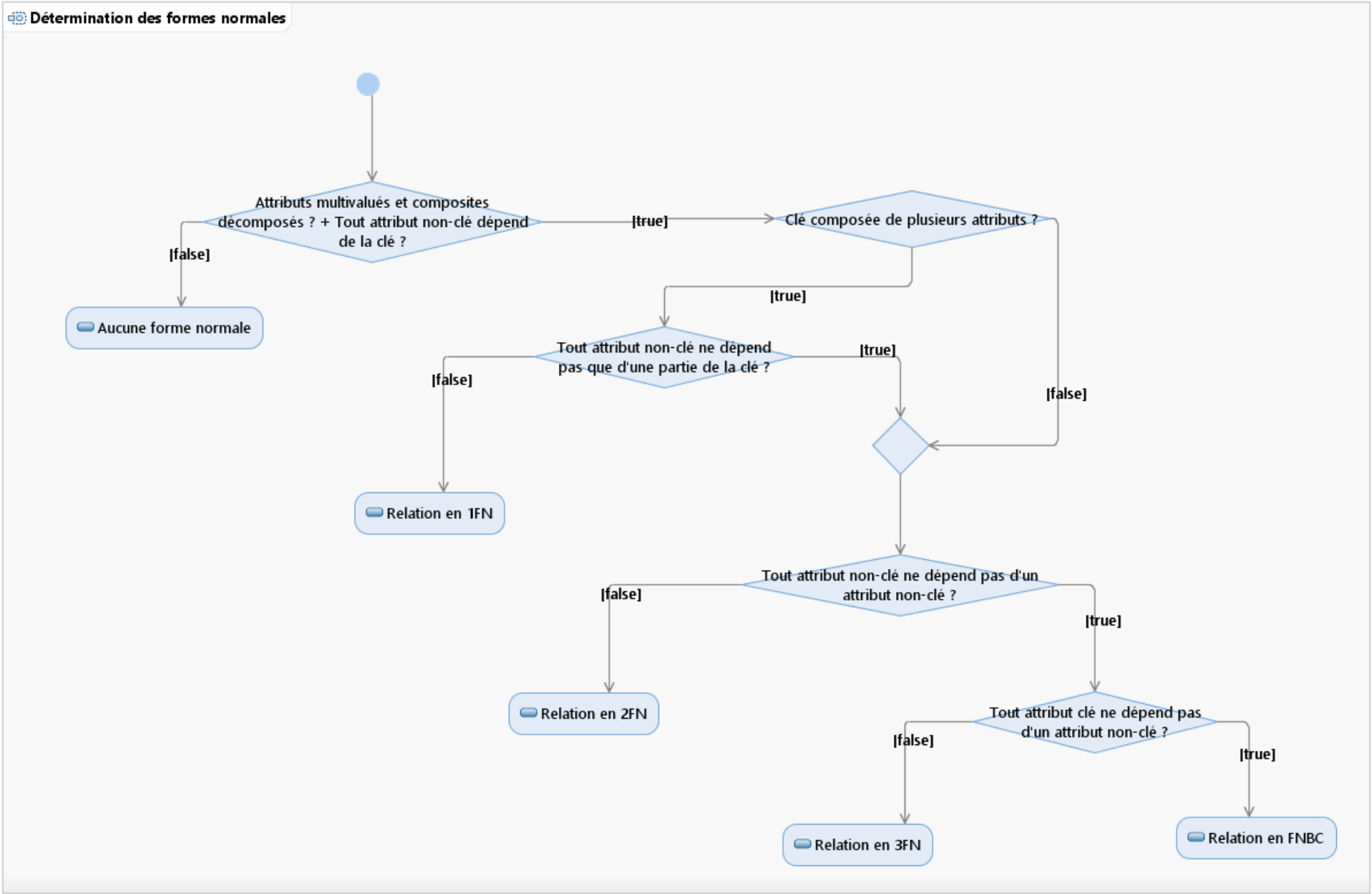
Une relation est en forme normale de Boyce-Codd si elle est en 3FN et qu'aucun attribut faisant partie de la clé de la relation n'est dépendant fonctionnellement d'un autre attribut non-clé de la relation. Cette forme normale ne doit donc être vérifiée que dans le cas des clés primaires multi-attributs.

La quatrième forme normale (4FN)

Une relation est en quatrième forme normale si elle est en FNBC et elle ne contient aucune DM non triviale. Une dépendance multivaluée (DM) est une dépendance qui fait qu'un attribut détermine des valeurs d'un autre attribut et que ceci est indépendant des autres attributs. On dit alors que le premier attribut multi-détermine le second : A →→ B.
Ex : Un type de produits détermine les produits qui en font partie et un type de produits détermine quels employés travaillent sur celui-ci. La relation donne TRAVAIL_EMPLOYE<TYPEPROD, NUMPROD, NUMEMPL>. Cette relation est en FNBC mais TYPEPROD →→ NUMPROD car un produit ne fait partie que d'un type de produits et TYPEPROD →→ NUMEMPL car un employé travaille sur un type de produits. Lorsqu'il faut ajouter un employé, il faut donc ajouter autant de tuples qu'il y a de produits dans le type de produits.
La dépendance multivalué est non-triviale si le premier attribut n'est pas un sous-ensemble du deuxième et que l'union des deux attributs ne permet pas de retrouver toute la relation. Elle est triviale dans le cas contraire.
Les DM non-triviales sont remplacées par de nouvelles relations. La relation précédente donnerait donc TYPE_PRODUIT<TYPEPROD, NUMPROD> et TYPE_EMPL<TYPEPROD, NUMEMPL>.

La cinquième forme normale (5FN)

Une relation est en cinquième forme normale si elle ne possède aucune dépendance de jointure. Il y a une dépendance de jointure si une relation en extension possède des redondances et que sa décomposition en plus de 2 relations nécessite la jointure des projections pour récupérer la totalité de la relation de départ.
Exemple : un client achète un produit à un vendeur ACHAT<NUMCLI, NUMPROD, NUMVENDEUR>. Cette relation possède des redondances. La relation deviendra CLIPROD<NUMCLI, NUMPROD> (un client achète des produits), VENDPROD<NUMVENDEUR, NUMPROD> (un vendeur vend des produits) et CLIVEND<NUMCLI, NUMVENDEUR> (un client achète chez un vendeur). Pour reconstruire la relation initiale, il faut utiliser les jointures entre les relations, la relation de départ n'était donc en 5FN et la relation doit être décomposée en plusieurs relations.


Détermination des formes normales
Attributs multivalués et composites décomposés ? + Tout attribut non-clé dépend de la clé ?
[false]
[true]
Aucune forme normale
Clé composée de plusieurs attributs ?
[true]
[false]
Tout attribut non-clé ne dépend pas que d'une partie de la clé ?
[false]
[true]
Relation en 1FN
Tout attribut non-clé ne dépend pas d'un attribut non-clé ?
[false]
[true]
Relation en 2FN
Tout attribut clé ne dépend pas d'un attribut non-clé ?
[false]
[true]
Relation en 3FN
Relation en FNBC