librecours.net

Conception des bases de données I : Introduction

bdd1.pdf

utc Formation

Stéphane Crozat

Publiée le 16 février 2020

Table des matières

I

Introduction générale aux bases de données

A. Cours

Les BD sont nées à la fin des années 1960 pour combler les lacunes des systèmes de fichiers et faciliter la gestion qualitative et quantitative des données informatiques. Les SGBD sont des applications informatiques permettant de créer et de gérer des BD (comme Oracle ou PostgreSQL par exemple)

Les BD relationnelles, issues de la recherche de Codd, sont celles qui ont connu le plus grand essor depuis les années, et qui reste encore aujourd'hui les plus utilisées. On utilise des SGBDR pour les implémenter. Le langage SQL est le langage commun à tous les SGBDR, ce qui permet de concevoir des BD relativement indépendamment des systèmes utilisés.

Les usages de BD se sont aujourd'hui généralisés pour entrer dans tous les secteurs de l'entreprise, depuis les petites bases utilisées par quelques personnes dans un service pour des besoins de gestion de données locales, jusqu'aux bases qui gèrent de façon centralisée des données partagées par tous les acteurs de l'entreprise.

L'accroissement de l'utilisation du numérique comme outil de manipulation de toutes données (bureautique, informatique applicative, etc.) et comme outil d'extension des moyens de communication (réseaux) ainsi que les évolutions technologiques (puissance des PC, Internet, etc.) ont rendu indispensable, mais aussi complexifié la problématique des BD.

Les conséquences de cette généralisation et de cette diversification des usages se retrouvent dans l'émergence de solutions conceptuelles et technologiques nouvelles, les bases de données du mouvement NoSQL particulièrement utilisées par les grands acteurs du web.

1. Notions fondamentales : donnée, base de données, système de gestion de bases de données

a) Base de données

Logiciel et données

Un logiciel informatique est composé de programmes, c'est à dire d'instructions données à l'ordinateur, et de données auxquelles s'appliquent ces instructions.

Par exemple un logiciel de traitement de texte est composé de fonctions - ouvrir, copier, coller, insérer une image, changer la police, enregistrer... - et de fichiers sur lesquels elles s'appliquent. Dans ce cas les fichiers de traitement de texte sont les données.

Définition : Définition lâche de base de données : un ensemble de données

On appelle parfois base de données tout ensemble de données stocké numériquement et pouvant servir à un ou plusieurs programme. De ce point de vue des fichiers sur un disque dur, un fichier de tableur, voire un fichier de traitement de texte peuvent constituer des bases de données.

Définition : Définition restreinte de base de données : un ensemble de données structuré

On appellera base de données un ensemble de données numériques qui possède une structure ; c'est à dire dont l'organisation répond à une **logique** systématique.

On parlera de modèle logique de données pour décrire cette structure.

Exemple : Base de données relationnelle

Une base de données relationnelle permet d'organiser les données en tableaux (appelés relations).

espèce	eucaryote	multicellulaire	propriété
bactéries	false	false	
archées	false	false	
protistes	true	false	
champignons	true	true	décompose
végétaux	true	true	photosynthétise
animaux	true	true	ingère

Tableau 1 Base de données de classification classique des espèces animales

Fondamental : Fonctions d'une base de données

Une base de données est structurée afin de pouvoir mieux répondre à des fonctions fondamentales en informatique, telles que :

- Stocker l'information de façon fiable (c'est à dire être capable de restituer l'information entrée dans le système)
- Traiter de grands volumes de données (massification)

- Traiter rapidement les données (optimisation)
- Sécuriser les accès aux données (gérer les autorisations selon les utilisateurs)
- Contrôler la qualité des données (par exemple la cohérence par rapport à un modèle pré-établi)
- Partager les données (entre plusieurs applications dédiées à plusieurs métiers)
- Rendre accessible les données en réseau (gérer la concurrence des accès parallèles)
- ...

b) Système de gestion de bases de données (SGBD)

La conception d'un système d'information se doit d'adopter un certain nombre de principes, tels que :

- une description des données indépendante des traitements,
- une gestion automatique de la cohérence de données,
- le recours à des langages non procéduraux structurants.

Définition : Système de Gestion de Bases de Données

Un SGBD est un logiciel qui prend en charge la structuration, le stockage, la mise à jour et la maintenance d'une base de données. Il est l'unique interface entre les informaticiens et les données (définition des schémas, programmation des applications), ainsi qu'entre les utilisateurs et les données (consultation et mise à jour).

Exemple : Exemples de SGBD

- Oracle est un SGBD relationnel et relationnel-objet très utilisé pour les applications professionnelles.
- PostgreSQL est un SGBD relationnel puissant qui offre une alternative libre (licence BSD) aux solutions commerciales comme Oracle ou IBM.
- Access est un SGBD relationnel Microsoft, qui offre une interface graphique permettant de concevoir rapidement des applications de petite envergure ou de réaliser des prototypes.
- MongoDb est un SGBD non-relationnel libre (licence Apache) orienté document. Il permet de gérer facilement de très grandes quantités de données - dans un format arborescent JSON - réparties sur de nombreux ordinateurs.

Fondamental : Objectifs des SGBD

- **Indépendance physique des données**

  Le changement des modalités de stockage de l'information (optimisation, réorganisation, segmentation, etc.) n'implique pas de changements des programmes.
- **Indépendance logique des données**

  L'évolution de la structure d'une partie des données n'influe pas sur l'ensemble des données.
- **Manipulation des données par des non-informaticiens**

  L'utilisateur n'a pas à savoir comment l'information est stockée et calculée par la machine, mais juste à pouvoir la rechercher et la mettre à jour à travers des IHM ou des langages assertionnels simples.

- **Administration facilitée des données**
  Le SGBD fournit un ensemble d'outils (dictionnaire de données, audit, tuning, statistiques, etc.) pour améliorer les performance et optimiser les stockages.
- **Optimisation de l'accès aux données**
  Les temps de réponse et de débits globaux sont optimisés en fonctions des questions posées à la BD.
- **Contrôle de cohérence (intégrité sémantique) des données**
  Le SGBD doit assurer à tout instant que les données respectent les règles d'intégrité qui leurs sont imposées.
- **Partageabilité des données**
  Les données sont simultanément consultables et modifiables.
- **Sécurité des données**
  La confidentialité des données est assurée par des systèmes d'authentification, de droits d'accès, de chiffrement des mots de passe, etc.
- **Sûreté des données**
  La persistance des données, même en cas de panne, est assurée, grâce typiquement à des sauvegardes et des journaux qui gardent une trace persistante des opérations effectuées.

Complément

Pourquoi des SGBD ?

Complément : SGBD relationnel et non-relationnel

Les SGBR relationnels (SGBDR) sont les plus courants des SGBD ; jusqu'au début des années 2000, la plupart des bases de données étaient relationnelles.

Mais avec l'arrivée des géants du web, ces entreprises qui gèrent des quantités énormes de données comme Google, Amazon ou Facebook, s'est développé un mouvement important de développement de bases de données non-relationnelles, également appelées NoSQL.

c) Application de base de données

Une base de données seule n'est pas directement utilisable par un utilisateur humain ; elle n'est utilisable que par les informaticiens qui connaissent son langage de programmation et par les applications qui ont été programmées pour s'en servir.

Définition

On appelle application de base de données un logiciel informatique permettant à un utilisateur final de manipuler (lire ou écrire) les données d'une base de données.

Exemple : Application web

Une application web est composée d'interfaces en HTML qui permettent d'écrire et de lire des données dans une base de données, via un langage applicatif, comme par exemple PHP.

Exemple : Twitter

L'application Twitter est composée d'interfaces web permettant d'entrer des données (saisir son profil, *twitter*, *retwitter*, ...) et de sortir des données (consulter un fil *twitter*, faire une recherche sur un *hashtag*...) d'une base de données.

Cette base de données est stockée sur les serveurs de Twitter et elle contient tous les profils de tous les utilisateurs, tous les *tweets*, tous les *hashtags*...

Exemple : Compagnie aérienne

Une base de données de gestion de l'activité d'une compagnie aérienne concerne les voyageurs, les vols, les avions, le personnel, les réservations...

Une application à partir d'une telle base de données pourra permettre la gestion des réservations, des disponibilités des avions en fonction des vols à effectuer, des affectations des personnels volants...

Exemple : Application de bureau Access

Avec un logiciel comme Access on peut réaliser à la fois une base de données et une application permettant de manipuler cette base de données pour des besoins bureautiques simples.

d) Donnée (en relationnel) : table, objet, propriété, domaine, atomicité

Rappel : Base de données relationnelle

Une base de données relationnelle permet d'organiser les données en tables (appelés relations).

Chaque case de la table contient une information atomique.

Définition : Objet (ligne)

Chaque ligne de la table correspond à un objet que l'on veut gérer dans la base de données : une voiture, une personne, une espèce...

Fondamental

Toutes les lignes d'une même table correspondent à des objets du même type, donc dans une table, on met soit des voitures, soit des personnes, mais on ne mélange pas les deux.

Définition : Propriété et domaine (colonne)

Chaque colonne de la table correspond à une propriété des objets qui se trouvent dans la table ; tous les objets de la table partagent donc les mêmes propriétés.

Fondamental : Domaine

Chaque colonne de la table est associée à un domaine de valeur fixé **a priori**, par exemple : entier, texte, booléen...

Définition : Donnée en relationnel (cellule)

Une donnée en relationnel, c'est une cellule d'une table, qui correspond à la propriété d'un objet.

propriété 1 domaine : d1	**propriété 2 domaine : d2**	**...**
objet1, donnée 1	objet1, donnée 2	...
objet2, donnée 1	objet2, donnée 2	...
...	...	...

Tableau 2 Une table ou relation (en relationnel)

Exemple

espèce domaine : texte	**eucaryote domaine : booléen**	**...**
bactéries	false	...
archées	false	...
...	...	...

Tableau 3 Exemple de relation instanciée

Attention : Atomicité (contre-exemple)

Pour que la base de données fonctionne correctement on veille à ne mettre qu'une seule donnée par case, c'est le principe d'atomicité en relationnel.

espèce, domaine : texte
bactéries : procaryotes unicellulaires
archées : procaryotes unicellulaires
protistes : eucaryotes unicellulaires
champignons : eucaryotes multicellulaires qui décomposent
végétaux : eucaryotes multicellulaires qui photosynthétisent
animaux : eucaryotes multicellulaires qui ingèrent

Tableau 4 Un mauvais exemple de relation : les données ne sont pas atomiques (il y a plusieurs données par case de la table)

e) Langage de données : l'exemple du langage SQL

Définition : Langage de données

Un langage de données est un langage informatique permettant de décrire et de manipuler les schémas et les données d'une BD.

Synonymes : Langage orienté données

Fondamental : SQL

SQL est le langage consacré aux SGBD relationnels et relationnels-objet.

Il permet de :

- créer des tables, en définissant le domaine de chaque colonne ;
- insérer des lignes dans les tables
- lire les données entrées dans la base de données

Exemple : Création de table en SQL (définition du schéma de données)

```
1  CREATE TABLE etudiant (
2    numetu INTEGER PRIMARY KEY,
3   nom VARCHAR,
4    ville VARCHAR)
```

Cette instruction permet de créer une relation "etudiant" comportant les propriétés "numetu", "nom" et "ville" de domaines, respectivement, entier, texte et texte.

Exemple : Insertion de ligne en SQL (création de données)

```
1  INSERT INTO etudiant (numetu, nom, ville) VALUES (1, 'Holmes',
   'Londres')
```

Cette instruction permet de créer l'étudiant numéro 1, de nom Holmes qui habite la ville de Londres.

Exemple : Manipulation de données en SQL (exploitation des données)

```
1  SELECT nom
2  FROM etudiant
3  WHERE ville = 'Compiègne'
```

Cette instruction permet de rechercher les noms de tous les étudiants habitant la ville de Compiègne.

Complément : Autres langages de données

- XQuery est un langage de données mobilisé dans les bases de données arborescentes XML.
- Les bases NoSQL proposent des langages de données spécifiques, souvent inspirés du SQL. Par exemple le langage de MongoDB permet de manipuler une base de contenus JSON.

2. Approche générale pour la conception des bases de données

a) Exercice : Étapes de la conception d'une base de données relationnelle

Mettre dans l'ordre les étapes de conception suivantes.

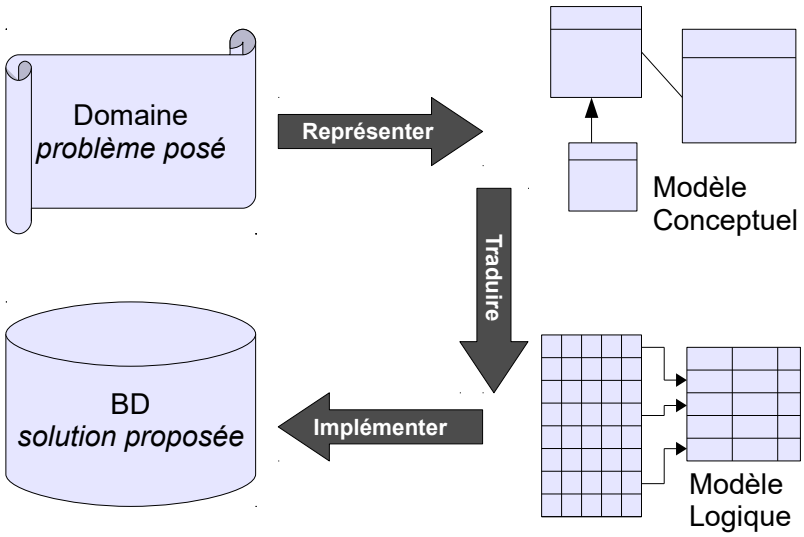
1. Modélisation conceptuelle en UML ou E-A
2. Création du code SQL pour un SGBDR
3. Analyse de la situation existante et des besoins
4. Élaboration du modèle logique en relationnel

Réponse : ___ ___ ___ ___

b) Méthodologie générale de conception d'une base de données

Méthode : Étapes de la conception d'une base de données

1. **Analyse** de la situation existante et des besoins (clarification)
2. Création d'un **modèle conceptuel** qui permet de représenter tous les aspects importants du problème
3. Traduction du modèle conceptuel en **modèle logique** (et normalisation de ce modèle logique)
4. Implémentation d'une **base de données** dans un SGBD, à partir du modèle logique (et optimisation)



Graphique 1 Processus de conception d'une base de données

On distingue quatre étapes dans la conception d'une base de données :

- **L'analyse**

  Elle consiste à étudier le problème et à consigner dans un document, la note de clarification, les besoins, les choix, les contraintes.
- **La modélisation conceptuelle**

  Elle permet de décrire le problème posé, de façon non-formelle (en générale graphique), en prenant des hypothèses de simplification. Ce n'est pas une description du réel, mais une représentation simplifiée d'une réalité.
- **La modélisation logique**

  Ele permet de décrire une solution, en prenant une orientation informatique générale (type de SGBD typiquement), formelle, mais indépendamment de choix d'implémentation spécifiques.
- **L'implémentation**

  Elle correspond aux choix techniques, en terme de SGBD choisi et à leur mise en œuvre (programmation, optimisation...).

Fondamental

- **Bien analyser** le problème posé en amont
- **Bien modéliser** le problème au niveau conceptuel avant de passer au niveau logique et à l'implémentation

Conseil : L'importance de l'étape d'analyse

La première étape de la conception repose sur l'analyse de l'existant et des besoins. De la qualité de la réalisation de cette première étape dépendra ensuite la pertinence de la base de données par rapports aux usages. Cette première étape est donc essentielle et doit être menée avec soins.

Si la première étape est fondamentale dans le processus de conception, elle est aussi la plus délicate. En effet, tandis que des formalismes puissants existent pour la modélisation conceptuelle puis pour la modélisation logique, la perception de l'existant et des besoins reste une étape qui repose essentiellement sur l'expertise d'analyse de l'ingénieur.

Conseil : L'importance de l'étape de modélisation conceptuelle

Étant donnée une analyse des besoins correctement réalisée, la seconde étape consiste à la traduire selon un modèle conceptuel. Le modèle conceptuel étant formel, il va permettre de passer d'une spécification en langage naturel, et donc soumise à interprétation, à une spécification non ambigüe. Le recours aux formalismes de modélisation tels que E-A ou UML est donc une aide fondamentale pour parvenir à une représentation qui ne sera plus liée à l'interprétation du lecteur.

La traduction d'un cahier des charges spécifiant l'existant et les besoins en modèle conceptuel reste néanmoins une étape délicate, qui va conditionner ensuite l'ensemble de l'implémentation informatique. En effet les étape suivantes sont plus mécaniques, dans la mesure où un modèle logique est déduit de façon systématique du modèle conceptuel et que l'implémentation logicielle est également réalisée par traduction directe du modèle logique.

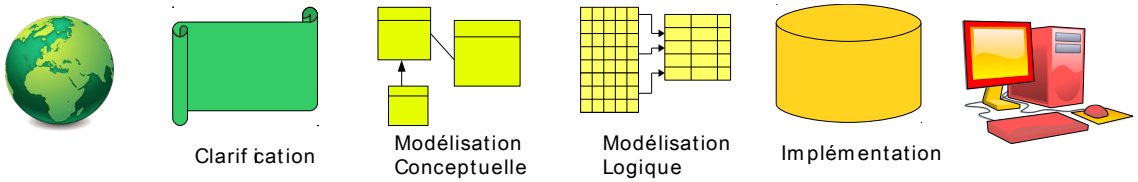
Remarque : Les étapes de traduction logique et d'implémentation

Des logiciels spécialisés sont capables à partir d'un modèle conceptuel d'appliquer des algorithmes de traduction qui permettent d'obtenir directement le modèle logique, puis les instructions pour la création de la base de données dans un langage orienté données tel que SQL. L'existence de tels algorithmes de traduction montre que les étapes de traduction logique et d'implémentation sont moins complexes que les précédentes, car plus systématiques.

Néanmoins ces étapes exigent tout de même des compétences techniques pour optimiser les modèles logiques (normalisation), puis les implémentations en fonction d'un contexte de mise en œuvre matériel, logiciel et humain.

c) Quatre étapes pour réduire la complexité

La réalisation d'une base de données est une tâche complexe, le découpage en quatre étapes permet de gérer cette complexité.



Conception en quatre étapes

L'idée générale est de ne pas mélanger la nature des tâches : typiquement, on ne veut pas en même temps se poser des questions générales relatives aux besoins (ce que je veux faire) et des questions techniques très spécifiques (comment représenter telle information).

Méthode

Conception en quatre étapes

L'approche est donc :

- de "regarder à gauche" : on regarde le monde quand on clarifie, on regarde la note de clarification quand on fait le MCD, on regarde le MCD quand on fait le MLD, on regarde le MLD quand on implémente ;
- et de ne "pas regarder trop loin" : on anticipe déjà le MCD pendant la clarification, mais on ne se préoccupe pas de questions d'implémentation ; quand on fait le MLD, on ne se pose plus de question sur le monde, la clarification et le MCD ont dû déjà répondre ; quand on implémente on ne se pose plus de question de modélisation, on s'occupe des programmes, de la machine.

Fondamental

On peut toujours revenir sur une étape, la conception est itérative, mais on ne doit pas tout le temps se poser tous les problèmes en même temps.

d) Éléments pour l'analyse de l'existant et des besoins

La phase d'analyse de l'existant et des besoins est une phase essentielle et complexe. Elle doit aboutir à des spécifications générales qui décrivent en langage naturel les données manipulées, et les traitements à effectuer sur ces données.

On se propose de donner une liste **non exhaustive** d'actions à mener pour rédiger de telles spécifications.

Méthode : L'analyse de documents existants

La conception d'une base de données s'inscrit généralement au sein d'usages existants. Ces usages sont généralement, au moins en partie, instrumentés à travers des documents électroniques ou non (papier typiquement). Il est fondamental d'analyser ces documents et de recenser les données qu'ils manipulent.

Exemple : Exemples de document existants à analyser

- Fichiers papiers de stockage des données (personnel, produits, etc.)
- Formulaires papiers d'enregistrement des données (fiche d'identification d'un salarié, fiche de description d'un produit, bon de commande, etc.)
- Documents électroniques de type traitement de texte (lettres, mailing, procédures, etc.)
- Documents électroniques de type tableurs (bilans, statistiques, calculs, etc.)
- Bases de données existantes, à remplacer ou avec lesquelles s'accorder (gestion des salaires, de la production, etc.)
- Intranet d'entreprise (information, téléchargement de documents, etc.)
- etc.

Méthode : Le recueil d'expertise métier

Les données que la base va devoir manipuler sont toujours relatives aux métiers de l'entreprise, et il existe des experts qui pratiquent ces métiers. Le dialogue avec ces experts est une source importante d'informations. Il permet également de fixer la terminologie du domaine.

Exemple : Exemples d'experts à consulter

- Praticiens (secrétaires, ouvrier, contrôleurs, etc.)
- Cadres (responsables de service, contre-maîtres, etc.)
- Experts externes (clients, fournisseurs, etc.)
- etc.

Méthode : Le dialogue avec les usagers

La base de données concerne des utilisateurs cibles, c'est à dire ceux qui produiront et consommeront effectivement les données de la base. Il est nécessaire de dialoguer avec ces utilisateurs, qui sont les détenteurs des connaissances relatives aux besoins réels, liés à leur réalité actuelle (aspects de l'organisation fonctionnant correctement ou défaillants) et à la réalité souhaitée (évolutions, lacunes, etc.).

Exemple : Exemples d'utilisateurs avec qui dialoguer

- Personnes qui vont effectuer les saisies d'information (à partir de quelles sources ? Quelle est leur responsabilité ? etc.)
- Personnes qui vont consulter les informations saisies (pour quel usage ? pour quel destinataire ? etc.)
- Personnes qui vont mettre à jour les informations (pour quelles raisons ? comment le processus est enclenché ? etc.)
- etc.

Méthode : L'étude des autres systèmes informatiques existants

la base de données va généralement (et en fait quasi systématiquement aujourd'hui) s'insérer parmi un ensemble d'autres logiciels informatiques travaillant sur les données de l'entreprise. Il est important d'analyser ces systèmes, afin de mieux comprendre les mécanismes existants, leurs forces et leurs lacunes, et de préparer l'intégration de la base avec ces autres systèmes. Une partie de ces systèmes seront d'ailleurs souvent également des utilisateurs de la base de données, tandis que la base de données sera elle même utilisatrice d'autre

systèmes.

Exemple : Exemples d'autres systèmes coexistants à étudier

- Autres bases de données (les données sont elle disjointes ou partiellement communes avec celles de la base à concevoir ? quelles sont les technologies logicielles sur lesquelles reposent ces BD ? etc.)
- Systèmes de fichiers classiques (certains fichiers ont-ils vocations à être supplantés par la base ? à être générés par la base ? à alimenter la base ? etc.)
- Applications (ces applications ont elles besoins de données de la base ? peuvent-elles lui en fournir ? etc.)
- etc.

Complément : Méthodes d'analyse

Il existe des outils et méthodes d'analyse en ingénierie, comme l'analyse fonctionnelle (AF), qui dépassent le cadre de la conception des bases de données, mais sont tout à fait complémentaires.

e) Note de clarification

Définition : Note de clarification (NDC)

La note de clarification est une reformulation du cahier des charges, qui précise, ajoute et supprime des éléments (en justifiant). C'est cette référence qui sera utilisée pour la suite du projet. Si le périmètre d'un projet est modifié, la NDC doit être ajouté en conséquence.

Méthode

Il existe plusieurs façons de réaliser une NDC. En base de données, on se focalisera sur l'explicitation des points suivants :

- Liste des objets qui devront être gérés dans la base de données
- Liste des propriétés associées à chaque objet
- Liste des contraintes associées à ces objets et propriétés
- Liste des utilisateurs (rôles) appelés à modifier et consulter les données
- Liste des fonctions que ces utilisateurs pourront effectuer

On formulera explicitement les hypothèses qui ont permis de faire des choix, lorsque le CDC

Attention

La NDC est la reformulation du cahier des charge sur laquelle on se base pour réaliser le MCD. Si aucune NCD n'est associée à un MCD, cela signifie que :

- soit le CDC est réputé sans erreur et parfaitement clair (ce qui est rare) ;
- soit on est pas capable de savoir à quoi se rapporte la modélisation.

Fondamental

On modélise toujours quelque chose dans un contexte, on pose que c'est ce qui est consigné dans la NDC. Si on ne sait pas ce qui est modélisé, la validité du modèle n'est pas évaluable.

Méthode

Un NDC peut revêtir plusieurs formats, mais on privilégiera :

- un document facile à mettre à jour (un projet évolue) et dont les versions sont traçables (on doit pouvoir remonter des modifications pour retrouver des bifurcations dans les attendus) ;
- un texte très clair, avec une rédaction minimaliste mais non ambiguë, des mots précis (par exemple on utilisera un seul mot pour un seul concept, la répétition dans ce contexte étant toujours préférable à un synonyme).

f) Modélisation conceptuelle de données

L'objection du modèle conceptuel est de représenter le problème à l'aide de représentations graphiques et partiellement formelles.

Les principales caractéristiques du modèle conceptuel sont :

- Une représentation graphique simple
- Une puissance d'expression élevée pour un nombre de symboles raisonnables
- Une lecture accessible à tous et donc un bon outil de dialogue entre les acteurs techniques et non techniques
- Une formalisation peu ambiguë et donc un bon outil de spécification détaillée

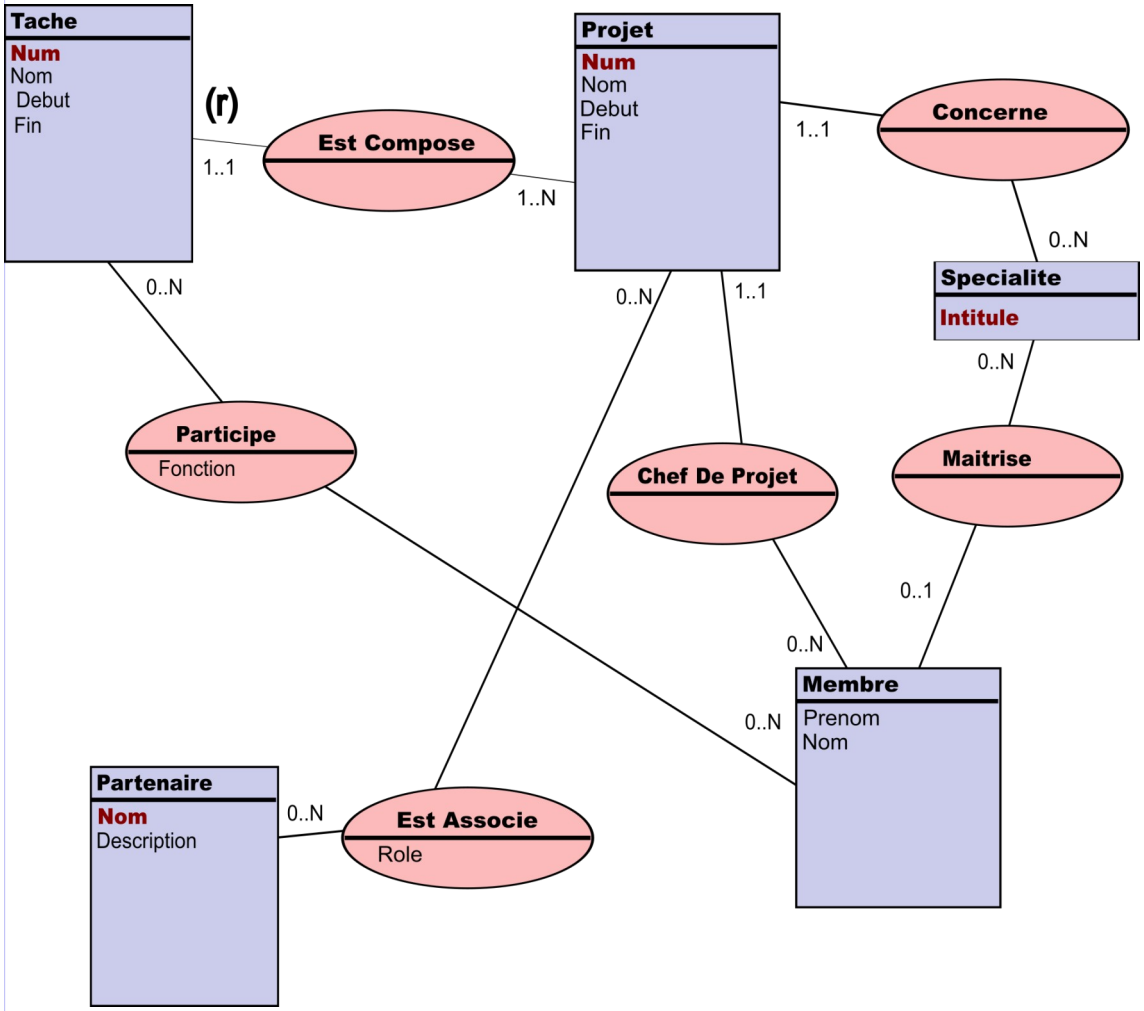
Remarque

Le modèle n'est pas encore formel, donc certaines représentations peuvent être équivoques, même si on a levé de très nombreuses ambiguïtés.

E-A

La modélisation conceptuelle en bases de données relationnelle était à l'origine faite avec le formalisme E-A de la méthode MERISE.

Exemple



Modèle E-A "gestion de projets"

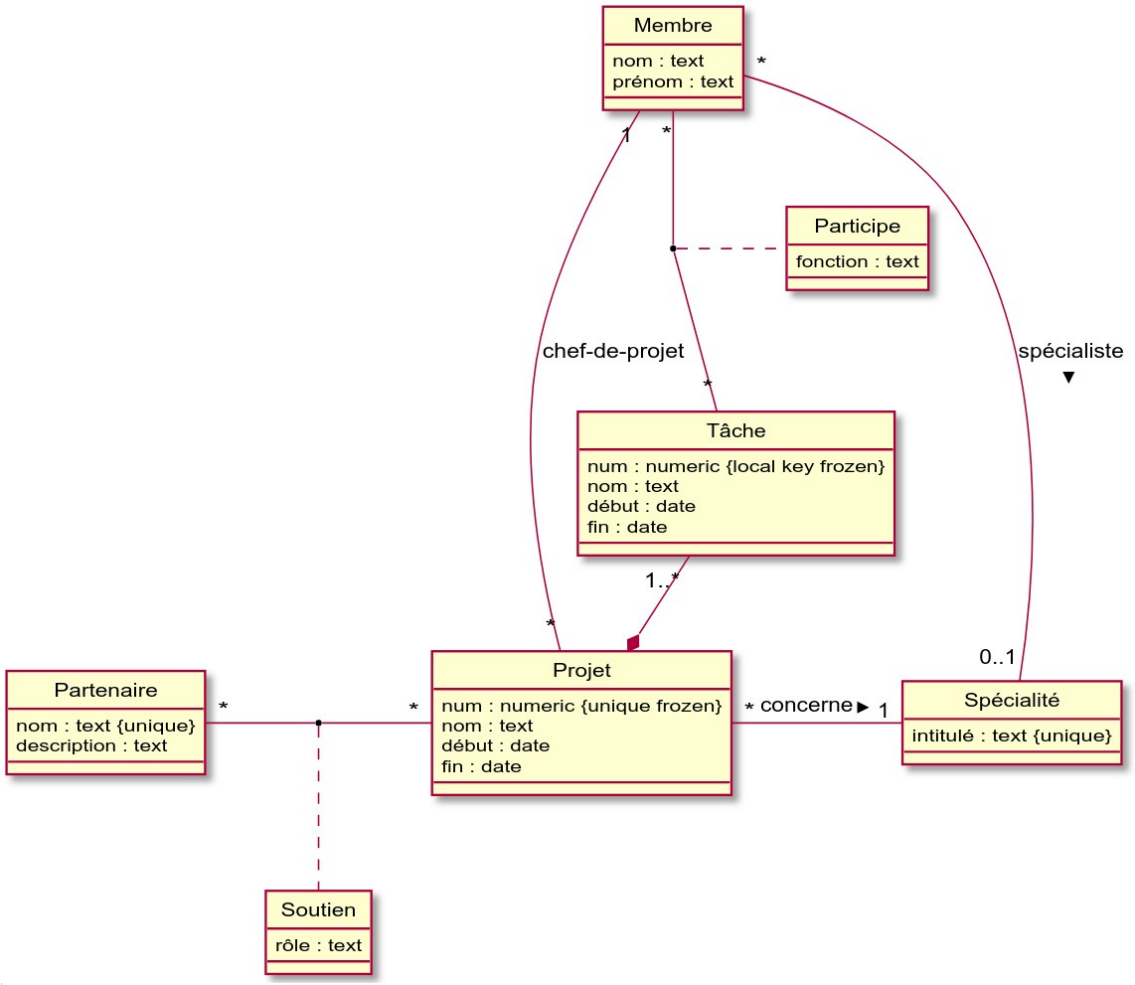
UML

UML est un autre langage de modélisation, plus récent que E-A et couvrant un spectre plus large que les bases de données. En tant que standard de l'OMG et en tant que outil très utilisé pour la programmation orientée objet, il a supplanté la modélisation E-A.

Remarque

En BD on utilise uniquement le **diagramme de classe** d'UML pour modéliser conceptuellement les données.

Exemple



Modèle UML "gestion de projets"

g) Modélisation logique de données

Définition : Modèle logique de données

Un modèle logique de données est une description, au moyen d'un langage formel, d'un ensemble de données.

Un schéma permet de décrire la structure d'une base de données, en décrivant l'ensemble des types de données de la base. Une instance de base de données est constituée d'un ensemble de données qui respectent le schéma de la base.

Synonyme : schéma de données, schéma

Exemple : Modèle logique de données relationnel

Un modèle logique de données relationnel permet de représenter une base de données relationnelles, c'est à dire : des tables, des propriétés, des domaines...

Exemple : Schéma d'une relation

```
1  Espece(nom:chaîne, eucaryote:booléen, multicellulaire:booléen,
   propriété:chaîne)
```

Exemple : Schéma d'une base de données avec plusieurs relations

```
1  Etudiant (num:entier, nom:chaîne, ville:chaîne)
2  Module(num:entier, titre:chaîne)
3  Inscription(numetu:entier, nummod:entier, année:entier(4))
```

Exemple : Instance de base de données

Etudiant

172	Dupont	Lille
173	Durand	Paris
174	Martin	Isabelle

Module

1	SGBD
2	OS

Inscription

172	1	2016
172	2	2016
173	1	2015
174	2	2017

Complément : Exemple de formalismes de modélisation logique

- Le modèle CODASYL, antérieur au modèle relationnel est un modèle hiérarchique (Tardieu, 1983 [Tardieu83]).
- Le modèle relationnel (tabulaire) est le modèle dominant à la base des SGBDR.
- Le modèle relationnel-objet (adaptation des modèles relationnels et objets au cadre des SGBD) est actuellement en croissance.
- D'autres modèles (document, graphe, ...) se développent dans le cadre du mouvement NoSQL.

Complément : Voir aussi

Définition du mouvement NoSQL

h) Synthèse : Les trois niveaux de conception

- Niveau Conceptuel

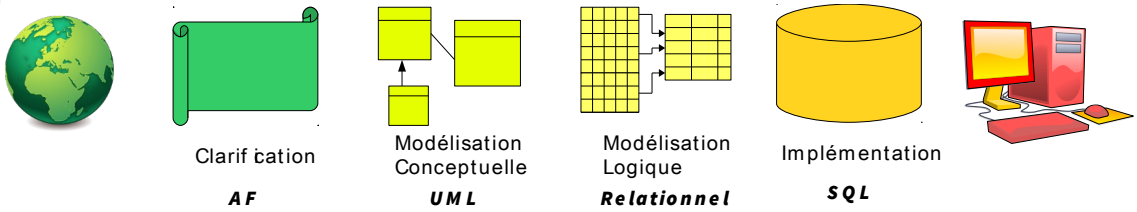
  Modèle conceptuel graphique
  - Exemples
    - E-A
    - UML
- Niveau Logique

  Schéma logique indépendant d'un SGBD
  - Exemples
    - Relationnel
    - Objet
    - Relationnel-Objet
    - Graphe
    - Document
- Niveau Informatique

  Implémentation pour un SGBD particulier

- Exemples
  - Oracle
  - MySQL
  - PostgreSQL
  - DB2
  - Access
  - SQLServer
  - MongoDB
  - Cassandra

Exemple



Conception en quatre étapes : exemple de formalismes

3. Découverte d'une base de données relationnelle

Objectifs

- Découvrir le modèle relationnel
- Découvrir un SGBDR
- Découvrir le langage SQL

Cette série d'exercices est destinée à faire expérimenter un SGBDR, afin de se familiariser avec les concepts classiques des bases de données relationnelles.

Pour la réalisation de cet exercice, se connecter sur le site Db Disco et conserver la fenêtre du navigateur ouverte.

a) Notion de table

Créer sa première table

Une base de données **relationnelle** est principalement constituée de **tables** (ou « relations » d'où le nom de relationnel). Une table est basiquement un élément d'organisation de l'information constitué de colonnes (ou attributs) et de lignes (ou enregistrements).

Nous allons dans un premier temps créer le **schéma** d'une table, c'est à dire définir ses colonnes. Pour cela nous utiliserons l'instruction SQL LDD « CREATE ».

Question 1

Exécuter l'instruction suivante et décrire ce qu'elle fait.

```
1  CREATE TABLE tEtu (
2  pk_numSecu CHAR(13) PRIMARY KEY,
3  k_numEtu VARCHAR(20) UNIQUE NOT NULL,
4  nom VARCHAR(50),
5  prenom VARCHAR(50));
```

Alimenter la table

Une fois les colonnes de la table définies, nous pouvons en déclarer les lignes. Nous utilisons pour cela l'instruction SQL LMD « INSERT ».

Question 2

Exécuter les deux instructions suivantes et décrire ce qu'elles font.

```
1  INSERT INTO tEtu (pk_numSecu, k_numEtu, nom, prenom)
2  VALUES ('1800675001066', 'AB3937098X', 'Dupont', 'Pierre');
3  INSERT INTO tEtu (pk_numSecu, k_numEtu, nom, prenom)
4  VALUES ('2820475001124', 'XGB67668', 'Durand', 'Anne');
```

Interroger la table

Une fois une table créée, il est possible à tout moment d'en inspecter le contenu. Nous utilisons pour cela l'instruction SQL LMD « SELECT ».

Question 3

Exécuter l'instruction suivante et décrire ce qu'elle fait.

```
1  SELECT pk_numSecu, k_numEtu, nom, prenom
2  FROM tEtu;
```

Question 4

Exécuter l'instruction suivante et décrire ce qu'elle fait.

```
1  SELECT nom, prenom
2  FROM tEtu
3  WHERE pk_numSecu='2820475001124';
```

b) Notion de contraintes

Contrainte de domaine

Lorsque l'on définit une table, on définit également des contraintes sur cette table, qui serviront à contrôler son **intégrité**, par rapport à des règles que l'on aura fixées.

C'est notamment le cas des contraintes de domaine, qui permettent de vérifier qu'une colonne prend ses valeurs parmi un ensemble déterminé (les chaînes de 10 caractères au plus, les entier de 1 à 1000, etc.).

Question 1

Exécuter l'instruction suivante et expliquer pourquoi le système renvoie une erreur.

```
1  INSERT INTO tEtu (pk_numSecu, k_numEtu, nom, prenom)
2  VALUES ('XXXXXXXXXXXXXXX', 'XXXXXX', 'Dupont', 'Pierre');
```

Question 2

Donner un exemple de contrainte qui n'est pas formulée dans la définition de la table *tEtu* et que l'on aurait pu souhaiter.

Indice :

> *Pour indiquer qu'un élément est obligatoire, on ajoute la clause `NOT NULL` après la définition de son domaine dans l'instruction `CREATE TABLE`.*

Contraintes de clé

Les contraintes de clé se composent de contraintes d'**unicité** et de contraintes de

non nullité. Elles permettent d'assurer que toutes les valeurs d'une colonne seront différentes pour chaque ligne.

Question 3

Exécuter les trois instructions suivantes (les unes après les autres) et expliquer ce qui se passe.

```
1  INSERT INTO tEtu (pk_numSecu, k_numEtu, nom, prenom)
2  VALUES ('1800675001066', 'HGYT67655Y', 'Dupont', 'Pierre');
3  INSERT INTO tEtu (pk_numSecu, k_numEtu, nom, prenom)
4  VALUES ('2810592012232', 'XGB67668', 'Durand', 'Anne');
5  INSERT INTO tEtu (pk_numSecu, k_numEtu, nom, prenom)
6  VALUES ('2810592012232', 'HGYT67655Y', 'Duchemin', 'Aline');
```

Question 4

Explorer le contenu de votre table en exécutant l'instruction suivante, et vérifier vos explications précédentes.

```
1  SELECT *
2  FROM tEtu;
```

Question 5

Pourrait-on insérer dans la table une seconde personne qui aurait le prénom "Aline" et le nom "Duchemin" ? Pourquoi ?

c) Notion de références

Clé étrangère

Une base de données est en général constituée de plusieurs tables. Ces tables se référencent entre elles en utilisant une **clé étrangère** : c'est à dire qu'une des colonnes de la table est utilisée pour faire référence à la colonne d'une autre table.

On va à présent créer une seconde table, qui permettra d'associer des Unités de Valeurs (UVs) aux étudiants, puis insérer des valeurs dans cette table.

```
1  CREATE TABLE tUv (
2  pk_code CHAR(4) NOT NULL,
3  fk_etu CHAR(13) NOT NULL,
4  PRIMARY KEY (pk_code, fk_etu),
5  FOREIGN KEY (fk_etu) REFERENCES tEtu(pk_numSecu));
```

```
1  INSERT INTO tUV (pk_code, fk_etu)
2  VALUES ('NF17', '1800675001066');
3  INSERT INTO tUV (pk_code, fk_etu)
4  VALUES ('NF26', '1800675001066');
5  INSERT INTO tUV (pk_code, fk_etu)
6  VALUES ('NF29', '1800675001066');
```

Question 1

Expliciter ce qu'exprime le contenu de la table tUv .

Contraintes d'intégrité référentielle

Lorsque nous avons défini la table tUv, nous avons défini une contrainte supplémentaire, dite d'intégrité référentielle : contrainte de type FOREIGN KEY.

Question 2

En exécutant les instructions suivantes, expliquer quel est le rôle d'une contrainte d'intégrité référentielle.

```
1  INSERT INTO tUV (pk_code, fk_etu)
2  VALUES ('NF17', '2810592012232');
3  INSERT INTO tUV (pk_code, fk_etu)
4  VALUES ('NF17', '1700792001278');
```

d) Projection, restriction et jointure

L'instruction SELECT du langage SQL LMD nous donne de larges possibilités pour interroger les tables d'une base de données. Cette instruction se fonde notamment sur les opérations mathématiques de l'algèbre relationnelle, dont les principales sont la projection, la restriction, le produit et la jointure.

Question 1

Exécuter l'instruction suivante et expliquer pourquoi c'est une **projection**.

```
1  SELECT nom, prenom
2  FROM tEtu;
```

Question 2

Exécuter l'instruction suivante et expliquer pourquoi c'est une **restriction**.

```
1  SELECT *
2  FROM tEtu
3  WHERE nom='Dupont';
```

Question 3

Exécuter l'instruction suivante et expliquer pourquoi c'est un **produit** (cartésien).

```
1  SELECT *
2  FROM tEtu,tUv;
```

Question 4

Exécuter l'instruction suivante et expliquer pourquoi c'est une **jointure**.

```
1  SELECT *
2  FROM tEtu JOIN tUv ON pk_numSecu=fk_etu;
```

Question 5

Exécuter l'instruction suivante et montrer qu'une jointure est la composition d'un produit et d'une restriction.

```
1  SELECT *
2  FROM tEtu,tUv
3  WHERE pk_numSecu=fk_etu;
```

e) Fonctions et agrégats

L'instruction SELECT permet également d'effectuer des calculs qui portent sur plusieurs lignes, ce que l'on appelle des agrégats.

Question 1

Exécuter la requête SQL suivante et expliquer le résultat obtenu.

```
1  SELECT COUNT(pk_code)
2  FROM tUv
3  WHERE fk_etu='1800675001066';
```

Question 2

Exécuter la requête SQL suivante et expliquer le résultat obtenu.

```
1  SELECT fk_etu, COUNT(pk_code)
2  FROM tUv
3  GROUP BY fk_etu;
```

Question 3

Compléter la requête SQL suivante afin qu'elle renvoie, pour chaque UV, le nombre d'étudiants inscrits.

```
1  SELECT _______, COUNT(______)
2  FROM tUv
3  GROUP BY _______
```

* *

*

À l'issue de cette série d'exercices, vous devez savoir définir les termes suivants :

- table ou relation
- schéma relationnel
- domaine
- clé
- clé étrangère
- opérations de projection, restriction, jointure, produit

B. Exercices

1. Lab 0

Description du problème

[30 min]

Un laboratoire souhaite gérer les médicaments qu'il conçoit.

Un médicament est décrit par un nom, qui permet de l'identifier. En effet il n'existe pas deux médicaments avec le même nom. Un médicament comporte une description courte en français, ainsi qu'une description longue en latin. On gère aussi le conditionnement du médicament, c'est à dire le nombre de pilules par boîte (qui est un nombre entier).

Ce problème est un exemple très simple que l'on pourra modéliser avec une base de données relationnelle à une seule table.

Vous pouvez tester vos instructions SQL ici : Db Disco

Question 1

Proposer une clarification du problème (par exemple sous la forme d'une liste des propriétés de la relation visée). On fera l'hypothèse qu'un seul type d'utilisateur existe et qu'il peut à la fois modifier et consulter toutes les données.

Question 2

Proposer un exemple de données.

Question 3

Dessiner un modèle conceptuel de données en UML. Il ne contient qu'une seule classe.

Indice :

Modélisation conceptuelle de données

Question 4

Proposer un modèle logique de données sous forme de schéma relationnel. Il ne contient qu'une seule relation.

Indice :

Modélisation logique de données

Question 5

Proposer une implémentation en SQL standard de votre modèle relationnel. Il ne contient qu'une seule instruction CREATE TABLE.

Utiliser Db Disco pour valider votre code.

Indice :

Langage de données : l'exemple du langage SQL

Question 6

Écrivez les instructions SQL permettant d'insérer vos données de test dans votre base de données.

Utiliser Db Disco pour valider votre code.

Indice :

Langage de données : l'exemple du langage SQL

2. Site de livres électroniques sous licence libre

[30 min]

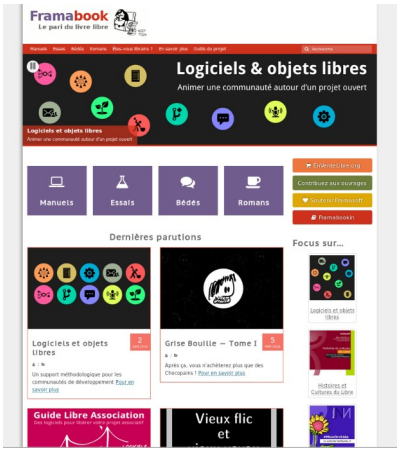
Vous avez comme projet de proposer un site de référencement de livres électroniques au format EPUB sous licence libre.

Question

Proposer, **librement**, une petite note de clarification d'environ une page (500 mots), qui identifiera :

- cinq ou six objets dotés de cinq ou six propriétés chacun
- deux ou trois rôles
- une dizaine de fonctions

Vous vous inspirerez en priorité du site similaire *Framabook*[1], dont la simplicité vous aidera à démarrer, mais vous pouvez chercher des éléments complémentaires dans votre propre expérience, ou bien sur d'autres sites similaires comme *inlibroveritas.net*[2], sur des sites commerciaux grands publics comme Amazon ou la Fnac, ou des sites plus spécialisés comme *C&F éditions*[3] ou *Epagine*[4].



1 - http://framabook.org/
2 - http://www.inlibroveritas.net/
3 - http://cfeditions.com
4 - http://www.epagine.fr/

La modélisation logique relationnelle

II

A. Cours

Le modèle relationnel est aux fondements des SGBDR. Il a été, et continue d'être, le modèle théorique dominant pour la représentation logique des base de données, même si le mouvement NoSQL propose des alternatives.

Le modèle relationnel permet de reformuler le modèle conceptuel dans un formalisme - le tableau - beaucoup plus proche de l'implémentation informatique, bien qu'encore indépendant d'une solution technologique particulière.

1. Introduction au modèle relationnel

Objectifs

Connaître les fondements théoriques du modèle relationnel.

a) Niveau logique

Le niveau logique est le lien entre le niveau conceptuel et l'implémentation effective de l'application. Le modèle conceptuel étant un modèle formel, le modèle logique a pour vocation d'être également un modèle formel, mais spécifiant non plus la réalité existante ou recherchée comme le modèle conceptuel, mais les données telles qu'elles vont exister dans l'application informatique.

Pour assumer cette fonction, le modèle relationnel s'est imposé en réaction aux insuffisances des modèles antérieurs, les modèles hiérarchique et réseau, et de part la puissance de ses fondements mathématiques.

Encore aujourd'hui dominant le modèle relationnel est un fondement indispensable

à la conception de bases de données.

Rappel

Méthodologie générale de conception d'une base de données

b) Définition du modèle relationnel

Introduction

Le modèle relationnel a été introduit par Codd [Codd70], en 1970 au laboratoire de recherche d'IBM de San José. Il s'agit d'un modèle simple et puissant à la base de la majorité des bases de données, encore aujourd'hui.

Les objectifs du modèle relationnel, formulés par Codd, sont les suivants :

- Assurer l'indépendance des applications et de la représentation interne des données
- Gérer les problèmes de cohérence et de redondance des données
- Utiliser des langages de données basés sur des théories solides

Définition : Modèle relationnel

On appelle modèle relationnel un ensemble de concepts permettant de formaliser logiquement la description d'articles de fichiers plats, indépendamment de la façon dont ils sont physiquement stockés dans une mémoire numérique.

Le modèle relationnel inclut des concepts pour la **description** de données, ainsi que des concepts pour la **manipulation** de données.

Fondamental : Représenter le monde en tables

Le modèle relationnel permet de représenter les données que l'on va gérer à l'aide d'un très petit nombre de concepts très simples :

- Les relations ou tables : des lignes et des colonnes
- Les domaines de valeurs : chaque case d'une table prend une unique valeur dans un domaine pré-défini
- Les clés : il existe des cases dont les valeurs doivent être uniques et non nulles
- Les clés étrangères : il existe des cases qui doivent prendre une valeur existante dans les cases d'une autre table

Complément : Extension du modèle relationnel

Le modèle relationnel est un standard, normalisé par l'ISO à travers son langage, le SQL. Il se veut néanmoins dès l'origine extensible, pour permettre de gérer des données plus complexes que les données tabulaires. Le modèle relationnel-objet est né de cette extension.

2. Les concepts fondamentaux du modèle relationnel : attributs, enregistrement, domaine

Objectifs

Connaître les fondements théoriques du modèle relationnel.

a) Domaine

Définition : Domaine

Ensemble, caractérisé par un nom, dans lequel des données peuvent prendre leurs valeurs.

Remarque

Un domaine peut-être défini en intension (c'est à dire en définissant les propriétés caractéristiques des valeurs du domaine, on parle aussi de compréhension) ou en extension (c'est à dire en énumérant toutes les valeurs du domaine)

Exemple : Domaines définis en intension

- Tous les entiers
- Les réels inférieur à 5
- Les booléen (vrai ou faux)
- Toutes les chaînes de 1 à 255 caractères
- Les valeurs monétaires, définie comme des décimaux avec deux chiffres après la virgule
- Les dates, définies comme des chaînes de 10 caractères comprenant des chiffres et des tirets selon le patron "00-00-0000"
- Les salaires, définis comme des valeurs monétaires compris entre 15.000 et 100.000

Exemple : Domaines définis en extension

- Couleur : {Bleu, Vert, Rouge, Jaune, Blanc, Noir}
- SGBD : {Hiérarchique, Réseau, Relationnel, Objet, Relationnel-Objet}

b) Exercice

Indiquez quelle définition et quel exemple correspondent respectivement aux mots ***intension*** *et* ***extension****.*

1 - Énonciation exhaustive de l'ensemble des objets du domaine

2 - Le domaine des couleurs

3 - {bleu, rouge, vert}

4 - Explicitation d'un domaine par la description de ses caractéristiques (en vue de sa compréhension abstraite, générale).

Intension Extension

c) Attribut et enregistrement

Définition : Attribut

On appelle attribut d'une relation, une colonne de cette relation. Un attribut est caractérisé par un nom et un domaine dans lequel il prend ses valeurs.

Synonymes : Champs, Propriété, Colonne

Définition : Enregistrement

On appelle enregistrement d'une relation, une ligne de cette relation. Un

enregistrement prend une valeur pour chaque attribut de la relation.
Synonymes : Tuple, N-uplet, Vecteur, Ligne

Exemple

A	B
1	1
1	2
2	2

Tableau 5 Relation R

La relation R comporte les deux attributs A et B et les trois enregistrements <1,1>, <1,2> et <2,2>

Remarque : Attribut, domaine, ordre

Un attribut se distingue d'un domaine car il peut ne comporter que certaines valeurs de ce domaine.

Les colonnes de la relation ne sont pas ordonnées et elles ne sont donc repérées que par le nom de l'attribut.

Remarque : Valeur nulle

Un enregistrement peut ne pas avoir de valeur pour certains attributs de la relation, parce que cette valeur est inconnue ou inapplicable, sa valeur est alors "null".

d) Exemple : La relation "Vol"

Exemple

Numero	Compagnie	Avion	Départ	Arrivée	Date
AF3245	Air France	747	Paris	Oulan Bator	01-08-2002
AF6767	Air France	A320	Paris	Toulouse	30-07-2002
KLM234	KML	727	Paris	Amsterdam	31-07-2002

Tableau 6 Relation Vol

3. Clés et clés étrangères dans le modèle relationnel :

Objectifs

Connaître les notions de clés candidates, naturelles, artificielles, primaire, étrangère

Aborder le principe d'éclatement des relations et de non-redondance.

a) Clé

Définition : Clé

Une clé est un groupe d'attributs minimum qui permet d'identifier de façon univoque un tuple dans une relation.

Fondamental

Toute relation doit comporter au moins une clé, ce qui implique qu'une relation ne peut pas contenir deux tuples identiques.

Attention : Attributs de clés unique et non null

Afin d'être déterminants pour l'identification d'un enregistrement, tous les attributs d'une clé doivent être valués, c'est-à-dire qu'aucun ne peut avoir de valeur *null*.

Dire qu'un groupe d'attributs est une clé implique qu'il est unique et non *null*.

Exemple : Numéro d'étudiant

- Le numéro d'étudiant d'une relation *Etudiant* est une bonne clé car il y aura systématiquement une valeur non nulle.
- Le groupe d'attributs (nom, prénom) d'une relation *Etudiant* est en général une mauvaise clé, car les homonymes existent.

b) Déterminer les clés

Détermination d'une clé

Définir un groupe d'attributs comme étant une clé nécessite une réflexion sémantique sur les données composant ces attributs, afin de s'assurer de leur unicité.

Fondamental

La définition des clés est un acte de **modélisation**, elle ne renvoie pas donc pas à une vérité intangible, mais à la réalité telle qu'elle est représentée dans le modèle que l'on élabore.

Exemple

L'attribut numéro de sécurité sociale d'une relation personne peut paraître une bonne clé a priori car son unicité est assurée. Mais tout le monde n'en dispose pas forcément (les enfants, des étrangers), donc ce n'est une clé que si l'on considère des personnes affiliées à la sécurité sociale.

c) Clé primaire et clés candidates

Définition : Clé primaire

Si plusieurs clés existent dans une relation, on en choisit une parmi celles-ci. Cette clé est appelée **clé primaire**.

La clé primaire est généralement choisie :

- de façon à être **immuable**, c'est à dire à ne jamais changer une fois valuée pour la première fois
- de façon à ce qu'elle soit la plus **simple**, c'est à dire portant sur le moins d'attributs et sur les attributs de domaine les plus basiques (entiers ou chaînes courtes typiquement).

Définition : Clés candidates

On appelle **clés candidates** l'ensemble des clés d'une relation qui n'ont pas été choisies comme clé primaire (elles étaient candidates à cette fonction).

d) Clé artificielle

Fondamental

S'il est impossible de trouver une clé primaire, ou que les clés candidates ne conviennent pas (non immuables, trop complexes...), il est possible de faire appel à une **clé artificielle**.

Définition : Clé artificielle

Une clé artificielle est un attribut supplémentaire ajouté au schéma de la relation, qui n'est lié à aucune signification, et qui sert uniquement à identifier de façon unique les enregistrements et/ou à simplifier les références de clés étrangères à l'intérieur de la base de données.

Définition : Clé signifiante

Une clé est signifiante si elle n'est pas artificielle.

Synonyme : Clé naturelle

Attention : Clé artificielle et niveau logique

Au niveau du modèle logique, il faut éviter la simplicité consistant à identifier toutes les relations avec des clés artificielles, et ne réserver cet usage qu'aux cas particuliers.

Conseil

1. Si au moins une clé naturelle immuable composée d'un seul attribut existe en choisir une parmi celles-ci comme clé primaire
2. Sinon, choisir une clé naturelle immuable composée de plusieurs attributs si elle ne pose pas de problème identifié
3. Toujours justifier l'emploi d'une clé artificielle (au niveau logique uniquement pour des raisons de complexité du modèle, les questions de performance sont étudiées au niveau physique)

Remarque : Clé artificielle et niveau physique, évolutivité, maintenance et performance

Au niveau de l'implémentation physique par contre, il est courant que des clés artificielles soient utilisées de façon systématique.

- Du point de vue de **l'évolutivité** de la BD, il existe toujours un risque qu'une clé non-artificielle perde sa propriété d'unicité ou de non-nullité.
- Du point de vue de **la maintenance** de la BD, il existe toujours un risque qu'une clé non-artificielle voit sa valeur modifiée (remise en cause de l'immutabilité) et dans ce cas, la répercussion de ce changement pour mettre à jour toutes les références peut poser problème.
- Du point de vue de **la performance** de la BD, les clés non-artificielles ne sont pas en général optimisées en terme de type et de taille, et donc peuvent limiter les performances dans le cadre des jointures. Précisons néanmoins qu'inversement les clés artificielles ont pour conséquence de systématiser des jointures qui auraient pu être évitées avec des clés primaires signifiantes.

Exemple : Problème d'évolutivité posé par une clé signifiante

Soit le numéro de sécurité sociale la clé primaire d'une table d'une BD française,

elle ne permettra pas d'entrer un individu non-français issu d'un pays ne disposant pas d'un tel numéro.

Exemple : Problème de maintenance posé par une clé signifiante

Soit le numéro de sécurité sociale la clé primaire d'une table d'une BD centrale dont les données sont exploitées par d'autres tables d'autres BD qui viennent "piocher" dans cette BD pour leurs propres usages, sans que la BD centrale ne connaisse ses "clients". Soit une erreur dans la saisie d'un numéro de sécurité sociale dans la BD centrale, si ce numéro est corrigé, il faudrait (ce qui n'est pas possible dans notre cas) impérativement en avertir toutes les bases utilisatrices pour qu'elles mettent à jour leurs références.

Exemple : Problème de performance posé par une clé signifiante

Soit le numéro de sécurité sociale la clé primaire d'une table comptant un million d'enregistrements, ce numéro est généralement un nombre à 13 chiffres ou une chaîne à 13 caractères, ce qui dans les deux cas est supérieur au nombre à 7 chiffres suffisant pour identifier tous les individus de la BD. Les performances seront donc toujours moins bonnes, lors des jointures, si une clé prend deux fois plus de place en mémoire que son optimum. Mais ajoutons que cette perte de performance n'a pas toujours de conséquence sur la réalité perceptible par les utilisateurs de la BD.

Inversement, soit une clé artificielle la clé primaire d'une table T1, par ailleurs référencée par une autre table T2. Soit le numéro de sécurité sociale un attribut clé de T1. Si l'on veut par requête disposer des informations de T2 ainsi que du numéro de sécurité sociale de T1, alors il faudra faire une jointure, tandis que si ce numéro signifiant avait été choisi comme clé primaire, cela n'aurait pas été nécessaire.

e) Clé étrangère

Définition : Clé étrangère

Une clé étrangère est un attribut ou un groupe d'attributs d'une relation R1 devant apparaître comme clé primaire dans une relation R2 afin de matérialiser une référence entre les tuples de R1 et les tuples de R2.

Une clé étrangère d'un tuple référence une clé primaire d'un autre tuple.

Attention

Seule une clé primaire peut être référencée par une clé étrangère.

C'est même la seule fonction de la clé primaire : être la clé qui peut être référencée par les clés étrangères.

Définition : Contrainte d'intégrité référentielle

Une clé étrangère respecte la contrainte d'intégrité référentielle si sa valeur est effectivement existante dans la clé primaire d'un tuple de la relation référencée, ou si sa valeur est *null*.

Une clé étrangère qui ne respecte pas la contrainte d'intégrité référentielle exprime un lien vers un tuple qui n'existe pas et donc n'est pas cohérente.

f) Référence entre relations

Le modèle relationnel a pour objectif la structuration de données selon des

relations. L'enjeu est de parvenir à traduire un modèle conceptuel en modèle logique relationnel. Or, il n'y a pas de notion d'association en relationnel, donc il faudra pouvoir traduire les associations avec les concepts dont on dispose : relation, clé, clé étrangère.

Afin de représenter des références entre relations dans un modèle relationnel, la seule solution est de stocker l'information dans une relation, et donc que certains attributs d'une relation servent à pointer sur d'autres relations.

Attention

Il n'y a pas vraiment de référence ou de lien en relationnel, puisque nous ne disposons que de tables, de clés, de clés étrangère et de valeurs.

On va donc devoir se servir de ces outils pour matérialiser une notion de référence.

Méthode : Référence

La référence entre deux tuples T1 et T2 de deux relations différentes est exprimable par une valeur identique entre une clé étrangère du tuple T1 et la clé primaire de l'autre tuple T2.

Synonyme : Lien

Exemple

R1

#a1	*a2=>R2*
A	2
B	1
C	1
D	4

R2

#b1	*b2*
1	F
2	F
3	F
4	V

Image 1

L'attribut *a2* de la relation *R1* référence l'attribut *b1* de la relation *R2* car *a2* est une clé étrangère de R1 vers R2 (*b1* est la clé primaire de *R2*).

Ici on a donc par exemple les tuples identifiés par *B* et *C* de *R1* qui référencent le tuple identifié par *1* dans *R2*.

g) Schéma relationnel

Définition : Schéma d'une relation

Le schéma d'une relation définit cette relation en intension. Il est composé :

- du nom de la relation,
- de la liste de ses attributs avec les domaines respectifs dans lesquels ils prennent leurs valeurs,
- de la clé primaire,
- des clés étrangères,
- des clés candidates.

Définition : Schéma relationnel d'une base de donnée

Le schéma relationnel d'une BD est la définition en intension de cette BD (par opposition à l'instance de la BD qui est une extension de la BD). Il est composé de l'ensemble des schémas de chaque relation de la BD.

Syntaxe : Relation

```
1 Relation (Attribut1:Domaine1, Attribut2:Domaine2, ... ,
  AttributN:DomaineN)
```

La relation "Relation" contient N attributs chacun défini sur son domaine.

Syntaxe : Clé primaire

```
1 Relation (#Attribut1:Domaine1, ... , #AttributM:DomaineM, ... ,
  AttributN:DomaineN)
```

La clé de la relation "Relation" est composée des attributs "Attribut1" à "AttributM" (attribut précédés de # ou bien soulignés)

En général on note la clé primaire en premier dans la relation.

Syntaxe : Clé étrangère

```
1 Relation1 (..., AttributM=>Relation2, ... , AttributN=>Relation2)
```

La relation "Relation1" comporte une clé étrangère (composée des attributs "AttributM" à "AttributN") référençant la clé primaire de "Relation2". Bien sûr il peut exister plusieurs clés étrangères vers plusieurs relations distinctes. Une clé étrangère et sa clé primaire référencée sont toujours composées du même nombre d'attributs. Il n'est pas nécessaire de préciser les domaines des attributs appartenant à la clé étrangère car ce sont forcément les mêmes que ceux de la clé primaire référencée. Il n'est pas non plus en général nécessaire de préciser dans le schéma relationnel quels attributs de la clé étrangère référencent quels attributs de la clé primaire (cela est généralement évident) mais il est possible de la faire on notant "Attribut=>Relation.Attribut".

En général on note les clés étrangères en dernier dans la relation, sauf pour les clés étrangères qui font partie de la clé primaire (clés identifiantes).

Syntaxe : Clé candidates

```
1 Relation1 (...AttributM:DomaineM, ......) avec AttributM clé
```

Les clés candidates doivent être notées sur le schéma relationnel :

- S'il n'y a qu'une ou deux clés candidates, les noter directement après la définition de la relation
- S'il y a beaucoup de clés, pour ne pas trop alourdir la notation, renvoyer à un tableau à part

Attention : Clés candidates et clé primaire

La notation `R(#a,#b)` signifie toujours que `R` a comme clé primaire `(a,b)`, et non que `R` aurait deux clés `a` et `b` (dont on ne saurait pas laquelle est primaire).

La notation `R(#a,b) avec b clé` signifie bien que `a` et `b` sont deux clés de `R`, et que `a` est primaire.

Il ne faut pas confondre **une clé composée de deux attributs** avec **deux clés**.

h) Exemple de schéma relationnel pour la géographie

Exemple

```
1  Personne (#Numero:Entier, Nom:Chaîne, Prénom:Chaîne,
   LieuNaissance=>Ville)
2  Pays (#Nom:Chaîne, Population:Entier, Superficie:Réel,
   Dirigeant=>Personne)
3  Région (#Pays=>Pays, #Nom:Chaîne, Superficie, Dirigeant=>Personne)
4  Ville (#CodePostal:CP, Nom:Chaîne, Pays=>Région.Pays,
   Région=>Région.Nom, Dirigeant=>Personne)
```

Exemple : Exemple d'instance de la base de données

Personne

#Numero	Nom	Prenom	LieuNaissance
1	Durand	Pierre	60200
2	Dupont	Marie	60200

Pays

#Nom	Population	Superficie	Dirigeant
France	60	500001,01	2
Allemagne	80	600000,23564	2
Espagne	40	350000,1	1

Région

#Pays	#Nom	Superficie	Dirigeant
France	Picardie	50	1
Espagne	Picardie	40	1
France	Normandie	30	2

Ville

#CodePostal	Nom	Pays	Région	Dirigeant
F60200	Compiègne	France	Picardie	1
F60300	Senlis	France	Picardie	2
F60301	Senlis	France	Picardie	2
E8000	Senlis	Espagne	Picardie	2

Le schéma relationnel précédent décrit :

- **Des personnes**

  Elles sont identifiées par un numéro qui est en fait une clé artificielle. En effet, même une clé composée de tous les attributs (Nom, Prénom, LieuNaissance) laisse une possibilité de doublons (homonymes nés dans la même ville).

  La clé étrangère LieuNaissance fait référence à la relation Ville, et plus précisément à sa clé primaire CodePostal, ce qui est est laissé implicite car évident.

- **Des pays**

  Ils sont identifiés par leur nom, puisque deux pays ne peuvent avoir le même nom.

  Les pays sont dirigés par des personnes, et ce lien est matérialisé par la clé étrangère Dirigeant.

- **Des régions**

  Elles font partie d'un pays et ont un nom. Deux régions de pays différents

pouvant avoir le même nom, il faut utiliser une clé primaire composée à la fois du nom de la région et du nom du pays, qui est une clé étrangère (le nom est appelé clé locale car il n'est pas suffisant pour identifier un tuple de la relation Région, et la clé étrangère vers la relation Pays est appelée clé identifiante).

- **Des villes**

  Elles sont identifié par un code postal qui est unique dans le monde (en utilisant le préfixe de pays de type "F-60200"). Ce code postal a pour domaine CP qui est une chaîne composée d'une ou deux lettres, d'un tiret, puis d'une série de chiffres.

  Le lien d'appartenance entre une ville et une région est matérialisé par la clé étrangère composée des deux attributs Pays et Région. Cette clé référence la clé primaire de la relation Région, également composée de deux attributs. Pour clairement expliciter les références (bien que sémantiquement la dénomination des attributs ne laisse pas de place au doute) on utilise la syntaxe Région.Pays et Région.Nom.

4. Synthèse

a) Synthèse : Schéma relationnel

Schéma relationnel

Un schéma relationnel permet une formalisation d'un modèle logique.

- Relation ou table

  Sous-ensemble d'un produit cartésien
  - Attribut ou colonne
    Prend ses valeurs dans un domaine
  - Enregistrement ou ligne
    Pose une valeur (y compris la valeur nulle) pour chaque attribut
- Clé

  Groupe d'attributs ayant un rôle d'identification au sein d'un enregistrement
  - Clé candidate
    Identifie de façon unique un enregistrement
  - Clé primaire
    Clé candidate choisie pour représenter un enregistrement pour sa facilité d'usage
  - Clé étrangère
    Référence la clé primaire d'un tuple d'une autre relation pour exprimer un lien

b) Bibliographie commentée sur le modèle relationnel

Complément : Synthèses

SQL2 SQL3, applications à Oracle [Delmal01]

Une définition synthétique et efficace du domaine relationnel : relation, domaine, attribut, clé, intégrité, opérateurs (Premier chapitre).

5. Définition formelle d'une relation

a) Produit cartésien

Définition : Produit cartésien

Le produit cartésien, noté "X", des domaines D1, D2, ... , Dn, noté "D1 X D2 X ... X Dn" est l'ensemble des tuples (ou n-uplets ou vecteurs) <V1,V2,...,Vn> tel que Vi est une valeur de Di et tel que toutes les combinaisons de valeurs possibles sont exprimées.

Exemple

```
1  D1 = {A, B, C}
2  D2 = {1, 2, 3}
3  D1 X D2 = {<A,1>, <A,2>, <A,3>, <B,1>, <B,2>, <B,3>, <C,1>, <C,2>,
   <C,3>,}
```

b) Relation

Définition : Relation

Une relation sur les domaines D1, D2, ... , Dn est un sous-ensemble du produit cartésien "D1 X D2 X ... X Dn". Une relation est caractérisée par un nom.

Synonymes : Table, tableau

Syntaxe

On peut représenter la relation R sur les domaine D1, ... , Dn par une table comportant une colonne pour chaque domaine et une ligne pour chaque tuple de la relation.

D1	...	Dn
V1	...	Vn
...	...	...
V1	...	Vn

Tableau 7 Relation R

Remarque

Une relation est définie en extension, par l'énumération des tuples la composant.

B. Exercices

1. Exercice

[10 min]

Question 1

Écrivez le résultat du produit cartésien R1 X R2

a	1
b	2

Tableau 8 R1

x
NULL

Tableau 9 R2

Question 2

Écrivez le résultat du produit cartésien R1 X R2

x	1

Tableau 10 R1

x
y

Tableau 11 R2

2. Lab I-

[20 min]

Un laboratoire souhaite gérer les médicaments qu'il conçoit.

- Un médicament est décrit par un nom, qui permet de l'identifier. En effet il n'existe pas deux médicaments avec le même nom. Un médicament comporte une description courte en français, ainsi qu'une description longue en latin. On gère aussi le conditionnement du médicament, c'est à dire le nombre de pilules par boîte (qui est un nombre entier).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte *« Médicament contre la chute des choux »* et pour description longue *« Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare. »*. Il est conditionné en boîte de 13.

  Ses contre-indications sont :
  - CI1 : Ne jamais prendre après minuit.
  - CI2 : Ne jamais mettre en contact avec de l'eau.
- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :
  - CI3 : Garder à l'abri de la lumière du soleil

Question 1

Dessiner des relations instanciées (en extension donc) remplies avec les données

fournies en exemple.

Question 2

Écrivez le schéma relationnel (définition en intension donc) permettant de représenter une base de données relationnelle pour le laboratoire.

III

Introduction à la modélisation conceptuelle de données avec UML

A. Cours

La **modélisation conceptuelle** est l'étape **fondatrice** du processus de conception de BD. Elle consiste à abstraire le problème réel posé pour en faire une reformulation qui trouvera une solution dans le cadre technologique d'un SGBD.

Si le modèle dominant en conception de bases de données a longtemps été le modèle E-A, le modèle UML se généralise de plus en plus. Nous proposons ici une introduction au diagramme de classes à travers la représentation de classes et d'associations simples (il existe d'autres diagrammes UML, par exemple le diagramme de cas, et d'autres primitives de représentation dans le diagramme de classe, par exemple l'héritage).

1. Notion de modèle

Objectifs

Savoir ce qu'est un modèle
Savoir ce qu'est le langage UML

a) Exercice : Centre médical

[10 min]

Soit le modèle conceptuel suivant représentant des visites dans un centre médical. Quelles sont les assertions vraies selon ce schéma ?

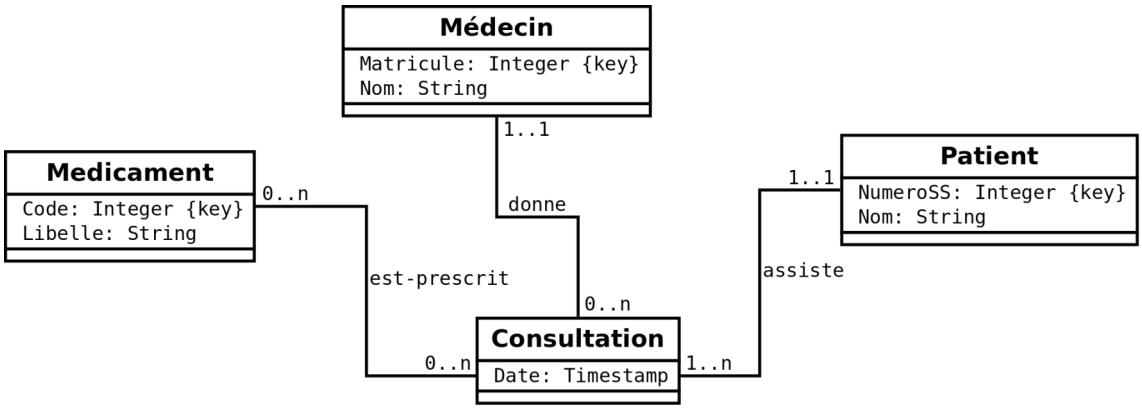


Image 2

☐	Un patient peut effectuer plusieurs visites.
☐	Tous les patients ont effectué au moins une consultation.
☐	Un médecin peut recevoir plusieurs patients pendant la même consultation.
☐	Un médecin peut prescrire plusieurs médicaments lors d'une même consultation.
☐	Deux médecins différents peuvent prescrire le même médicament.

b) Qu'est ce qu'un modèle ?

Définition : Modèle

« Modeling, in the broadest sense, is the cost-effective use of something in place of something else for some cognitive purpose. It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose. A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. » (Rothenberg, 1989 [Rothenberg et al., 1989], cité par Arribe, 2014 [Arribe, 2014])

« Système physique, mathématique ou logique représentant les structures essentielles d'une réalité et capable à son niveau d'en expliquer ou d'en reproduire dynamiquement le fonctionnement. » (TLFi)

Fondamental : Modèle

Un modèle est une représentation simplifiée de la réalité en vue de réaliser quelque chose.

c) Qu'est ce qu'un modèle en informatique ?

Définition : Modèle informatique

Un modèle informatique est une représentation simplifiée de la réalité en vue de réaliser un traitement avec un ordinateur.

Complément : Numérisation et abstraction : Toute information numérique a été codée selon un modèle donné

« Tout numérisation est une représentation de la réalité sous la forme d'une

modélisation numérique. Cette modélisation procède d'une abstraction au sens où c'est une séparation d'avec le réel, au sens où c'est une construction destinée à la manipulation (algorithmique en l'occurrence) et au sens où c'est une simplification de la réalité. »

http://aswemay.fr/co/tropism-pres.html[5]

d) Qu'est ce qu'un bon modèle ?

Attention

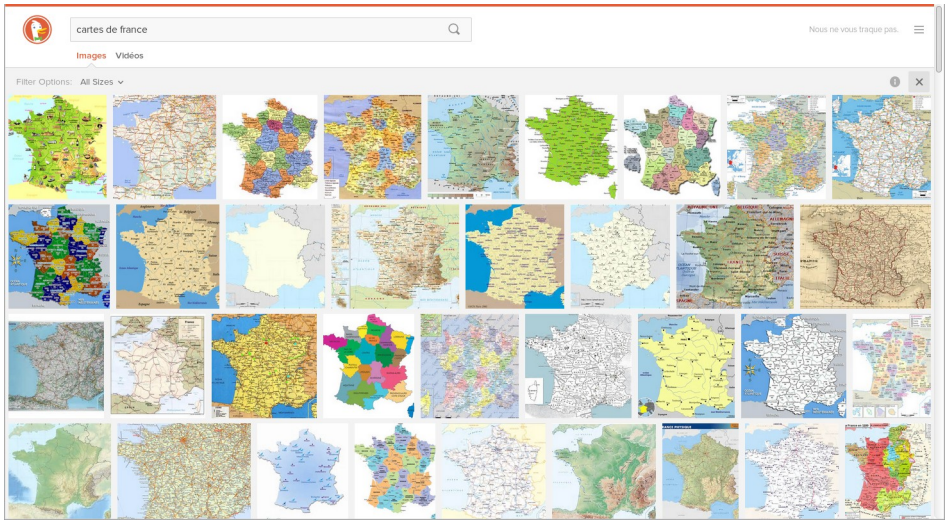
Un modèle est une abstraction, une simplification de la réalité, ce n'est pas la réalité, il n'est jamais complètement fidèle par construction.

Le seul modèle complètement fidèle à la réalité est la réalité elle-même, et ce n'est donc pas un modèle.

Exemple : La carte et le territoire

Une carte est un modèle d'un territoire. Elle est une représentation simplifiée destiné à un usage particulier :

- randonner à pied, en vélo ;
- se diriger en voiture sur des grands axes, sur des axes secondaires ;
- voler en avion de tourisme, en avion de ligne ;
- naviguer sur fleuve, sur mer ;
- étudier les frontières d'une région, d'un pays, de la terre ;
- étudier la démographie, l'économie... ;
- ...



Fondamental

À partir de cet exemple on notera que :

1. **Un modèle est orienté par un usage.**
   Chacune de ces cartes est très différente selon ce que l'on veut faire.
2. **Un modèle ne cherche pas à être proche de la réalité.**
   Chacune de ces cartes est très différente de la réalité qu'elle représente.
3. **Un modèle adresse un niveau d'information qui existe mais qui n'est pas accessible dans la réalité.**

5 - http://aswemay.fr/co/tropism-pres.html

Chacune de ces cartes permet quelque chose que ne permet pas l'accès direct à la réalité.

Méthode : Le rasoir d'Ockham : Entre deux modèles donnés le meilleur modèle est-il toujours le plus fourni ?

La méthode de raisonnement connue sous le nom de rasoir d'Ockham (du nom du philosophe éponyme) consiste à préférer les solutions les plus simples aux plus complexes, lorsqu'elles semblent permettre également de résoudre un problème donné ; entre deux théories équivalentes, toujours préférer la plus simple.

Ce principe s'applique très bien à la modélisation : étant donné un objectif et plusieurs modèles possibles, il ne faut pas choisir a priori celui qui représente le plus de choses, mais préférer le plus simple dès qu'il couvre le besoin.

C'est un principe d'économie (il coûte moins cher à produire) et d'efficacité (car les éléments inutiles du modèle plus fourni nuiront à l'efficacité de la tâche).

Exemple

Ainsi, pour naviguer en voiture, il est plus simple de ne **pas** avoir sur la carte les chemins de randonnées qui ne sont pas praticables en voiture.

2. Introduction au diagramme de classes UML : classes et associations

Objectifs

Savoir faire un modèle conceptuel.
Savoir interpréter un modèle conceptuel.

a) Lab I

Description du problème

[15 min]

Un laboratoire souhaite gérer les médicaments qu'il conçoit.

- Un médicament est décrit par un nom, qui permet de l'identifier. En effet il n'existe pas deux médicaments avec le même nom. Un médicament comporte une description courte en français, ainsi qu'une description longue en latin. On gère aussi le conditionnement du médicament, c'est à dire le nombre de pilules par boîte (qui est un nombre entier).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte *« Médicament contre la chute des choux »* et pour description longue *« Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare. »*. Il est conditionné en boîte de 13.

Ses contre-indications sont :
- CI1 : Ne jamais prendre après minuit.
- CI2 : Ne jamais mettre en contact avec de l'eau.

- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :
  - CI3 : Garder à l'abri de la lumière du soleil

Question 1

Réaliser le modèle conceptuel de données en UML du problème.

Question 2

Étendre le modèle conceptuel UML afin d'ajouter la gestion des composants. Un composant est identifié par un code unique et possède un intitulé. Tout médicament possède au moins un composant, souvent plusieurs. Tout composant peut intervenir dans la fabrication de plusieurs médicaments. Il existe des composants qui ne sont pas utilisés pour fabriquer des médicaments et que l'on veut quand même gérer.

b) Présentation d'UML

Contexte

UML est un langage de représentation destiné en particulier à la modélisation objet. UML est devenu une norme OMG en 1997.

UML propose un formalisme qui impose de "penser objet" et permet de rester indépendant d'un langage de programmation donné lors des premières phases de conception.

UML permet :
- d'énumérer et de décrire les concepts
- de les noter graphiquement.

Il est utilisé comme un moyen de communication entre les étapes de spécification conceptuelle et les étapes de spécifications techniques.

Fondamental : Diagramme de classe

Le diagramme de classes est un sous ensemble d'UML qui s'attache à la description statique d'un modèle de données représentées par des classes d'objets.

Remarque

Dans le domaine des bases de données, UML peut être utilisé à la place du modèle E-A pour modéliser le domaine. De la même façon, un schéma conceptuel UML peut alors être traduit en schéma logique (relationnel ou relationnel-objet typiquement).

c) Classes

Définition : Classe

Une classe est un type abstrait caractérisé par des propriétés (attributs et méthodes) communes à un ensemble d'objets et permettant de créer des instances de ces objets, ayant ces propriétés.

Syntaxe

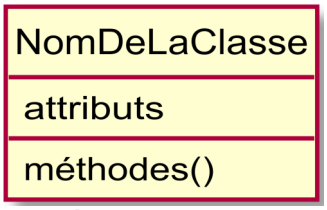


Image 3 Représentation UML d'une classe

Exemple

Image 4 La classe Voiture

Exemple

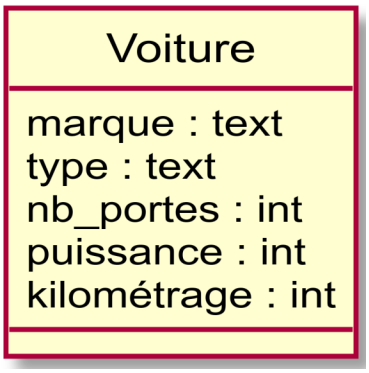


Image 5 La classe Voiture (avec attributs)

Exemple : Une instance de la classe Voiture

L'objet V1 est une instance de la classe Voiture.

V1 : Voiture

- Marque : 'Citroën'
- Type : 'Visa'
- Portes : 5
- Puissance : 60
- Kilométrage : 300000

Complément

La modélisation sous forme de diagramme de classes est une modélisation statique, qui met en exergue la structure d'un modèle, mais ne rend pas compte de son évolution temporelle. UML propose d'autres types de diagrammes pour traiter, notamment, de ces aspects.

d) Attributs

Définition : Attribut

Un attribut est une information élémentaire qui caractérise une classe et dont la valeur dépend de l'objet instancié.

Un attribut est typé : Le domaine des valeurs que peut prendre l'attribut est fixé a priori.

Exemple

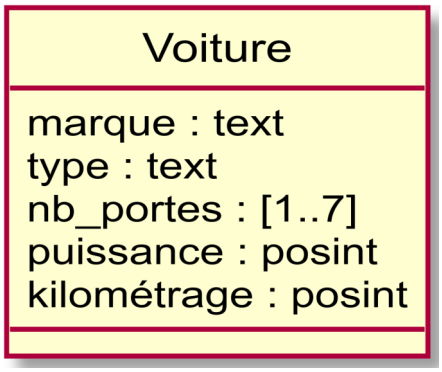


Image 6 La classe Voiture (avec attributs et types précis)

Syntaxe : Types

Le typage des attributs peut se faire dans une syntaxe libre, il n'est exprimé que pour :

- consigner des contraintes,
- expliciter la nature de l'attribut.

On peut utiliser une terminologie française, anglais, inventée, liée à un langage, etc. On veillera néanmoins à conserver une terminologie cohérente.

Conseil : Précision des types

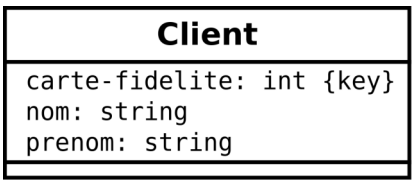
On conseille d'adopter sur le schéma un typage précis qui rend compte de ce qui est exprimé dans l'analyse des besoins. Si l'on sait qu'une voiture ne peut pas avoir plus de sept portes, on le consigne dès l'UML dans le typage.

e) Repérage des clés

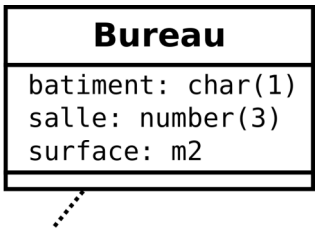
Un attribut ou un groupe d'attributs peut être annoté comme étant **clé** s'il permet d'identifier de façon unique un objet de la classe.

On ajoute le symbole `{unique}` à côté du ou des attributs concernés.

Exemple



Clé en UML



Clé composée de deux attributs

Méthode

Le repérage des clés n'est pas systématique en UML (la définition des clés se fera essentiellement au niveau logique). On cherchera néanmoins à repérer les clés rendues évidentes par la phase de clarification.

Une classe peut très bien ne pas avoir de clé repérée en UML.

Attention

On n'ajoutera jamais de clé artificielle au niveau du MCD. Si aucune clé n'est évidente, on laisse la classe sans clé.

Remarque : Notation {unique}

Les notation `{key}` et `{unique}` sont équivalente en UML}

Remarque : Attribut souligné et #

On trouvera dans ce cours des exemples d'attributs soulignés ou précédés de # pour exprimer l'unicité. Ce n'est pas une pratique standard et la notation `{key}` devrait lui être substituée.

- Un attribut souligné est normalement un attribut de classe, ou *static*, en UML,
- Un attribut précédé de # est normalement un attribut protégé en UML.

Mais les concepts d'attribut de classe et d'attribut protégé ne sont pas utilisés dans le cadre des bases de données.

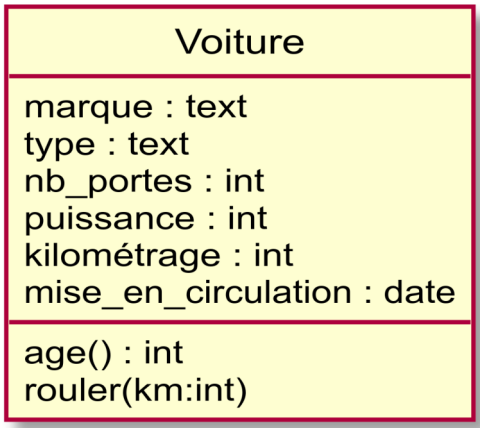
f) Méthodes

Définition : Méthode

Une méthode (ou opération) est une fonction associée à une classe d'objet qui permet d'agir sur les objets de la classe ou qui permet à ces objets de renvoyer des valeurs (calculées en fonction de paramètres).

Syntaxe

Les méthodes se notent sous les attributs.



La classe voiture (avec méthodes)

Méthode : Méthodes et modélisation de BD

Pour la modélisation des bases de données, les méthodes sont surtout utilisées pour :

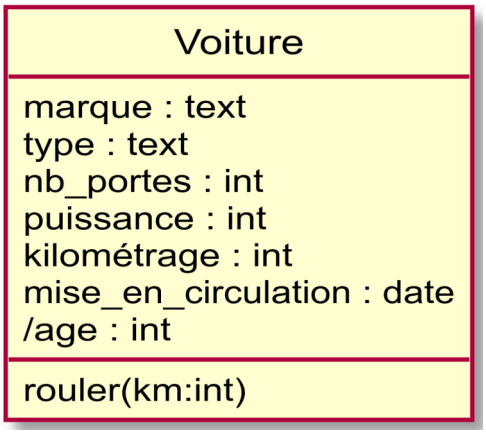
- représenter des données calculées : exemple de l'age calculé à partir de la date de mise en circulation ;
- ou pour mettre en exergue des fonctions importantes du système cible : exemple de la fonction rouler qui insiste sur la fonction de mise à jour du kilométrage (qui devra être particulièrement considérée dans l'application cible).

Seules les méthodes les plus importantes sont représentées (l'approche est moins systématique qu'en modélisation objet par exemple).

Complément : Attributs dérivés

La notation d'attributs dérivés (importée du modèle E-A en UML) consiste à faire précédé d'un / un nom d'attribut pour montrer que cet attribut est calculé à partir d'autres attributs de la même classe.

C'est donc une alternative aux méthodes dans ce cas. Mais en UML on préfère l'usage de méthodes aux attributs dérivés.



La classe voiture (avec attribut dérivé)

Complément

Transformation des méthodes par des vues

g) Associations

Définition : Association

Une association est une relation logique entre deux classes (association binaire) ou plus (association n-aire) qui définit un ensemble de liens entre les objets de ces classes.

Une association est **nommée**, généralement par un verbe. Une association peut avoir des propriétés (à l'instar d'une classe). Une association définit le nombre minimum et maximum d'instances autorisée dans la relation (on parle de cardinalité).

Syntaxe

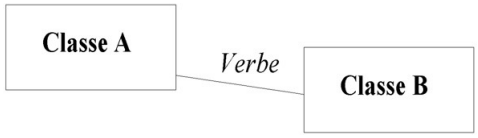


Image 7 Notation de l'association en UML

Attention

Le nom de l'association (verbe qui la décrit) est obligatoire, au même titre que le nom d'une classe ou d'un attribut.

Remarque

Une association est généralement bidirectionnelle (c'est à dire qu'elle peut se lire dans les deux sens). Les associations qui ne respectent pas cette propriété sont dites unidirectionnelles ou à navigation restreinte.

Exemple : L'association Conduit

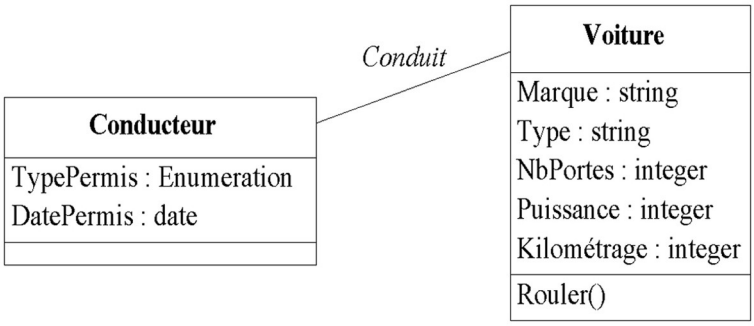


Image 8 Représentation d'association en UML

L'association Conduit entre les classes Conducteur et Voiture exprime que les conducteurs conduisent des voitures.

Complément : Voir aussi

- *Cardinalité*
- *Explicitation des associations*
- *Associations ternaires*
- *Contraintes standardisées sur les associations*

h) Cardinalité

Définition : Cardinalité d'une association

La cardinalité d'une association permet de représenter le nombre minimum et maximum d'instances qui sont autorisées à participer à la relation. La cardinalité est définie pour les deux sens de la relation.

Syntaxe

Si min_a (resp. max_a) est le nombre minimum (resp. maximum) d'instances de la classe A autorisées à participer à l'association, on note sur la relation, à côté de la classe A : $min_a..max_a$.

Si le nombre maximum est indéterminé, on note n ou *.

Attention

La notation de la cardinalité en UML est opposée à celle adoptée en E-A. En UML on note à gauche (resp. à droite) le nombre d'instances de la classe de gauche (resp. de droite) autorisées dans l'association. En E-A, on note à gauche (resp. à droite) le nombre d'instances de la classe de droite (resp. de gauche) autorisées dans l'association.

Remarque

Les cardinalités les plus courantes sont :

- 0..1 (optionnel)
- 1..1 ou 1 (un)
- 0..n ou 0..* ou * (plusieurs)
- 1..n ou 1..* (obligatoire)

Exemple : La cardinalité de l'association Possède

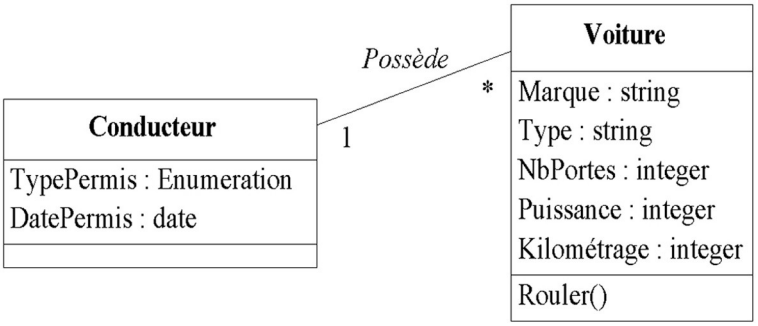


Image 9 Représentation de cardinalité en UML

Ici un conducteur peut posséder plusieurs voitures (y compris aucune) et une voiture n'est possédée que par un seul conducteur.

Fondamental : Terminologie

- On appelle association 1:1 les associations de type :
  - 0..1:0..1
  - 0..1:1..1
  - 1..1:0..1
  - 1..1:1..1

- On appelle association 1:N les associations de type :
  - 0..1:0..N
  - 0..1:1..N
  - 1..1:0..N
  - 1..1:1..N
- On appelle association N:M (ou M:N) les associations de type :
  - 0..N:0..N
  - 0..N:1..N
  - 1..N:0..N
  - 1..N:1..N

i) Classe d'association

Définition : Classe d'association

On utilise la notation des classes d'association lorsque l'on souhaite ajouter des propriétés à une association.

Syntaxe : Notation d'une classe d'association en UML

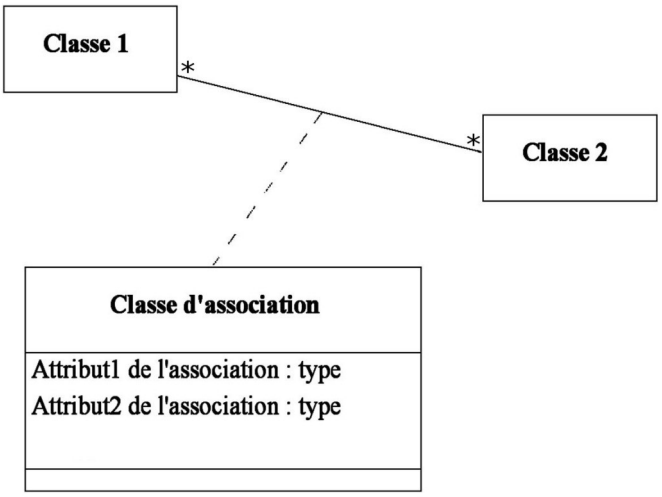


Image 10 Notation d'une classe d'association en UML

Méthode

On réserve en général les classes d'association aux associations N:M.

Méthode

Il est toujours possible de réduire une classe d'association sur une association 1:N en migrant ses attributs sur la classe côté N, et c'est en général plus lisible ainsi.

Exemple : Exemple de classe d'association

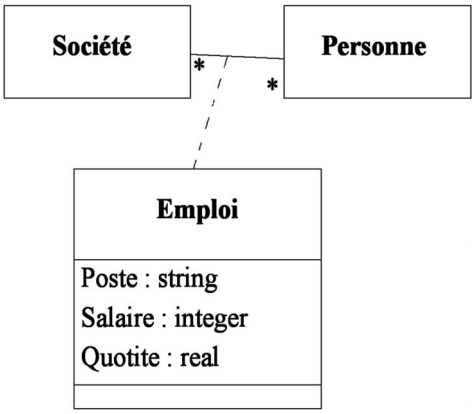


Image 11 Emplois

Conseil

Selon le standard UML une classe d'association est une classe et à ce titre elle peut être mobilisée dans d'autres associations ou dans des héritages. Nous déconseillons néanmoins ces notations qui ont tendance à complexifier la lecture et la transformation du diagramme.

Nous conseillons donc de ne jamais associer une classe d'association.

3. Quels outils pour faire de l'UML ?

a) Outils pour l'élaboration (papier/crayon)

Fondamental

L'élaboration d'un schéma UML est d'abord un outil permettant de chercher une solution, il est donc **indispensable** qu'il soit facile de le modifier de changer d'avis, et que sa manipulation ne soit pas parasitée par de la manipulation informatique.

Conseil : Papier/crayon

Pour élaborer un schéma UML la meilleure méthode est de se doter :

- d'une grande feuille (au minimum un A3 pour un petit projet de quelques classes) ;
- d'un crayon de papier et d'une gomme

On peut aussi travailler au tableau (noir, blanc, numérique...).

Attention

Un diagramme UML, même pour l'élaboration doit rester parfaitement visible, c'est un outil **graphique** et c'est un outil de **dialogue** s'il est mal lisible on fera des erreurs et il remplira mal son rôle.

Conseil : Éditeur graphique

Si l'on est doté d'une bonne expérience sur un outil **graphique** alors il est possible de l'utiliser à la place du papier crayon, mais :

- il faut bien connaître l'outil pour que son attention soit portée sur la

conception et non sur le fonctionnement de l'outil ;

- il faut que l'outil soit totalement souple pour permettre une expression graphique non contrainte (typiquement certains éditeurs contraignent la création afin de rester en mesure de faire de la génération automatique de code, ce qui est hors-sujet dans notre cas).

b) Dia

Dia est un petit outil graphique libre et multi-plate-forme.

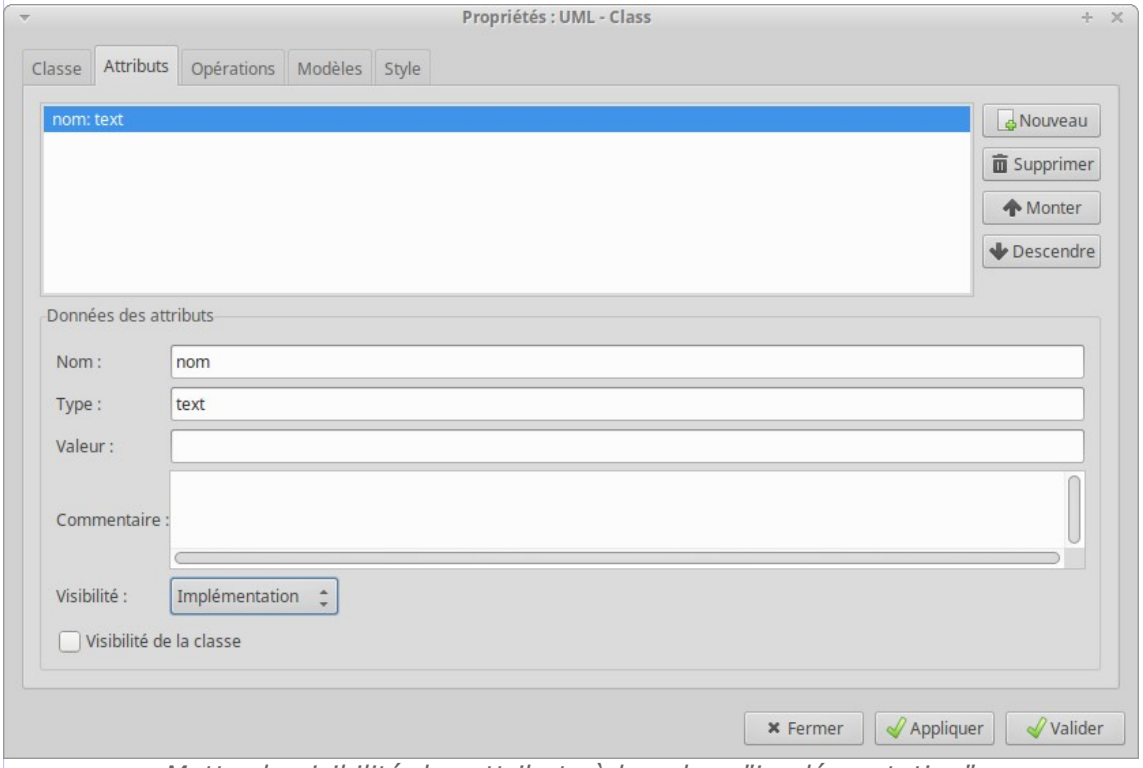
- *https://fr.wikipedia.org/wiki/Dia_(logiciel)*[6]
- *http://dia-installer.de/*[7]

Il présente pas mal de défauts, mais il est assez facile à prendre en main et il est possible d'utiliser le mode graphique pour dessiner ce qui n'est pas prévu dans le module UML.

Méthode : Ne pas afficher la visibilité des attributs

Dia fixe par défaut la visibilité de tous les attributs à la valeur "Public", ce qui a comme conséquence de faire précéder l'attribut d'un caractère "+".

C'est un concept qu'on utilise pas en base de données relationnelle, cela surcharge donc inutilement tous les diagrammes. Il faut donc systématiquement modifier cette propriété.



Mettre la visibilité des attributs à la valeur "implémentation"

Méthode : Export en PNG

Utiliser la fonction `Fichier > Export` pour transformer votre diagramme en fichier PNG.

Choisissez une résolution assez élevée si vous visez une impression papier.

6 - https://fr.wikipedia.org/wiki/Dia_(logiciel)
7 - http://dia-installer.de/

c) Outils pour la documentation (éditeur)

Pour la documentation de la conception d'une base de données, on est souvent amené à livrer le MCD en UML.

Conseil : Diagramme papier/crayon propre

On peut tout à fait livrer un UML fait à la main sur papier.

On veillera néanmoins à ce que le diagramme soit très propre et très clair et à ce que la numérisation soit de qualité (on préférera l'usage d'un scanner à celui d'un ordiphone, et si on prend une photo ce sera une photo de qualité, plane et parfaitement recadrée).

Attention

Le diagramme UML livré en documentation ne doit pas ressembler à un brouillon, mais bien à un document de référence.

Conseil : PlantUML

On conseille pour la documentation l'usage du logiciel PlantUML.

Combiné avec un dépôt Git il permet d'historiser la dynamique de conception.

d) PlantUML

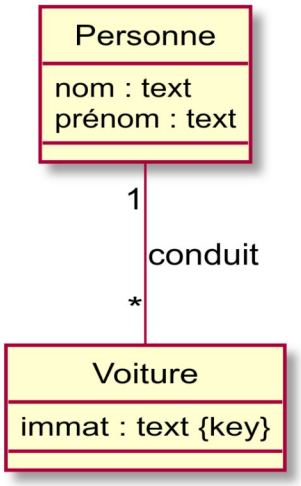
PlantUML est un outil libre et multi-plate-forme qui permet de créer des diagrammes UML à partir d'un langage textuel de description.

- *http://plantuml.com/fr/class-diagram*[8] (PDF (cf. PlantUML_Language_Reference_Guide_fr_diagramme_de_classe))
- *https://en.wikipedia.org/wiki/PlantUML*[9]

Exemple

```
1  @startuml
2  hide circle
3
4  class Voiture {
5    immat : text {key}
6  }
7
8  class Personne {
9    nom : text
10   prénom : text
11 }
12
13 Personne "1" -- "*" Voiture  : conduit
14
15 @enduml
```

8 - http://plantuml.com/fr/class-diagram
9 - https://en.wikipedia.org/wiki/PlantUML



Méthode : hide circle

La commande `hide circle` permet de supprimer des éléments de syntaxe propre à certaines représentations objet, non utilisés en base de données.

Méthode : Notation des clés composées

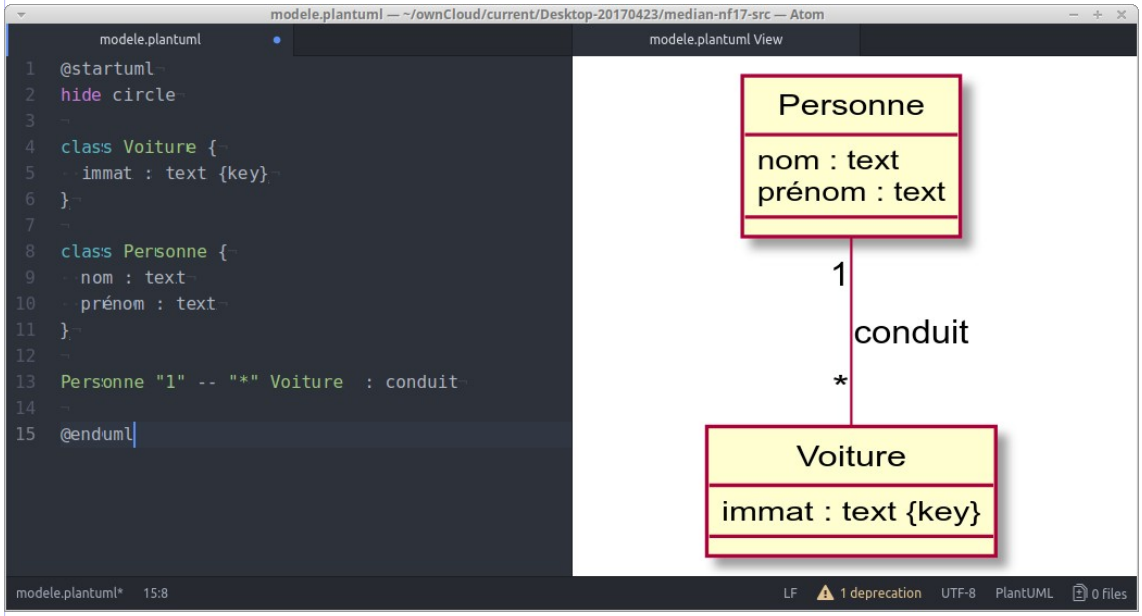
```
1  class Employé {
2    nom : text
3    prénom : text
4    ddn : date
5    date_embauche : date
6    quotite : pourcentage
7  }
8  note left : (nom, prénom, ddn) unique
```

Atom et PlantUML

PlantUML peut-être intégré à Atom avec les plugin `plantuml-viewer` et `language-plantuml`.

```
1  apm install plantuml-viewer language-plantuml
```

Exemple : PlantUML dans Atom



Remarque : Positionnement vertical / horizontal

Notez la différence entre :

- `-` : à côté (un seul tiret)
- `--` : en dessous (deux tirets ou plus)

L'ordre de déclaration des classes est pris en compte.

Complément : Ajustement de la longueur des associations avec PlantUML

- Horizontal : `Classe1 - Classe2 : "  Association  "` (on ajoute des espaces)
- Vertical : `Classe1 ------ Classe2 : "Association"` (on ajoute des tirets)

Complément : Ajustement du positionnement avec PlantUML

L'usage d'association `[hidden]` permet d'améliorer un peu la présentation d'un diagramme PlantUML (à utiliser avec parcimonie).

La bonne solution est plutôt l'usage de **package**.

```
@startuml

class Alice
class Bob
class Charlie
class Dan
class Eugene
class Foobar

Bob -[hidden] Alice : ré-ordonnancement
Alice -[hidden] Charlie : éloignement
 horizontal
Dan --[hidden] Eugene : positionnement vertical
Alice --------[hidden] Foobar : éloignement vertical
```

```
14
15 @enduml
```

Complément : Contournements pour exprimer des contraintes d'association avec PlantUML

```
1 Alice -- Bob : Assoc1
2 note right on link
3 {XOR}
4 end note
5 Alice -- Bob : Assoc2
```

```
1 Alice -- Bob : Assoc1....{XOR}....
2 Alice -- Bob : Assoc2
```

```
1 Alice -- Bob : Assoc1
2 Alice -- Bob : Assoc2
3 note "{XOR}" as N #white
4 (Alice,Bob) .. N
5 N .. (Alice,Bob)
```

Complément : Commentaires

Toute ligne qui commence par un ' est un commentaire, non interprété par le moteur.

B. Exercices

1. Exercice : Lire l'UML

[30 min]

Tennis

Le schéma suivant représente les rencontres lors d'un tournoi de tennis. Quelles sont les assertions vraies selon ce schéma ?

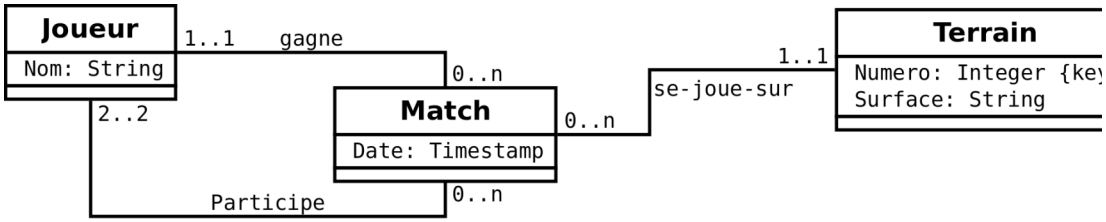


Image 12

- ☐ On peut jouer des matchs de double.
- ☐ Un joueur peut gagner un match sans y avoir participé.
- ☐ Il peut y avoir deux matchs sur le même terrain à la même heure.
- ☐ Connaissant un joueur, on peut savoir sur quels terrains il a joué.

Journal

Voici le schéma conceptuel du système d'information (très simplifié) d'un quotidien. Quelles sont les assertions vraies selon ce schéma ?

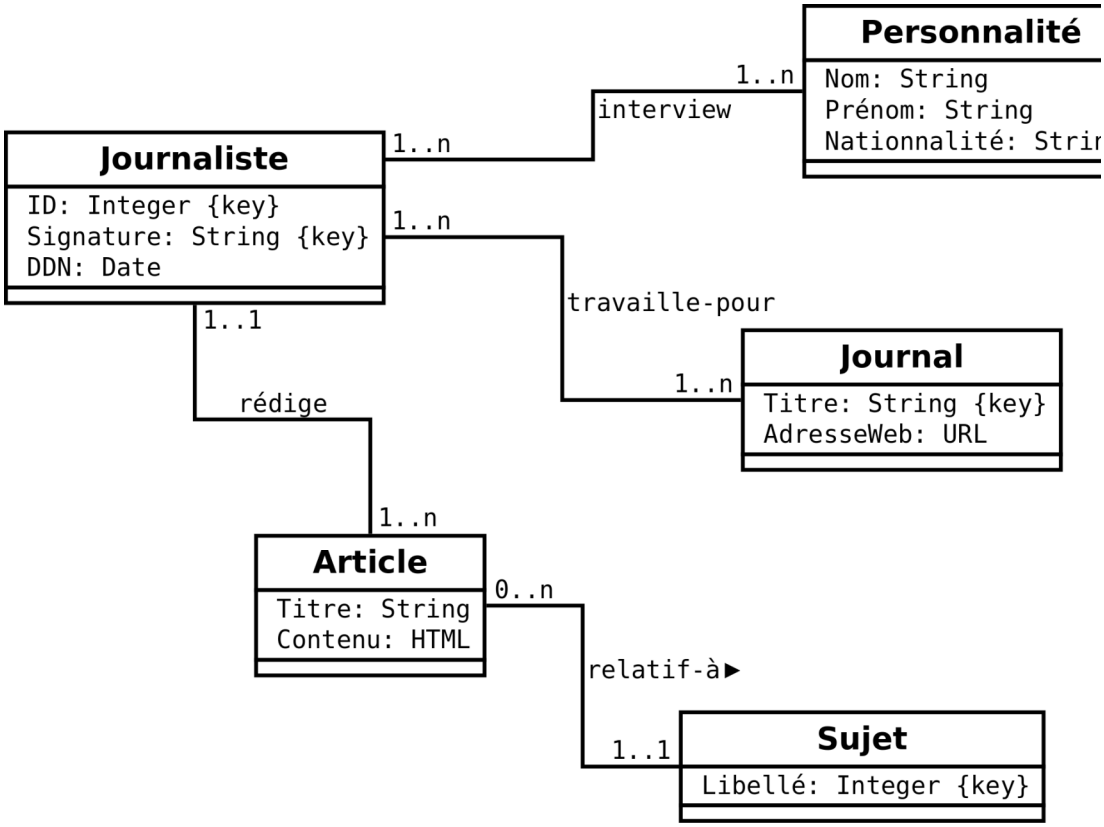


Image 13

- ☐ Un article peut être rédigé par plusieurs journalistes.
- ☐ Un article peut être publié plusieurs fois dans le même journal.
- ☐ Un article peut être publié dans un journal par un journaliste qui ne travaille pas pour ce journal.
- ☐ Il peut y avoir plusieurs articles sur le même sujet.
- ☐ Un journaliste peut interviewer une personnalité sans faire d'article à ce propos.

Logistique

Une société de transport routier veut installer un système d'information pour rendre plus efficace sa logistique. Embauché au service informatique de cette compagnie, vous êtes donc chargé de reprendre le travail déjà effectué (c'est à dire le schéma suivant).

Quelles sont les assertions vraies selon ce schéma ?

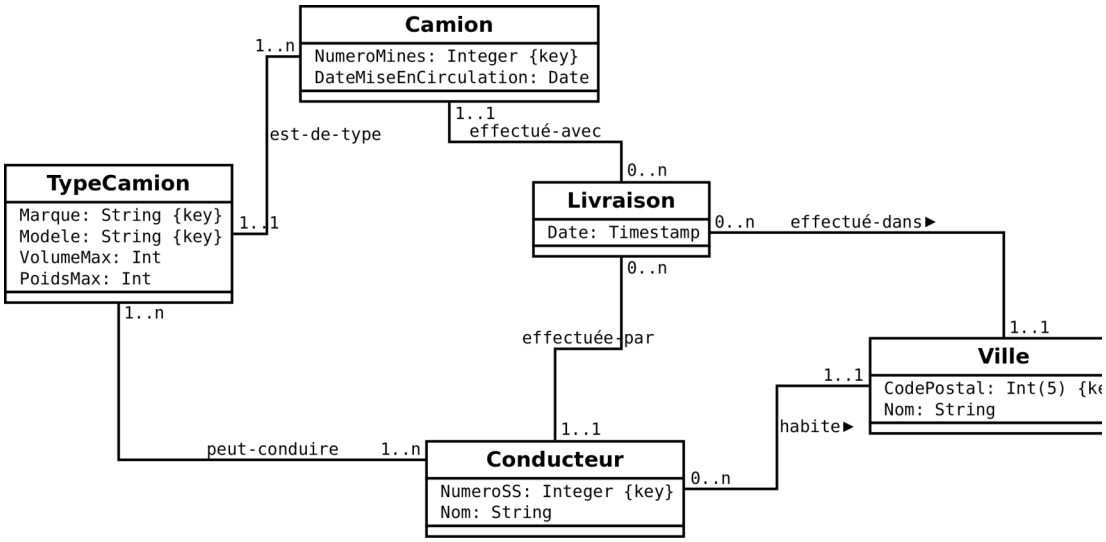


Image 14

- ☐ Un conducteur peut conduire plusieurs camions.
- ☐ Un conducteur peut conduire un camion sans y être autorisé.
- ☐ Il peut y avoir plusieurs conducteurs pour le même camion.
- ☐ Un conducteur peut livrer sa propre ville

2. Cours et intervenants

[20 min]

On souhaite réaliser une base de données pour gérer les cours dispensés dans une école d'ingénieur, ainsi que les personnes qui interviennent dans ces cours.

Chaque cours est identifié par une année et un numéro. Chaque cours a donc un numéro unique localement à chaque année. Un cours possède un titre et un type ('C' pour Cours, 'TD' ou 'TP'). Un cours possède également une date de début, et une date de fin, qui est toujours de 5 jours après la date de début.

Chaque intervenant est identifié par son nom (deux intervenants ne peuvent pas avoir le même nom). Il a un prénom, un bureau, un numéro de téléphone et des spécialités. Un bureau est défini par un centre ('R' pour Royallieu, 'BF' pour Benjamin Franklin et 'PG' pour Pierre Guillaumat), un bâtiment (une lettre de A à Z), et un numéro (inférieur à 1000). Les spécialités sont des couples de chaînes de caractères désignant un domaine (par exemple 'BD') et une spécialité (par exemple 'SGBDRO').

Chaque cours est donné par un unique intervenant.

Voici un exemple : Le cours 'Machines universelles', n°21 de l'année 2014 est donné par Alan Turing entre le 05/01/2014 et le 10/01/2014. C'est un cours de type 'C'. Alan Turing a le bureau 666 au bâtiment X de PG. Il a les numéros de téléphone 0666666666 et 0766666666. Il possède les spécialités suivantes :

- Domaine : Mathématique ; Spécialité : Cryptographie
- Domaine : Informatique ; Spécialité : Algorithmie
- Domaine : Informatique ; Spécialité : Intelligence Artificielle

Question

Réaliser le modèle UML de la base de données. Préciser les clés et les types des attributs.

3. Gestion méthodique du personnel

[30 minutes]

Le service de gestion du personnel d'une entreprise désire s'équiper d'un outil lui permettant de gérer informatiquement ses employés, leurs salaires et leurs congés. Les spécifications suivantes ont pu être mises en exergue par une analyse des besoins réalisée auprès des utilisateurs du service du personnel.

Généralités :

- tout employé est identifié par un nom, un prénom et une date de naissance ;
- tout employé remplit une fonction et appartient à un service ;
- pour chaque employé on gère la date d'embauche et la quotité (c'est à dire le pourcentage de temps travaillé par rapport au temps plein, en cas de travail à temps partiel).

Gestion des salaires :

- pour chaque employé on gère l'historique de ses salaires dans l'entreprise, chaque salaire étant affecté à une période de temps ;
- un salaire est composé d'un salaire brut, de charges patronales et de charges salariales ;
- on cherchera à partir des ces données à obtenir également le salaire chargé (brut + charges patronales), et le salaire net (brut - charges salariales), et ce en particulier pour le salaire en cours (celui que touche actuellement le salarié).

Gestion des congés :

- pour chaque employé, on mémorise chaque congé pris (posant qu'un congé concerne toujours une ou plusieurs journées entières) ;
- chaque employé a le droit aux jours de congés suivants :
  - 25 jours (pour une quotité de 1) et 25 x quotité sinon,
  - chaque fonction ouvre les droits à un certain nombre de jours de RTT,
  - chaque service ouvre les droits à un certain nombre de jours de RTT,
  - chaque tranche de 5 ans passés dans l'entreprise donne droit à 1 jour supplémentaire,
  - les employés de plus de 40 ans ont un jour supplémentaire, et ceux de plus de 50 ans deux.
- pour chaque employé on cherchera à connaître le nombre total de jours de congés autorisés, le nombre de jours pris et le nombre de jours restants sur l'année en cours.

Question

Réaliser le diagramme de classes permettant de modéliser ce problème.

Indice :

On fera apparaître les méthodes pertinentes étant donné le problème posé.
Méthodes

Introduction au passage UML-Relationnel : classes et associations

IV

A. Cours

Afin de pouvoir implémenter une base de données, il faut pouvoir traduire le modèle conceptuel en modèle logique. Cela signifie qu'il faut pouvoir convertir un modèle UML en modèle relationnel. Les modèles conceptuels sont suffisamment formels pour que ce passage soit systématisé dans la plupart des cas.

1. Transformation des classes et attributs

Objectifs

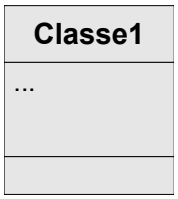
Savoir faire le passage d'un schéma conceptuel UML à un schéma relationnel pour les cas simples.

a) Transformation des classes

Méthode : Classe

Pour chaque classe non abstraite,

- on crée une relation dont le schéma est celui de la classe ;
- la clé primaire de cette relation est une des clés de la classe.



Graphique 2 Classe

```
Classe1(...)
```

Remarque

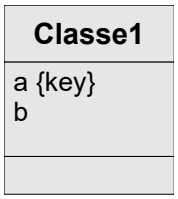
Les classes abstraites sont ignorées à ce stade, et n'étant pas instanciables, ne donnent généralement pas lieu à la création de relation.

b) Transformation des attributs

Méthode

Pour chaque attribut élémentaire et monovalué d'une classe,

- on crée un attribut correspondant dans la relation correspondant à la classe.



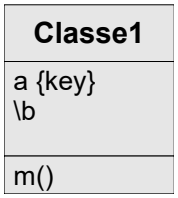
Graphique 3 Attribut

```
Classe1(#a,b)
```

c) Transformation des attributs dérivés et méthodes

Méthode : Attributs dérivés et méthodes

On ne représente pas en général les attributs dérivés ni les méthodes dans le modèle relationnel, ils seront calculés dynamiquement soit par des procédures internes à la BD (procédures stockées), soit par des procédures au niveau applicatif.



Graphique 4 Attribut dérivé et méthodes

```
Classe1(#a)
```

Complément : Attribut dérivé stockés

On peut décider (pour des raisons de performance essentiellement) de représenter l'attribut dérivé ou la méthode comme s'il s'agissait d'un attribut simple, mais il sera nécessaire dans ce cas d'ajouter des mécanismes de validation de contraintes dynamiques (avec des *triggers* par exemple) pour assurer que la valeur stockée évolue en même temps que les attributs sur lesquels le calcul dérivé porte.

Notons qu'introduire un attribut dérivé ou un résultat de méthode dans le modèle

relationnel équivaut à introduire de la redondance, ce qui est en général déconseillé, et ce qui doit être dans tous les cas contrôlé.

2. Transformation des associations

Objectifs

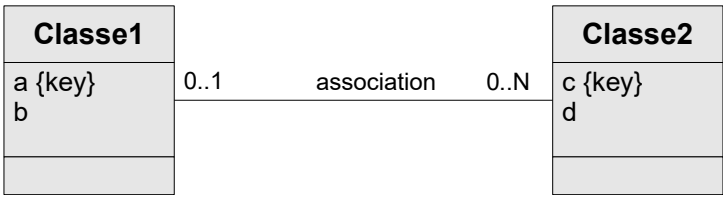
Savoir faire le passage d'un schéma conceptuel UML à un schéma relationnel pour les cas simples.

a) Transformation des associations 1:N

Méthode

Pour chaque association binaire de type 1:N :

- on ajoute à la relation côté N une clé étrangère vers la relation côté 1.



Graphique 5 Association 1:N

```
Classe1(#a,b)
Classe2(#c,d,a=>Classe1)
```

Complément

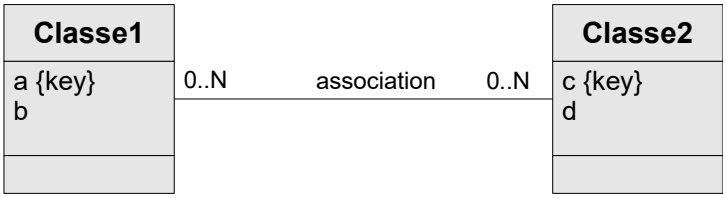
Contrainte de cardinalité minimale 1 dans les associations 1:N

b) Transformation des associations N:M

Méthode

Pour chaque association binaire de type M:N :

- on crée une nouvelle relation,
- composée de clés étrangères vers chaque relation associée,
- et dont la clé primaire est la concaténation de ces clés étrangères.



Graphique 6 Association N:M

```
Classe1(#a,b)
Classe2(#c,d)
Assoc(#a=>Classe1,#c=>Classe2)
```

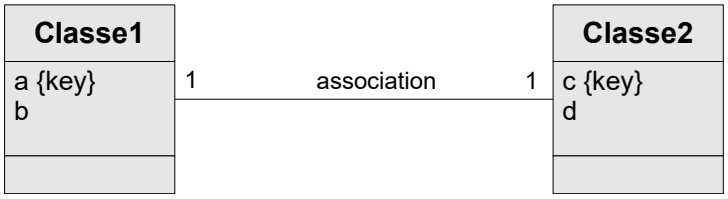
Complément

Contrainte de cardinalité minimale 1 dans les associations N:M

c) Transformation des associations 1:1 (approche simplifiée)

Méthode

La solution la plus simple et la plus générale pour transformer une association 1:1 consiste à traiter cette association 1:1 comme une association 1:N, puis à ajouter une contrainte UNIQUE sur la clé étrangère pour limiter la cardinalité maximale à 1.



Graphique 7 Association 1:1

```
Classe1(#a,b,c=>Classe2) avec c UNIQUE
Classe2(#c,d)
```

ou

```
Classe1(#a,b)
Classe2(#c,d,a=>Classe1) avec a UNIQUE
```

Remarque

Il existe toujours deux solutions selon que l'on choisit une ou l'autre relation pour accueillir la clé étrangère. Selon la cardinalité minimale, un des deux choix peut être plus pertinent.

Complément

Il est parfois possible de choisir de fusionner les deux classes au sein d'une seule relation plutôt que d'opter pour une clé étrangère.

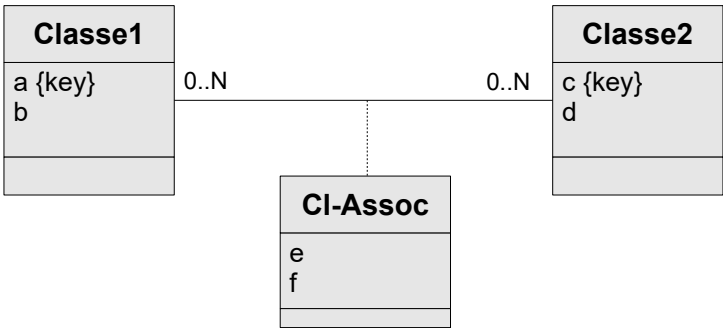
Complément

Transformation des associations 1:1 (approche générale)

d) Transformation des classes d'association

Méthode : Classe d'association N:M

Les attributs de la classe d'association sont ajoutés à la relation issue de l'association N:M.



Graphique 8 Classe assocation (N:M)

```
Classe1(#a,b)
Classe2(#c,d)
Assoc(#a=>Classe1,#c=>Classe2,e,f)
```

Complément : Classe d'association 1:N

Les attributs de la classe d'association sont ajoutés à la relation issue de la classe côté N.

Complément : Classe d'association 1:1

Les attributs de la classe d'association sont ajoutés à la relation qui a été choisie pour recevoir la clé étrangère. Si les deux classes ont été fusionnées en une seule relation, les attributs sont ajoutés à celle-ci.

B. Exercices

1. Lab I+

Description du problème

[45 min]

Un laboratoire souhaite gérer les médicaments qu'il conçoit.

- Un médicament est décrit par un nom, qui permet de l'identifier. En effet il n'existe pas deux médicaments avec le même nom. Un médicament comporte une description courte en français, ainsi qu'une description longue en latin. On gère aussi le conditionnement du médicament, c'est à dire le nombre de pilules par boîte (qui est un nombre entier).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte *« Médicament contre la chute des choux »* et pour description longue *« Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare. »*. Il est conditionné en

boîte de 13.

Ses contre-indications sont :
- CI1 : Ne jamais prendre après minuit.
- CI2 : Ne jamais mettre en contact avec de l'eau.

- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :
  - CI3 : Garder à l'abri de la lumière du soleil

Question 1

Réaliser le modèle conceptuel de données en UML du problème.

Question 2

Étendre le modèle conceptuel UML afin d'ajouter la gestion des composants. Un composant est identifié par un code unique et possède un intitulé. Tout médicament possède au moins un composant, souvent plusieurs. Tout composant peut intervenir dans la fabrication de plusieurs médicaments. Il existe des composants qui ne sont pas utilisés pour fabriquer des médicaments et que l'on veut quand même gérer.

Question 3

En mobilisant les règles adéquates, proposer un modèle logique de données correspondant en relationnel. Le repérage des domaines et des clés est obligatoire.

Question 4

Dessiner des tableaux remplis avec les données fournies en exemple, afin de montrer que le modèle fonctionne selon le besoin exprimé initialement. On pourra mettre le premier mot seulement des descriptions pour gagner du temps.

2. Usine de production

[30 minutes]

Une usine cherche à modéliser sa production de véhicules et de moteurs :

- Les véhicules sont identifiés par un numéro d'immatriculation alphanumérique et caractérisés par une couleur, dont la dénomination est une chaîne de caractères. Chaque véhicule peut comporter un unique moteur et/ou un nombre quelconque de pneus.
- Chaque moteur est monté sur un et un seul véhicule et est identifié par un numéro de série. Un moteur est caractérisé par une puissance, en chevaux.
- Tout pneu est monté sur un unique véhicule et est identifié par un numéro de série. Sa position est définie localement sur ce véhicule et par rapport à l'essieu : *Dn* pour les pneus situé sur la droite de l'essieu et *Gn* pour les pneus situés à gauche ; *n* représentant le numéro de l'essieu (1 pour celui situé devant, 2 pour la deuxième rangée, etc.). Un pneu est caractérisé par un diamètre et une largeur en pouces.
- Les moteurs, les pneus et les véhicules sont fabriqués sous une marque. Les mêmes marques peuvent fabriquer indifféremment des moteurs, des pneus et/ou des véhicules, et un véhicule d'une certaine marque peut comporter un moteur et/ou des pneus de marque différente.

Question 1

Réaliser le modèle UML de ce problème en faisant apparaître les domaines et les

clés.

Question 2

Réaliser le passage au modèle relationnel, en faisant apparaître les clés primaires, candidates et étrangères.

Question 3

Dessiner les tableaux correspondant aux relations du modèle. Instancier au minimum deux véhicules et quatre marques.

Question 4

Donner quatre exemples d'enregistrements qui seront refusés - étant données les données déjà insérées - pour quatre raisons différentes :

1. contrainte de clé sur une clé primaire
2. contrainte de clé sur une clé candidate
3. contrainte d'intégrité référentielle
4. contrainte de non nullité

V

Création et alimentation de bases de données SQL

A. Cours

SQL est un langage standardisé, implémenté par tous les SGBDR, qui permet, indépendamment de la plate-forme technologique et de façon déclarative, de définir le modèle de données, de le contrôler et enfin de le manipuler.

1. Le langage SQL

a) Exercice

Parmi les assertions suivantes concernant le langage SQL, sélectionner celles qui sont correctes.

☐	Le langage SQL permet d'implémenter physiquement un modèle logique exprimé en relationnel.
☐	Le langage SQL permet d'entrer et de sortir des données d'une base de données.
☐	Le langage SQL permet de donner et d'enlever des droits en lecture sur les données d'une base de données.
☐	Le langage SQL permet de donner et d'enlever des droits en écriture sur les données d'une base de données.
☐	Le langage SQL permet de créer une interface graphique utilisable par les utilisateurs finaux.
☐	Le langage SQL permet de faire des traitements algorithmiques complexes.
☐	Le langage SQL permet de créer une application informatique complète.

b) Le langage SQL

Définition : SQL

SQL (pour langage de requêtes structuré) est un langage déclaratif destiné à la manipulation de bases de données au sein des SGBD et plus particulièrement des SGBDR.

SQL : LDD, LCD, LMD, LCT

SQL est un langage déclaratif, il n'est donc pas a proprement parlé un langage de programmation, mais plutôt une interface standard pour accéder aux bases de données.

Il est composé de quatre sous ensembles :

- Le Langage de Définition de Données (LDD, ou en anglais DDL, *Data Definition Language*) pour créer et supprimer des objets dans la base de données (tables, contraintes d'intégrité, vues, etc.).

  Exemple de commandes : `CREATE DROP ALTER`
- Le Langage de Contrôle de Données (LCD, ou en anglais DCL, *Data Control Language*) pour gérer les droits sur les objets de la base (création des utilisateurs et affectation de leurs droits).

  Exemple de commandes : `GRANT REVOKE`
- Le Langage de Manipulation de Données (LMD, ou en anglais DML, *Data Manipulation Language*) pour la recherche, l'insertion, la mise à jour et la suppression de données. Le LMD est basé sur les opérateurs relationnels, auxquels sont ajoutés des fonctions de calcul d'agrégats et des instructions pour réaliser les opérations d'insertion, mise à jour et suppression.

  Exemple de commandes : `INSERT UPDATE DELETE SELECT`
- Le Langage de Contrôle de Transaction (LCT, ou en anglais TCL, *Transaction Control Language*) pour la gestion des transactions (validation ou annulation de modifications de données dans la BD)

  Exemple de commandes : `COMMIT ROLLBACK`

Complément : Origine du SQL

Le modèle relationnel a été inventé par E.F. Codd (Directeur de recherche du centre IBM de San José) en 1970, suite à quoi de nombreux langages ont fait leur apparition :

- IBM Sequel (Structured English Query Language) en 1977
- IBM Sequel/2
- IBM System/R
- IBM DB2

Ce sont ces langages qui ont donné naissance au standard SQL, normalisé en 1986 au États-Unis par l'ANSI pour donner SQL/86 (puis au niveau international par l'ISO en 1987).

Complément : Versions de SQL

- SQL-86 (ou SQL-87) : Version d'origine
- SQL-89 (ou SQL-1) : Améliorations mineures
- SQL-92 (ou SQL-2) : Extensions fonctionnelles majeures (types de données, opérations relationnelles, instruction LDD, transactions, etc.
- SQL-99 (ou SQL-3) : Introduction du PSM (couche procédurale sous forme de procédure stockées) et du RO
- SQL-2003 : Extensions XML
- SQL-2006 : Améliorations mineures (pour XML notamment)
- SQL-2008 : Améliorations mineures (pour le RO notamment)

Remarque : Version SQL et implémentations SGBD

Selon leur niveau d'implémentation de SQL, les SGBD acceptent ou non certaines fonctions.

Certains SGBD ayant entamé certaines implémentations avant leur standardisation définitive, ces implémentations peuvent différer de la norme.

c) Conseils pour l'apprentissage du SQL

Conseil : Le SQL, c'est du code informatique, il faut le faire fonctionner

dbdisco.crzt.fr

Conseil : Alimenter vos bases de données

Pour tester une base de données et voir si elle fonctionne, il faut l'alimenter avec des données, pour vérifier qu'elle permet effectivement de faire ce que l'on souhaite.

Une base de données sans données, c'est au mieux une base, mais ça n'est pas suffisant !

Conseil : Utiliser une référence en ligne

- Utiliser une référence correspondant à son SGBDR.
- Dans le doute PostgreSQL est un des SGBDR les plus proches du standard et dont la documentation est la mieux faite : *https://docs.postgresql.fr*[10]

10 - https://docs.postgresql.fr

Fondamental : Référence SQL : SQL-99 complete, really

Gulutzan and Pelzer, 1999 [Gulutzan and Pelzer, 1999]

2. Créer des tables en SQL (Langage de Définition de Données)

Objectifs

Maîtriser les bases du SQL pour créer et modifier des tables et des vues.

Le LDD est la partie du langage SQL qui permet de créer de façon déclarative les objets composant une BD. Il permet notamment la définition des schémas, des relations, des contraintes d'intégrité, des vues.

Rappel : Le code SQL peut être testé avec Db Disco

a) Lab I++

Description du problème

[20 min]

Un laboratoire souhaite gérer les médicaments qu'il conçoit.

- Un médicament est décrit par un nom, qui permet de l'identifier. En effet il n'existe pas deux médicaments avec le même nom. Un médicament comporte une description courte en français, ainsi qu'une description longue en latin. On gère aussi le conditionnement du médicament, c'est à dire le nombre de pilules par boîte (qui est un nombre entier).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte *« Médicament contre la chute des choux »* et pour description longue *« Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare. »*. Il est conditionné en boîte de 13.

  Ses contre-indications sont :
  - CI1 : Ne jamais prendre après minuit.
  - CI2 : Ne jamais mettre en contact avec de l'eau.
- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :
  - CI3 : Garder à l'abri de la lumière du soleil

Question 1

Réaliser le modèle conceptuel de données en UML du problème.

Question 2

En mobilisant les règles adéquates, proposer un modèle logique de données correspondant en relationnel. Le repérage des domaines et des clés est obligatoire.

Question 3

Créer une base de données en SQL correspondant au modèle relationnel.

Question 4

Insérer les données fournies en exemple dans la base de données.

b) Création de tables

Rappel

Qu'est ce que le SQL ?

Introduction

La création de table est le fondement de la création d'une base de données en SQL.

Définition : Création de table

La création de table est la définition d'un schéma de relation en intension, par la spécification de tous les attributs le composant avec leurs domaines respectifs.

Syntaxe

```
1  CREATE TABLE nom_table (
2    nom_colonne1 domaine1,
3    nom_colonne2 domaine2,
4    ...
5    nom_colonneN domaineN
6  );
```

Exemple

```
1  CREATE TABLE Personne (
2    Nom VARCHAR(25),
3    Prenom VARCHAR(25),
4    Age NUMERIC(3)
5  );
```

Attention : Contrainte d'intégrité

La définition des types n'est pas suffisante pour définir un schéma relationnel, il faut lui adjoindre la définition de **contraintes d'intégrité**, qui permette de poser les notions de clé, d'intégrité référentielle, de restriction de domaines, etc.

c) Domaines de données

Introduction

Un attribut d'une relation est défini pour un certain domaine ou type. Les types de données disponibles en SQL varient d'un SGBD à l'autre, on peut néanmoins citer un certain nombre de types standards que l'on retrouve dans tous les SGBD.

Fondamental : Les types standard

- INTEGER ou INT, SMALLINT
- NUMERIC(X)
- DECIMAL(X,Y) ou NUMERIC(X,Y)
- FLOAT(X), REAL
- CHAR(X)
- VARCHAR(X)
- DATE (AAAA-MM-JJ)
- DATETIME (AAAA-MM-JJ HH:MM:SS)

Les types numériques standard

- **Les nombres entiers**

  INTEGER (ou INT) et SMALLINT, permettent de coder des entiers sur 4 octets (-2.147.483.648 à 2.147.483.647) ou 2 octets (-32.768 à 32.767).
- **Les nombres entiers**

  NUMERIC(X) désigne un entier de X chiffres au maximum.
- **Les nombres décimaux**

  DECIMAL(X,Y), où X et Y sont optionnels et désignent respectivement le nombre de chiffres maximum pouvant composer le nombre, et le nombre de chiffres après la virgule.

  NUMERIC(X,Y) est un synonyme standard.
- **Les nombres à virgule flottante**

  FLOAT(X), avec X définissant la précision (nombre de bits de codage de la mantisse).

  REAL est un synonyme standard de FLOAT(24).

Conseil : FLOAT versus DECIMAL

Il est conseillé d'utiliser DECIMAL qui est un nombre exact, plutôt que FLOAT qui est un nombre approximatif, si la précision requise est suffisante. FLOAT sera réservé typiquement à des calculs scientifiques nécessitant un degré de précision supérieur.

Les types chaîne de caractères standard

On distingue principalement les types CHAR(X) et VARCHAR(X), où X est obligatoire et désigne la longueur de la chaîne.

- CHAR définit des chaînes de longueur fixe (complétée à droites par des espaces, si la longueur est inférieure à X) ;
- et VARCHAR des chaînes de longueurs variables.

CHAR et VARCHAR sont généralement limités à 255 caractères. La plupart des SGBD proposent des types, tels que TEXT ou CLOB (Character Long Object), pour représenter des chaînes de caractères longues, jusqu'à 65000 caractères par exemple.

Les types date standard

Les types date sont introduits avec la norme SQL2. On distingue :

- DATE qui représente une date selon un format de type "AAAA-MM-JJ" ;
- et DATETIME qui représente une date plus une heure, dans un format tel que "AAAA-MM-JJ HH:MM:SS".

Complément : Types de données PostgreSQL

https://www.postgresql.org/docs/current/static/datatype.html[11]

Complément : Les autres types

En fonction du SGBD, il peut exister de nombreux autres types. On peut citer par exemple :

- MONEY pour représenter des décimaux associés à une monnaie,
- BOOLEAN pour représenter des booléens,
- BLOB (pour Binary Long Oject) pour représenter des données binaires tels que des documents multimédia (images bitmap, vidéo, etc.)
- ...

d) La valeur NULL

L'absence de valeur, représentée par la valeur `NULL`, est une information fondamentale en SQL, qu'il ne faut pas confondre avec la chaîne de caractère *espace* ou bien la valeur 0. Il ne s'agit pas d'un type, ni d'une contrainte, mais d'une valeur possible dans tous les types.

Fondamental

Par défaut en SQL `NULL` fait partie du domaine, il faut l'exclure explicitement par la clause `NOT NULL` après la définition de type, si on ne le souhaite pas.

Syntaxe

```
CREATE TABLE nom_table (
  nom_colonne1 domaine1 NOT NULL,
  nom_colonne2 domaine2,
  ...
  nom_colonneN domaineN NOT NULL
);
```

e) Contraintes d'intégrité

Fondamental

- PRIMARY KEY (<liste d'attibuts>)
- UNIQUE (<liste d'attibuts>)
- FOREIGN KEY (<liste d'attibuts>) REFERENCES <nom table>(<nom colonnes>)
- CHECK (<condition>)

Une contrainte d'intégrité est une règle qui définit la cohérence d'une donnée ou d'un ensemble de données de la BD.

Il existe deux types de contraintes :

- sur une colonne unique,
- ou sur une table lorsque la contrainte porte sur une ou plusieurs colonnes.

Les contraintes sont définies au moment de la création des tables.

11 - https://www.postgresql.org/docs/current/static/datatype.html

Définition : Contraintes d'intégrité sur une colonne

Les contraintes d'intégrité sur une colonne sont :

- PRIMARY KEY : définit l'attribut comme la clé primaire
- UNIQUE : interdit que deux tuples de la relation aient la même valeur pour l'attribut.
- REFERENCES <nom table> (<nom colonnes>) : contrôle l'intégrité référentielle entre l'attribut et la table et ses colonnes spécifiées
- CHECK (<condition>) : contrôle la validité de la valeur de l'attribut spécifié dans la condition dans le cadre d'une restriction de domaine

Définition : Contraintes d'intégrité sur une table

Les contraintes d'intégrité sur une table sont :

- PRIMARY KEY (<liste d'attibuts>) : définit les attributs de la liste comme la clé primaire
- UNIQUE (<liste d'attibuts>) : interdit que deux tuples de la relation aient les mêmes valeurs pour l'ensemble des attributs de la liste.
- FOREIGN KEY (<liste d'attibuts>) REFERENCES <nom table>(<nom colonnes>) : contrôle l'intégrité référentielle entre les attributs de la liste et la table et ses colonnes spécifiées
- CHECK (<condition>) : contrôle la validité de la valeur des attributs spécifiés dans la condition dans le cadre d'une restriction de domaine

Syntaxe

```
1  CREATE TABLE nom_table (
2    nom_colonne1 domaine1 <contraintes colonne1>,
3    nom_colonne2 domaine2 <contraintes colonne2>,
4    ...
5    nom_colonneN domaineN <contraintes colonneN>,
6   <contraintes de table>
7  );
```

Exemple

```
1  CREATE TABLE Personne (
2    N°SS CHAR(13) PRIMARY KEY,
3    Nom VARCHAR(25) NOT NULL,
4    Prenom VARCHAR(25) NOT NULL,
5    Age INTEGER CHECK (Age BETWEEN 18 AND 65),
6    Mariage CHAR(13) REFERENCES Personne(N°SS),
7    UNIQUE (Nom, Prenom)
8  );
```

Remarque : Clé candidate

La clause UNIQUE NOT NULL sur un attribut ou un groupe d'attributs définit une clé candidate non primaire.

À condition qu'aucun des attributs du groupe ne soit lui même UNIQUE (sinon la clause de minimalité d'une clé n'est pas respectée).

Remarque

Les contraintes sur une colonne et sur une table peuvent être combinées dans la

définition d'un même schéma de relation.

Remarque

Une contrainte sur une colonne peut toujours être remplacée par une contrainte sur une table.

f) Exemple de contraintes d'intégrité

Exemple

```
1  CREATE TABLE Adresse (
2  CP INTEGER NOT NULL,
3  Pays VARCHAR(50) NOT NULL,
4  Initiale CHAR(1) CHECK (Initiale = LEFT(Pays, 1)),
5  PRIMARY KEY (CP, Pays)
6  );
7
8  CREATE TABLE Personne (
9  N°SS CHAR(13) PRIMARY KEY,
10 Nom VARCHAR(25) NOT NULL,
11 Prenom VARCHAR(25) NOT NULL,
12 Age INTEGER CHECK (Age BETWEEN 18 AND 65),
13 Mariage CHAR(13) REFERENCES Personne(N°SS),
14 Codepostal INTEGER,
15 Pays VARCHAR(50),
16 UNIQUE (Nom, Prenom),
17 FOREIGN KEY (Codepostal, Pays) REFERENCES Adresse (CP, Pays)
18 );
```

Dans la définition de schéma précédente on a posé les contraintes suivantes :

- La clé primaire de Personne est N°SS et la clé primaire de Adresse est (CP, Pays).
- Nom, Prénom ne peuvent pas être null et (Nom, Prénom) est une clé.
- Age doit être compris entre 18 et 65 et Initiale doit être la première lettre de Pays (avec la fonction LEFT qui renvoie la sous chaîne à gauche de la chaîne passée en premier argument, sur le nombre de caractères passés en second argument)
- Mariage est clé étrangère vers Personne et (Codepostal, Pays) est une clé étrangère vers Adresse.

Exemple : Réécriture avec uniquement des contraintes de table

```
1  CREATE TABLE Adresse (
2  CP INTEGER NOT NULL,
3  Pays VARCHAR(50) NOT NULL,
4  Initiale CHAR(1),
5  PRIMARY KEY (CP, Pays),
6  CHECK (Initiale = LEFT(Pays, 1))
7  );
8
9  CREATE TABLE Personne (
10 N°SS CHAR(13) ,
11 Nom VARCHAR(25) NOT NULL,
12 Prenom VARCHAR(25) NOT NULL,
13 Age INTEGER,
14 Mariage CHAR(13),
15 Codepostal INTEGER,
16 Pays VARCHAR(50),
17 PRIMARY KEY (N°SS),
18 UNIQUE (Nom, Prenom),
```

```
19  CHECK (Age BETWEEN 18 AND 65),
20  FOREIGN KEY (Mariage) REFERENCES Personne(N°SS),
21  FOREIGN KEY (Codepostal, Pays) REFERENCES Adresse (CP, Pays)
22  );
```

Ce schéma est strictement le même que le précédent, simplement les contraintes ont toutes été réécrites comme des contraintes de table.

3. Insérer, modifier et supprimer des données en SQL (Langage de Manipulation de Données)

Objectifs

Maîtriser les bases du SQL pour entrer, modifier et effacer des données dans les tables.

a) Exercice

Quelle valeur renvoie la dernière instruction SQL de la liste ci-dessous :

```
1   CREATE TABLE t (
2     a integer,
3     b integer,
4     c integer);
5
6   INSERT INTO t VALUES (0, 0, 0);
7
8   INSERT INTO t VALUES (1, 0, 0);
9
10  SELECT sum(a) + count(b) FROM t;
```

b) Insertion de données

Le langage SQL fournit des instructions pour ajouter des nouveaux tuples à une relation. Il offre ainsi une interface standard pour ajouter des information dans une base de données.

Il existe deux moyens d'ajouter des données, soit par fourniture directe des valeurs des propriétés du tuple à ajouter, soit par sélection des tuples à ajouter depuis une autre relation.

Syntaxe : Insertion directe de valeurs

```
1   INSERT INTO <Nom de la relation> (<Liste ordonnée des propriétés à
    valoriser>)
2   VALUES (<Liste ordonnée des valeurs à affecter aux propriétés
    spécifiées ci-dessus>)
```

Exemple

```
1   INSERT INTO Virement (Date, Montant, Objet)
2   VALUES (14-07-1975, 1000, 'Prime de naissance');
```

Remarque

- Les propriétés non valorisées sont affectées à la valeur NULL.
- Il est possible de ne pas spécifier les propriétés à valoriser, dans ce cas, toutes les propriétés de la relation seront considérées, dans leur ordre de définition dans la relation (à n'utiliser que dans les cas les plus simples).

Attention : Chaînes de caractères

- Les chaînes de caractère doivent être protégées par des apostrophes : '
- Pour insérer une apostrophe, doubler le caractère : ''

Exemple

```
1  INSERT INTO livre (titre) VALUES ('L''Attrape-cœurs')
```

Attention : Dates

La saisie des dates peut se faire de plusieurs façons dépendante du SGBD, la méthode la plus systématique consiste à utiliser la fonction `TO_DATE(chaîne,format)` où la chaîne de caractère respecte le format.

Par exemple `TO_DATE('20170330','YYYYMMDD')` ou `TO_DATE('30-03-2017','DD-MM-YYYY')` désignent tous les deux le 30 mars 2017.

Exemple

```
1  INSERT INTO livre (pubdate) VALUES (TO_DATE('20170330','YYYYMMDD'))
```

Complément

Fonctions SQL

Complément

https://www.postgresql.org/docs/current/static/functions-formatting.html[12]

c) Insertion de valeurs par l'intermédiaire d'une sélection

Syntaxe : Insertion de données par sélection de valeurs existantes dans la base

```
1  INSERT INTO <Nom de la relation> (<Liste ordonnée des propriétés à
   valoriser>)
2  SELECT ...
```

L'instruction SELECT projetant un nombre de propriétés identiques aux propriétés à valoriser.

Exemple

```
1  INSERT INTO Credit (Date, Montant, Objet)
```

12 - https://www.postgresql.org/docs/current/static/functions-formatting.html

```
2  SELECT Date, Montant, 'Annulation de débit'
3  FROM Debit
4  WHERE Debit.Date = 25-12-2001;
```

Dans cet exemple tous les débits effectués le 25 décembre 2001, sont re-crédités pour le même montant (et à la même date), avec la mention annulation dans l'objet du crédit. Ceci pourrait typiquement réalisé en cas de débits erronés ce jour là.

Remarque

- Les propriétés non valorisées sont affectées à la valeur NULL.
- Il est possible de ne pas spécifier les propriétés à valoriser, dans ce cas, toutes les propriétés de la relation seront considérées, dans leur ordre de définition dans la relation (à n'utiliser que dans les cas les plus simples).

d) Mise à jour de données

Le langage SQL fournit une instruction pour modifier des tuples existants dans une relation.

Syntaxe : Mise à jour directe de valeurs

```
1  UPDATE <Nom de la relation>
2  SET <Liste d'affectations Propriété=Valeur, Propriété=Valeur>
3  WHERE <Condition pour filtrer les tuples à mettre à jour>
```

```
1  UPDATE r
2  SET a=1, b='x'
3  WHERE c=0
```

Exemple : Mise à jour directe de valeurs

```
1  UPDATE Compte
2  SET Monnaie='Euro'
3  WHERE Monnaie='Franc'
```

Exemple : Mise à jour par calcul sur l'ancienne valeur

```
1  UPDATE Compte
2  SET Total=Total * 6,55957
3  WHERE Monnaie='Euro'
```

e) Suppression de données

Le langage SQL fournit une instruction pour supprimer des tuples existants dans une relation.

Syntaxe

```
1  DELETE FROM <Nom de la relation>
2  WHERE <Condition pour filtrer les tuples à supprimer>
```

Exemple : Suppression de tous les tuples d'une relation

```
1 DELETE FROM FaussesFactures
```

Exemple : Suppression sélective

```
1 DELETE FROM FaussesFactures
2 WHERE Auteur='Moi'
```

4. Supprimer et modifier des tables en SQL (Langage de Définition de Données)

Objectifs

Maîtriser les bases du SQL pour créer et modifier des tables et des vues.

Le LDD permet de créer les objets composant une BD de façon déclarative. Il permet notamment la définition des schémas des relations, la définition des contraintes d'intégrité, la définition de vues relationnelles.

a) Suppression d'objets

Il est possible de supprimer des objets de la BD, tels que les tables ou les vues.

Syntaxe

```
1 DROP <type objet> <nom objet>
```

Exemple

```
1 DROP TABLE Personne;
2 DROP VIEW Employe;
```

b) Modification de tables

Introduction

L'instruction ALTER TABLE permet de modifier la définition d'une table (colonnes ou contraintes) préalablement créée.

Cette commande absente de SQL-89 est normalisée dans SQL-92

Syntaxe : Ajout de colonne

```
1 ALTER TABLE <nom de table>
2 ADD <définition de colonne>
```

Syntaxe : Suppression de colonne

```
ALTER TABLE <nom de table>
DROP <nom de colonne>
```

Syntaxe : Ajout de contrainte

```
ALTER TABLE <nom de table>
ADD <définition de contrainte de table>
```

Remarque : Modification de table sans donnée sans la commande ALTER

Pour modifier une table ne contenant pas encore de donnée, la commande ALTER n'est pas indispensable, l'on peut supprimer la table à modifier (DROP) et la recréer telle qu'on la souhaite. Notons néanmoins que si la table est référencée par des clauses FOREIGN KEY, cette suppression sera plus compliquée, car il faudra également supprimer et recréer les tables référençantes (ce qui ce complique encore si ces dernières contiennent des données).

Remarque : Modification de table avec données sans la commande ALTER

Pour modifier une table contenant des données, la commande ALTER n'est pas indispensable. On peut en effet :

1. Copier les données dans une table temporaire de même schéma que la table à modifier
2. Supprimer et recréer la table à modifier avec le nouveau schéma
3. Copier les données depuis la table temporaire vers la table modifiée

c) Exemple de modifications de tables

Table initiale

Soit une table initiale telle que définie ci-après.

```
CREATE TABLE Personne (
 pk_n NUMERIC(4),
 nom VARCHAR(50),
 prenom VARCHAR(50),
 PRIMARY KEY (pk_n)
);
```

Modifications

On décide d'apporter les aménagements suivants à la table : on définit "nom" comme UNIQUE et on supprime le champ "prenom".

```
ALTER TABLE Personne
ADD UNIQUE (nom);

ALTER TABLE Personne
DROP prenom;
```

Table finale

La table obtenue après modification est identique à la table qui aurait été définie directement telle que ci-après.

```sql
CREATE TABLE Personne (
 pk_n NUMERIC(4),
 nom VARCHAR(50),
 PRIMARY KEY (pk_n),
  UNIQUE (nom)
);
```

B. Exercices

1. The show

[60 minutes]
Soit le schéma relationnel suivant décrivant un système de réservations de places de spectacles :

```
SPECTACLE (#nospectacle:int, nom:str, durée:minutes, type:{théâtre|danse|concert})
SALLE (#nosalle:int, nbplaces:int)
REPRESENTATION (#date:timestamp, #nospectacle=>SPECTACLE, #nosalle=>SALLE, prix:decimal)
```

a) Exercice : Étude du schéma

Exercice

Le schéma permet de gérer un espace de spectacles composés d'un ensemble de salles.

- ○ Vrai
- ○ Faux

Exercice

La durée du spectacle est donnée en minutes.

- ○ Vrai
- ○ Faux

Exercice

Le type SQL `TIMESTAMP` désigne un instant à la seconde près, par exemple : `20/03/2017 10:37:22`.

- ○ Vrai
- ○ Faux

Exercice

On peut supposer que le champ `date` désigne le début d'une représentation.

- ○ Vrai
- ○ Faux

Exercice

Deux spectacles identiques peuvent être joués en même temps dans deux salles différentes.

- ○ Vrai
- ○ Faux

Exercice

Deux spectacles différents peuvent être joués en même temps dans la même salle.

- ○ Vrai
- ○ Faux

b) Exercice

Question 1

Retro-concevoir le MCD en UML.

Question 2

Proposer des contraintes d'intégrité réalistes pour ce schéma (en français).

Question 3

Proposer une définition du schéma en SQL qui prenne en compte certaines de ces contraintes.

Question 4

Insérer des données réalistes dans votre schéma afin de vérifier son bon fonctionnement.

2. Du producteur au consommateur

[60 min]

Soit le modèle relationnel suivant :

```
1 Producteur(#raison_sociale:chaîne(25), ville:chaîne(255))
2 Consommateur(#login:chaîne(10), #email:chaîne(50),
  nom:chaîne(50),prenom:chaîne(50), ville:chaîne(255))
3 Produit(#id:entier, description:chaîne(100), produit-par=>Producteur,
  consomme-par-login=>Consommateur, consomme-par-email=>Consommateur)
```

On ajoute que :

- `(nom,prenom,ville)` est une clé candidate de `Consommateur`
- Tous les produits sont produits
- Tous les produits ne sont pas consommés

Question 1

Rétro-concevez le modèle conceptuel sous-jacent à ce modèle relationnel.

Question 2

Établissez le code LDD standard permettant d'implémenter ce modèle en SQL.

Question 3

Insérez les données dans votre base de données correspondant aux assertions suivantes :

- L'entreprise de Compiègne "Pommes Picardes SARL" a produit 4 lots de pommes, et 2 lots de cidre.
- Il existe trois utilisateurs consommateurs dans la base, donc les adresses mails sont :

  Al.Un@compiegne.fr - Bob.Deux@compiegne.fr - Charlie.Trois@compiegne.fr

  Ce sont des employés de la ville de Compiègne qui habitent cette ville. Leur mail est construit sur le modèle Prenom.Nom@compiegne.fr. Leur login est leur prénom.

Question 4

Modifiez les données de votre base de données pour intégrer les assertions suivantes :

- 1 lot de pommes a été consommé par Al Un.
- 2 lot de pomme ont été consommés par Bob Deux.
- Tous les lots de cidre ont été consommés par Al Un.

Question 5

Charlie Trois n'ayant rien consommé, modifiez votre base de données afin de le supprimer de la base.

L'héritage dans la modélisation conceptuelle de données

VI

A. Cours

1. Introduction à l'héritage

Objectifs

Savoir utiliser l'héritage lors une modélisation

a) Exercice : Mariages

Étant donné le schéma UML suivant, quelles sont les assertions vraies ?

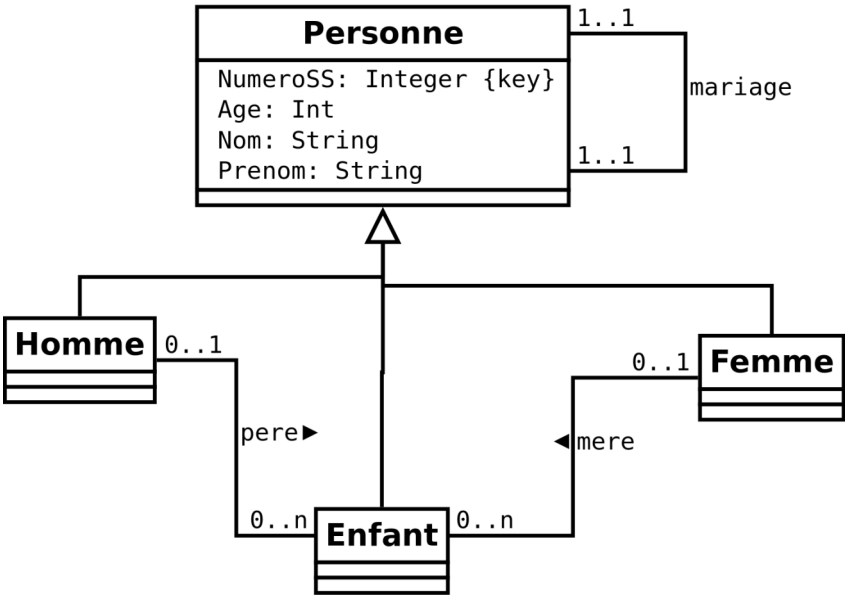


Image 15

- ☐ Les mariages homosexuels sont possibles.
- ☐ La polygamie est possible.
- ☐ Une femme peut ne pas être mariée.
- ☐ Les enfants peuvent être plus âges que leurs parents.
- ☐ Deux hommes peuvent avoir un enfant ensemble.
- ☐ Une femme peut avoir plusieurs enfants.
- ☐ Un enfant a obligatoirement deux parents.
- ☐ Les enfants peuvent se marier.
- ☐ Tous les enfants sont mariés.
- ☐ Les personnes mariées ont toujours le même nom.

b) Héritage

Définition : Héritage

L'héritage est l'association entre deux classes permettant d'exprimer que l'une est plus générale que l'autre. L'héritage implique une transmission automatique des propriétés (attributs et méthodes) d'une classe A à une classe A'.

Dire que A' hérite de A équivaut à dire que A' est une sous-classe de A. On peut également dire que A est une généralisation de A' et que A' est une spécialisation de A.

Syntaxe

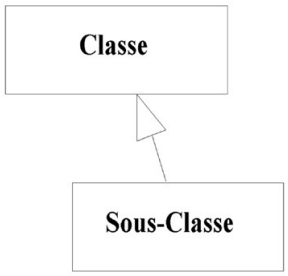
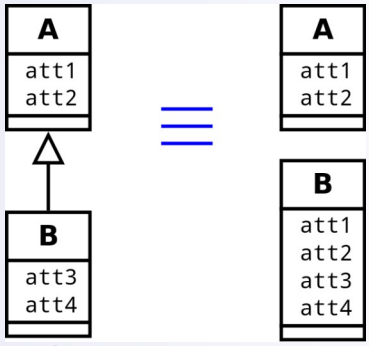


Image 16 Notation de l'héritage en UML

Fondamental : Factorisation

Outre qu'il permet de représenter une relation courante dans le monde réel, l'héritage a un avantage pratique, celui de factoriser la définition de propriétés identiques pour des classes proches.

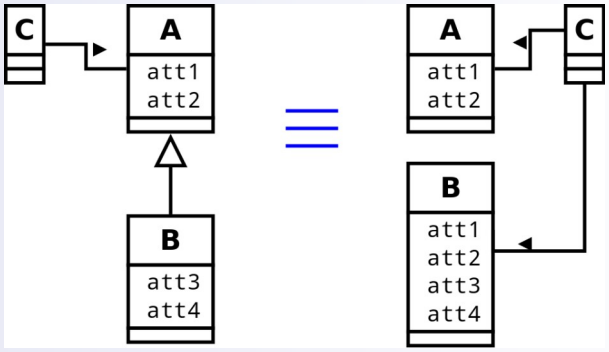


Héritage et factorisation

Fondamental : Is-a

L'héritage permet de représenter la relation "est-un" entre deux objets (*is-a* en anglais).

Donc tout ce qui est vrai pour la classe mère est vrai pour ses classes filles. En particulier si une classe C exprime une association avec une classe A dont hérite B, cela signifie que C peut être associée à B.



Héritage et propriété "is-a"

Exemple : La classe Conducteur

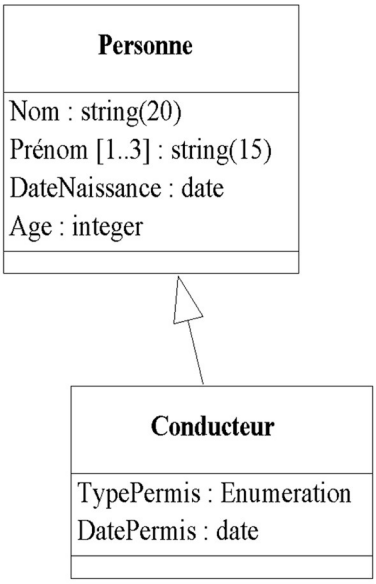


Image 17 Représentation d'héritage en UML

Dans cet exemple la classe Conducteur hérite de la classe Personne, ce qui signifie qu'un objet de la classe conducteur aura les attributs de la classe Conducteur (TypePermis et DatePermis) mais aussi ceux de la classe Personne (Nom, Prénom, DateNaissance et Age). Si la classe Personne avait des méthodes, des associations..., la classe Conducteur en hériterait de la même façon.

c) Exemple : Des voitures et des conducteurs

Exemple

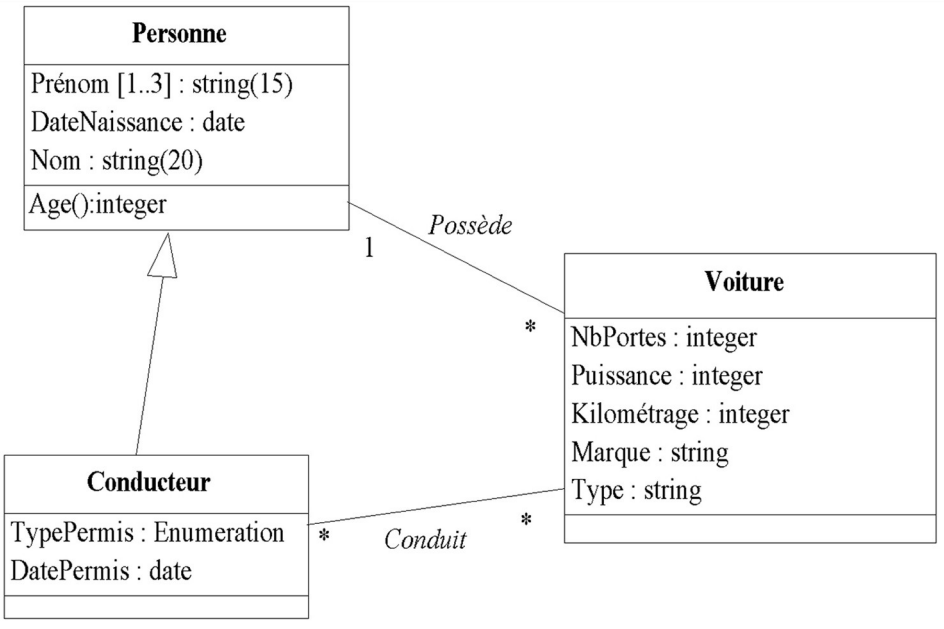


Image 18 Exemple très simple de diagramme de classes

Les relations de ce diagramme expriment que les conducteurs sont des personnes qui ont un permis ; que toute voiture est possédée par une unique personne (qui peut en posséder plusieurs) ; que les voitures peuvent être conduites par des conducteurs et que les conducteurs peuvent conduire plusieurs voitures.

Attention

Le but d'une modélisation UML n'est pas de représenter la réalité dans l'absolu, mais plutôt de proposer une vision d'une situation réduite aux éléments nécessaires pour répondre au problème posé. Donc une modélisation s'inscrit toujours dans un contexte, et en cela l'exemple précédent reste limité car son contexte d'application est indéfini.

d) Classe abstraite

Définition : Classe abstraite

Une classe abstraite est une classe non instanciable. Elle exprime donc une généralisation abstraite, qui ne correspond à aucun objet existant du monde.

Attention : Classe abstraite et héritage

Une classe abstraite est **toujours héritée**. En effet sa fonction étant de généraliser, elle n'a de sens que si des classes en héritent. Une classe abstraite peut être héritée par d'autres classes abstraites, mais en fin de chaîne des classes non abstraites doivent être présentes pour que la généralisation ait un sens.

Syntaxe : Italique

On note les classes abstraites en italique.

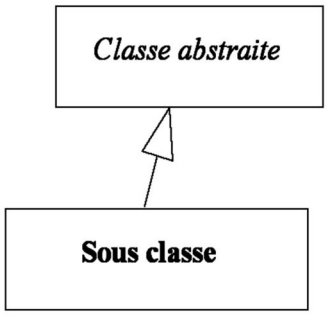
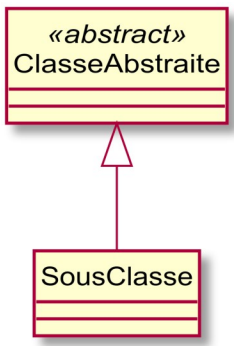


Image 19 Notation d'une classe abstraite en UML

Syntaxe : <<abstract>>

On peut utiliser le stéréotype `<<abstract>>` pour noter les classes abstraites (cette représentation est plus évidente à lire).



Notation <<abstract>>

Exemple : Exemple de classes abstraites

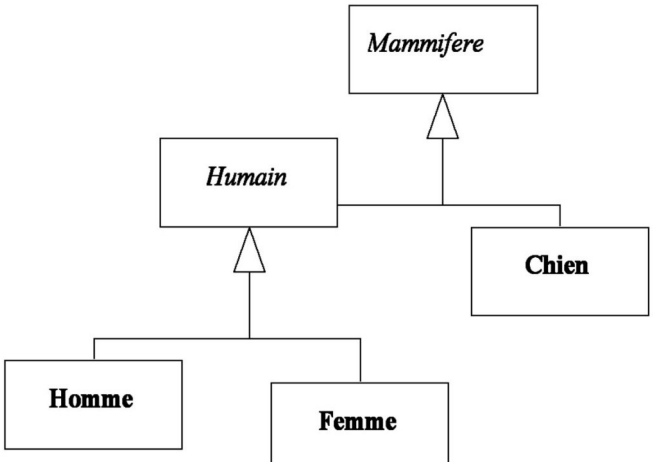


Image 20 Des chiens et des hommes

Dans la représentation précédente on a posé que les hommes, les femmes et les chiens étaient des objets instanciables, généralisés respectivement par les classes mammifère et humain, et mammifère.

Selon cette représentation, il ne peut donc exister de mammifères qui ne soient ni des hommes, ni des femmes ni des chiens, ni d'humains qui ne soient ni des hommes ni des femmes.

Complément

Stéréotype

2. Notions avancées pour l'usage de l'héritage en modélisation des BD

Objectifs

Savoir utiliser l'héritage lors une modélisation

a) Lab II

[15 min]

Description du problème

Un laboratoire souhaite gérer les médicaments qu'il conçoit.

- Un médicament est décrit par un nom, qui permet de l'identifier. En effet il n'existe pas deux médicaments avec le même nom. Un médicament comporte une description courte en français, ainsi qu'une description longue en latin. On gère aussi le conditionnement du médicament, c'est à dire le nombre de pilules par boîte (qui est un nombre entier).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.
- Tout médicament possède au moins un composant, souvent plusieurs. Un

composant est identifié par un code unique et possède un intitulé. Tout composant peut intervenir dans la fabrication de plusieurs médicaments. Il existe des composants qui ne sont pas utilisés pour fabriquer des médicaments et que l'on veut quand même gérer.

- Il existe des composants naturels et des composants artificiels. Pour les composants naturels, on gère l'espèce végétale qui porte le composant. Pour les composants artificiels, on gère le nom de la société qui le fabrique.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte *« Médicament contre la chute des choux »* et pour description longue *« Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare. »*. Il est conditionné en boîte de 13.

  Ses contre-indications sont :
  - CI1 : Ne jamais prendre après minuit.
  - CI2 : Ne jamais mettre en contact avec de l'eau.

  Ses composants sont le **HG79** et le **SN50**.
- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :
  - CI3 : Garder à l'abri de la lumière du soleil

  Son unique composant est le **HG79**.
- Les composants existants sont :
  - **HG79** : "Vif-argent allégé" ; il s'agit d'un composant naturel extrait de l'**edelweiss**.
  - **HG81** : "Vif-argent alourdi" ; il s'agit aussi d'un composant naturel extrait de l'**edelweiss**.
  - **SN50** : "Pur étain" ; il s'agit d'un composant artificiel fabriqué par **Lavoisier et fils SA**.

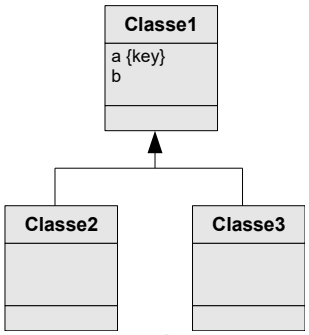
Question

Réaliser le modèle conceptuel de données en UML du problème.

b) Héritage complet

Définition : Héritage complet et presque complet

Un héritage est complet si ses classes filles n'ont aucune caractéristiques (attributs, méthodes, associations) propres.



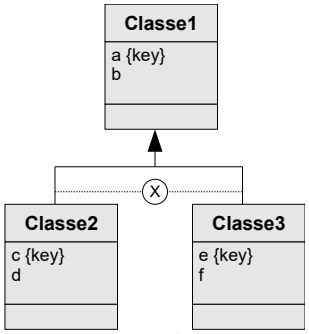
Graphique 9 Héritage complet

Un héritage est presque complet si les classes filles ont des méthodes propres, quelques attributs propres, et **aucune association propre**.

c) Héritage exclusif

Définition : Héritage exclusif

Un héritage est exclusif si les objets d'une classe fille ne peuvent appartenir aussi à une autre classe fille. On peut le noter avec la contrainte `{X}` sur le diagramme UML (`{XT}` si la classe mère est abstraite).



Graphique 10 Héritage exclusif

Définition : Héritage non exclusif

L'héritage non exclusif est une forme d'héritage qui exprime qu'un objet peut appartenir à deux classes filles en même temps.

Syntaxe : Notation des héritages exclusifs

Normalement la notation UML par défaut désigne un héritage non exclusif.

Mais l'héritage exclusif étant plus commun, on conviendra plutôt qu'un schéma UML qui ne comporte que des héritages non contraints exprime par défaut des héritages exclusifs.

Une note peut être ajoutée précisant que tous les héritages sont exclusifs pour lever toute ambiguïté.

En revanche un schéma qui comporte explicitement des contraintes sur certains héritages est pas sur d'autres permet effectivement d'exprimer de l'héritage non exclusif.

Une dernière possibilité, moins standard, est de marquer d'une contrainte de type {NON X} un héritage non exclusif.

Méthode : On évitera l'héritage non exclusif pour la conception de BD

Bien que cette forme d'héritage soit autorisée en modélisation conceptuelle de bases de données et en UML, on évitera de la mobiliser, sauf en cas d'apport vraiment important d'expressivité, car elle a tendance à complexifier la modalisation, que ce soit au niveau de son interprétation humaine ou de son implémentation en machine.

Il est toujours possible d'exprimer différemment un héritage non exclusif :

- en multipliant les classes filles pour les singulariser,
- et/ou en utilisant la composition pour factoriser les parties communes.

d) Héritage multiple

Définition : Héritage multiple

L'héritage multiple est une forme d'héritage qui exprime qu'une classe fille hérite de plusieurs classes mères.

Syntaxe : Notation des héritages multiples

On conseillera d'ajouter une note lors de l'utilisation d'un héritage multiple, expliquant le choix de modélisation et actant qu'il ne s'agit pas d'une erreur.

Méthode : On évitera l'héritage multiple pour la conception de BD

On évitera d'utiliser l'héritage multiple en modélisation de bases de données, sauf en cas d'apport vraiment important d'expressivité.

Il est toujours possible de remplacer un héritage multiple par plusieurs héritages simples.

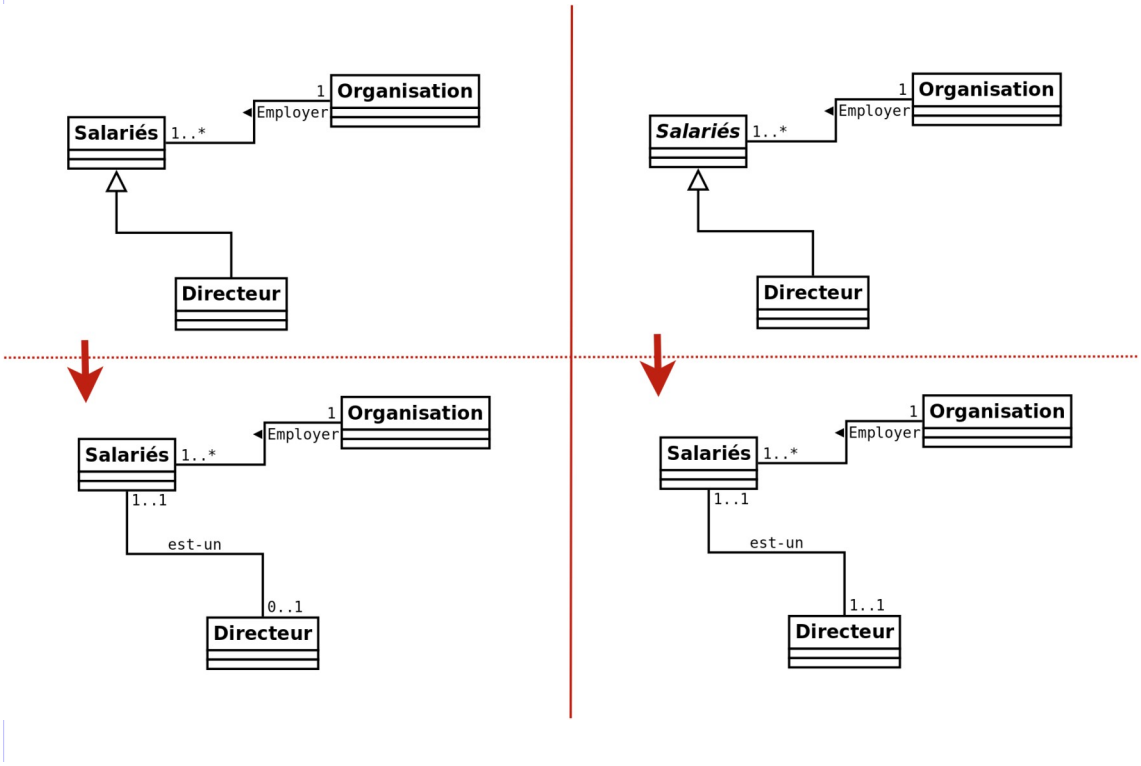
e) Équivalence entre association d'héritage et association 1:1

Remarque

Une association d'héritage est un cas particulier d'association 1:1.

Elle ajoute une sémantique de type "est-un".

Exemple



f) Gestion des défauts

[20 minutes]

Un groupe industriel construisant des moteurs cherche à organiser la gestion des défauts observés sur des moteurs confrontés à des tests en situation réelle. Pour cela un de ses ingénieurs modélise le processus de gestion des défauts, tel qu'il existe actuellement, par le diagramme de classes suivant.

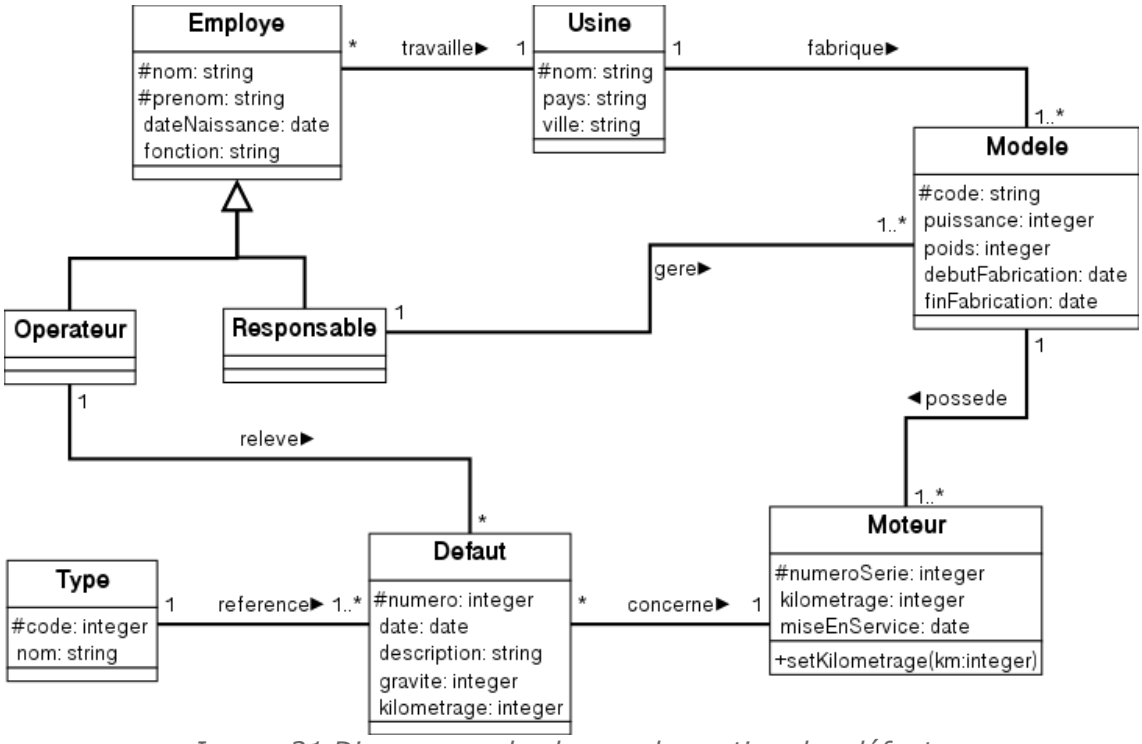


Image 21 Diagramme de classes de gestion des défauts

Question 1

Décrivez en français ce que représente ce diagramme.

Question 2

Étant donné ce modèle, est-il possible de savoir dans quelle usine a été fabriqué un moteur et qui est responsable de sa production ?

Question 3

La responsabilité d'un modèle est elle toujours assumée par un employé travaillant dans l'usine dans laquelle ce modèle est produit ?

Question 4

Pourquoi avoir fait le choix d'une classe **Type** pour codifier les défauts, plutôt qu'un attribut de type énuméré directement dans la classe **Defaut** ?

Question 5

Pourquoi l'attribut `kilométrage` apparaît-il à la fois dans les classes **Defaut** et **Moteur** et pourquoi avoir fait apparaître la méthode `SetKilometrage` ?

Question 6

Ce diagramme permet-il de répondre à la question : Quel est le nombre moyen de défauts rencontrés par les moteurs de chaque modèle ayant été mis en service avant 2000 ? Quelles sont les classes et attributs utiles ?

Question 7

Peut-on également répondre à la question : Quel est le kilométrage moyen des moteurs ayant été concernés par au moins deux défauts d'une gravité supérieure à 5 ?

B. Exercices

1. Armoires secrètes

[30 minutes]

Dans le cadre de la réalisation d'une base de données pour les services secrets français, vous disposez de l'analyse de besoins suivant :

- les agents secrets sont identifiés par un code sur 3 chiffres (comme *007*) et possède un nom et un prénom ;
- les agents secrets produisent des rapports, parfois seul, parfois à plusieurs. Tous les agents secrets ont produit au moins un rapport ;
- les agents secrets sont communément appelés par leurs initiales et leur code : ainsi James Bond 007 est en général appelé *JB007* ;
- un rapport est identifié par un titre (il n'existe pas deux rapports avec le même titre) et il possède une description ainsi que des mots-clés (au moins 2, au plus 10) ;
- le rangement des rapports est organisé comme suit : les rapports sont situés dans des dossiers, qui sont classés dans des casiers, qui sont rangés sur des étagères, dans des armoires ;
- les dossiers, casiers, étagères et armoires sont des rangements, qui sont identifiés par une lettre et un nombre (inférieur à 100). Chaque rangement a une capacité qui détermine le nombre de rangements qu'il peut contenir ;
- en dehors des armoires, il n'existe pas de rapport ou de rangement qui ne serait rangé nulle part.

Question

Proposez un MCD en UML de ce problème. L'on cherchera le modèle le plus expressif possible.

On fera apparaître les types des attributs, en étant le plus précis possible avec les informations dont nous disposons, ainsi que les clés.

2. Capitaine Krik

[45 min]

Le Capitaine Krik a pour tâche de développer une base de données sur les vaisseaux spatiaux et les équipages de la TarFleet.

La flotte ne possède que trois types de vaisseaux : les croiseurs, les frégates et les chasseurs. Chaque membre de la TarFleet a un nom, un prénom, une date de naissance, une planète d'origine et un numéro d'identifiant unique. Certains membres sont aussi soit pilotes, soit capitaines : les pilotes ont un nombre de chasseurs ennemis abattus, tandis que les capitaines ont un certain nombre d'étoiles (entre zéro et cinq).

Krik veut savoir dans la base de données quelles personnes sont affectées à quel vaisseau. Dans la flotte, un chasseur a pour équipage un unique pilote, une frégate a deux pilotes et cinq autres membres d'équipage, tandis qu'un croiseur a un capitaine et de nombreux autres personnels. Les croiseurs peuvent aussi transporter dans leur hangar plusieurs chasseurs (leurs pilotes ne sont alors pas comptés dans l'équipage du croiseur).

Pour chaque vaisseau, on veut connaître son nom et son identifiant, sachant que celui-ci est généré automatiquement à partir du nom et du type de vaisseau (par exemple "EnterpriseCruser"). Pour les frégates et les croiseurs, on veut également

connaître la puissance de leur bouclier (les chasseurs n'en sont pas équipés), et, pour un croiseur, le nombre de membres d'équipage maximal qu'il peut accueillir.

Krik veut aussi des informations sur les réacteurs de chaque vaisseau. Les réacteurs sont soit à fission nucléaire, soit à trou noir miniature. Chacun a un numéro d'emplacement (qui indique où le moteur est monté sur le vaisseau), un poids et une poussée, mais les réacteurs à fission ont également une quantité maximale de carburant, tandis que les réacteurs à trou noir ont une puissance critique.

Question

Proposer un schéma UML permettant de modéliser une telle base de données.

Transformation de l'héritage en relationnel

VII

A. Cours

1. Les trois représentations de l'héritage en relationnel

Objectifs

Savoir faire le passage d'un schéma conceptuel UML à un schéma relationnel.

Connaître les choix possibles pour le cas de l'héritage et savoir faire les bons choix.

La traduction en relationnel des héritages modélisés au niveau conceptuel peut se faire de plusieurs façons, et le choix demande d'étudier plus en détail la nature de cet héritage.

a) Transformation de la relation d'héritage

Le modèle relationnel ne permet pas de représenter directement une relation d'héritage, puisque que seuls les concepts de relation et de clé existent dans le modèle. Il faut donc appauvrir le modèle conceptuel pour qu'il puisse être représenté selon un schéma relationnel.

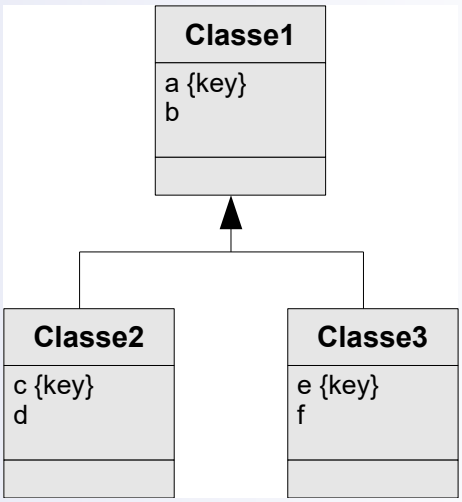
Attention

L'héritage est toujours délicat à traduire en relationnel.

Fondamental

Trois solutions existent pour transformer une relation d'héritage :

- Représenter l'héritage par une référence entre la classe mère et la classe fille.
- Représenter uniquement les classes filles par une relation chacune.
- Représenter uniquement la classe mère par une seule relation.



Graphique 11 Héritage

Méthode

En pratique le choix entre ces trois solutions va dépendre de l'étude détaillée de l'héritage :

- L'héritage est-il complet ?
- La classe mère est-elle abstraite ?
- Quelles sont les associations qui lient la classe mère et les classes filles à d'autres classes ?

Rappel

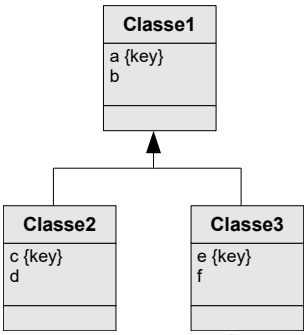
Classe abstraite

Héritage complet

b) Transformation de la relation d'héritage par référence (des filles vers la mère)

Méthode

- Chaque classe, mère ou fille, est représentée par une relation.
- La clé primaire de la classe mère est utilisée pour identifier chacune de ses classes filles : cette clé étant pour chaque classe fille à la fois la clé primaire et une clé étrangère vers la classe mère.



Graphique 12 Héritage

```
Classe1(#a,b)
Classe2(#a=>Classe1,c,d) avec c KEY
Classe3(#a=>Classe1,e,f) avec e KEY
```

Remarque

Si une classe fille a une clé définie dans le modèle conceptuel, cette clé n'est pas retenue pour être la clé primaire dans le modèle relationnel, étant donné que c'est la clé étrangère référence à la classe mère qui est retenue.

Remarque

La cardinalité d'un lien entre une classe fille et une classe mère est (1,1):(0,1) : En effet toute instance fille référence obligatoirement une et une seule instance mère (pas d'héritage multiple) et toute instance mère est référencée une ou zéro fois (zéro fois si un objet peut être directement du type de la classe mère) par chaque instance fille.

Complément

Héritage par une référence et vues

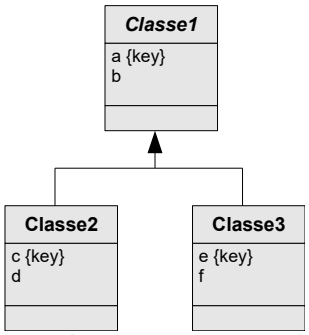
Complément : Méthode alternative

Transformation de la relation d'héritage par référence (de la mère vers les filles)

c) Transformation de la relation d'héritage par les classes filles

Méthode

- Chaque classe fille est représentée par une relation, la classe mère n'est pas représentée (si elle est abstraite).
- Tous les attributs de la classe mère sont répétés au niveau de chaque classe fille.
- La clé primaire de la classe mère est utilisée pour identifier chacune de ses classes filles.



Graphique 13 Héritage (classe mère abstraite)

```
Classe2(#a,b,c,d) avec c KEY
Classe3(#a,b,e,f) avec e KEY
```

Remarque

Si une classe fille a une clé primaire au niveau du MCD, cette clé n'est pas retenue, et c'est bien la clé héritée de la classe mère qui devient la clé primaire (mais elle est bien entendu maintenue comme clé candidate).

Complément : Héritage exclusif

Cette solution est adaptée dans le cas d'un héritage exclusif, c'est à dire si aucun objet d'une classe fille n'appartient aussi à une autre classe fille. Dans le cas contraire, le problème est que des redondances vont être introduites puisqu'un même tuple devra être répété pour chaque classe fille à laquelle il appartient.

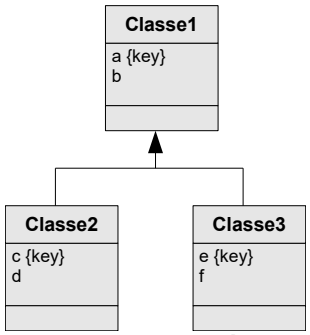
Complément

Héritage par les classes filles et vues

d) Transformation de la relation d'héritage par la classe mère

Méthode

- Seule la classe mère est représentée par une relation (ses classes filles ne sont pas représentées par des relations).
- Tous les attributs de chaque classe fille sont réintégrés au niveau de la classe mère.
- La clé primaire de la classe mère est utilisée pour identifier la relation.
- Un attribut supplémentaire de discrimination `t` (pour "type"), est ajouté à la classe mère, afin de distinguer les tuples. Cet attribut est de type énumération et a pour valeurs possibles les noms de la classe mère ou des différents classes filles.



Graphique 14 Héritage

```
Classe1(#a,b,c,d,e,f,t:{1,2,3}) avec c UNIQUE, e UNIQUE, t NOT NULL
```

Remarque

Si une classe fille a une clé primaire propre, cette clé sera réintégrée à la classe mère, au même titre qu'un autre attribut, mais elle n'officiera pas en tant que clé candidate car elle pourra contenir des valeurs nulles (elle sera néanmoins unique).

Conseil : Héritage complet

Cette solution est optimum dans le cas d'un héritage complet, c'est à dire si les classes filles ne définissent pas d'attributs autre que ceux hérités. Dans le cas contraire cela engendre des valeurs `null`.

En pratique l'héritage complet est rare, mais nombre d'héritages le sont presque et pourront alors être raisonnablement gérés selon cette approche.

Complément : Héritage non exclusif : représentation de la discrimination par des booléens

Si l'héritage concerne un nombre élevé de classes filles et qu'il est principalement non exclusif alors le domaine de l'attribut de discrimination peut impliquer une combinatoire importante de valeurs. Pour contourner cette lourdeur il est possible d'utiliser, en remplacement, un attribut de domaine booléen pour chaque classe fille spécifiant si un tuple est un objet de cette classe.

```
Classe1(#a,b,c,d,e,f,t:{1,2,3})
```

équivaut à :

```
Classe1(#a,b,c,d,e,f,t2:boolean:,t3:boolean)
```

Contraintes :

- `avec t2 X t3` si l'héritage est exclusif
- `avec t2 XOR t3` si l'héritage est exclusif et la classe mère abstraite

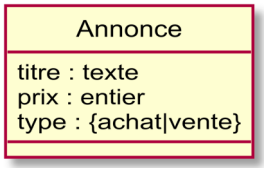
Complément

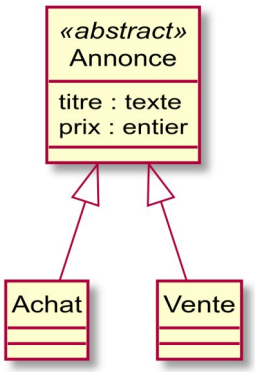
Héritage par la classe mère et vues

e) Exercice

Exercice

Les deux modèles UML suivants sont-ils équivalents au sens où ils conduiront au même modèle relationnel ?

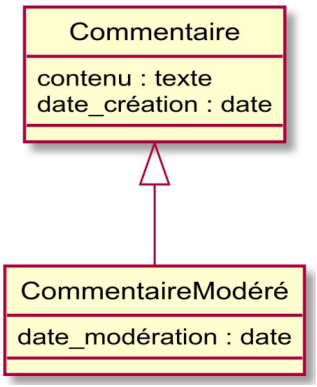




- ○ Les deux modèles sont équivalents
- ○ Les deux modèles ne sont pas équivalents

Exercice

Les deux modèles UML suivants sont-ils équivalents au sens où ils conduiront au même modèle relationnel ?



- ○ Les deux modèles sont équivalents
- ○ Les deux modèles ne sont pas équivalents

2. Choisir la meilleure modélisation de l'héritage en relationnel

a) Choix de la transformation

Les trois solutions de transformation fonctionnent dans tous les cas, mais avec plus ou moins de simplicité (en particulier du point de vue de l'implémentation en SQL). La difficulté consiste pour chaque relation d'héritage à choisir le bon mode de transformation, sachant que chaque solution possède ses avantages et ses inconvénients.

Fondamental

L'objectif est de choisir la transformation qui conduit au modèle relationnel le plus simple à implémenter.

Méthode : Méthode générale

La méthode générale consiste à réaliser les trois transformations pour choisir la plus simple. Avec l'expérience on peut écarter rapidement certaines solutions typiques, mais dès que l'on sort de ces cas, il est conseillé de procéder de façon systématique.

Méthode : Inconvénients typiques de chaque méthode

- Par référence
  - Lourdeur liée à la nécessité de représenter les données des classes filles sur deux relations.
  - Contrainte complexe systématique (dans le cas de l'héritage exclusif)
  - Contrainte complexe lorsque la classe mère est abstraite
- Par les classes filles
  - Contraintes complexes lorsque la classe mère n'est pas abstraite ou qu'elle possède des attributs uniques
  - Contraintes complexes dans le cas de certaines associations avec la classe mère
- Par la classe mère
  - Présence de valeurs nulles systématiques dans la cas de l'héritage non complet
  - Contraintes simples lorsque l'héritage est presque complet (réalisables avec des CHECK en SQL)
  - Contraintes complexes lorsque l'héritage n'est pas complet

Complément

Typologie des contraintes en bases de données

Complément : Penser aux vues !

Il est important de bien penser à ajouter les vues permettant de retrouver le schéma initialement recherché.

Héritage par une référence et vues

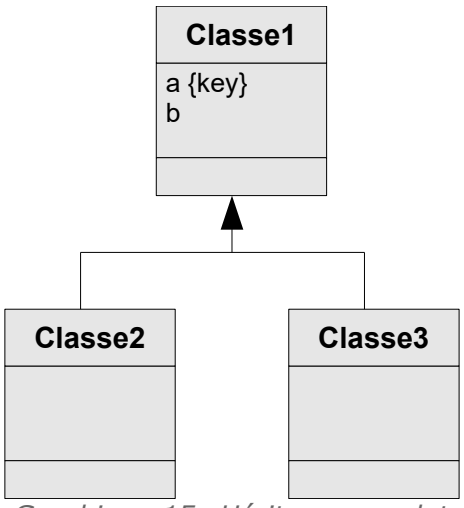
Héritage par les classes filles et vues

Héritage par la classe mère et vues

b) Cas simples de transformation par la classe mère

Méthode : Héritage complet

Dans ce cas, choisir un **héritage par la classe mère**.

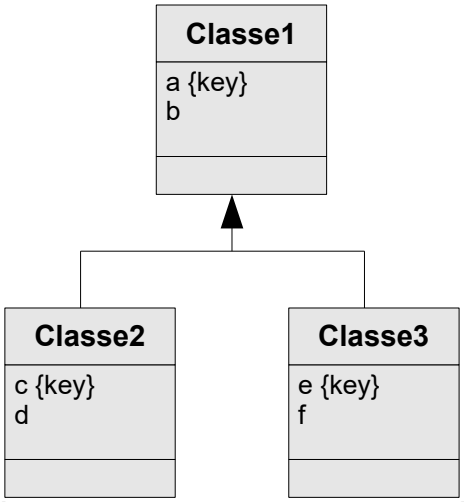


Graphique 15 Héritage complet

`Classe1(#a,b,t:{1,2,3})`

Méthode : Héritage presque complet

L'héritage presque complet peut être géré comme l'héritage complet, **par la classe mère**, surtout si les classes filles ne possèdent pas de clé propre.



Graphique 16 Héritage presque complet (classe mère non abstraite)

`Classe1(#a,b,c,d,e,f,t:{1,2,3}) avec c UNIQUE et e UNIQUE`

Contraintes : on devra vérifier que les nullités et non nullités de c, d, e et f en fonction du type t (cela peut se faire grâce à un CHECK en SQL)

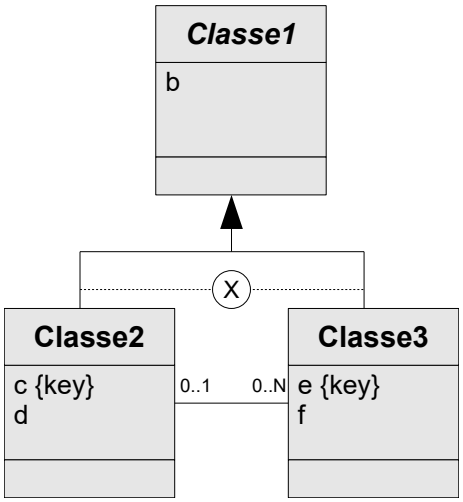
Complément

Contraintes de l'héritage par la classe mère

c) Cas simples de transformation par les classes filles

Méthode : Héritage non complet avec classe mère abstraite (sans association ni clé au niveau de la classe mère)

Dans ce cas, choisir un **héritage par les classes filles**.

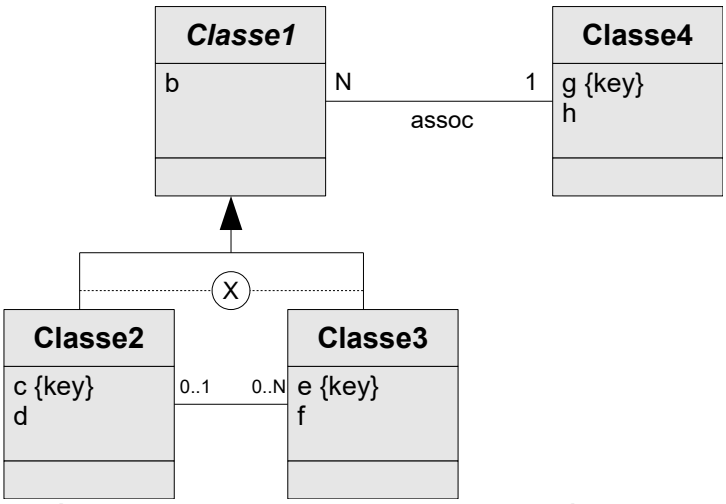


Graphique 17 Héritage exclusif non complet (classe mère abstraite)

```
Classe2(#c,b,d)
Classe3(#e,b,f,fka=>Classe2)
```

Méthode : Héritage non complet avec classe mère abstraite (sans association ni clé au niveau de la classe mère)

Si la classe mère possède une association sortante (c'est elle qui référence, mais elle n'est jamais référencée), alors l'**héritage pas les classes filles** fonctionne toujours très bien.



Graphique 18 Héritage exclusif non complet (classe mère abstraite, association sortante)

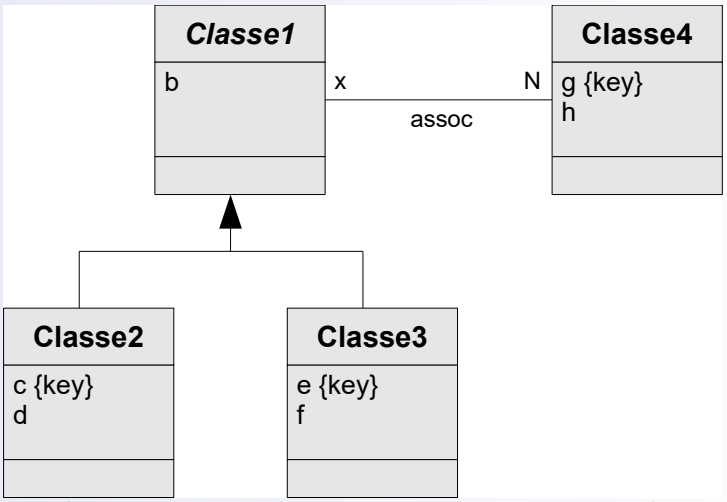
```
Classe2(#c,b,d, fkg=>Classe4)
Classe3(#e,b,f,fka=>Classe2, fkg=>Classe4)
Classe4(#g,h)
```

Méthode

Contraintes de l'héritage par les classes filles

d) Cas problématiques

Attention : Héritage par les classes filles avec association M:N ou 1:N (entrante) sur la classe mère



Héritage avec association x:N sur la classe mère

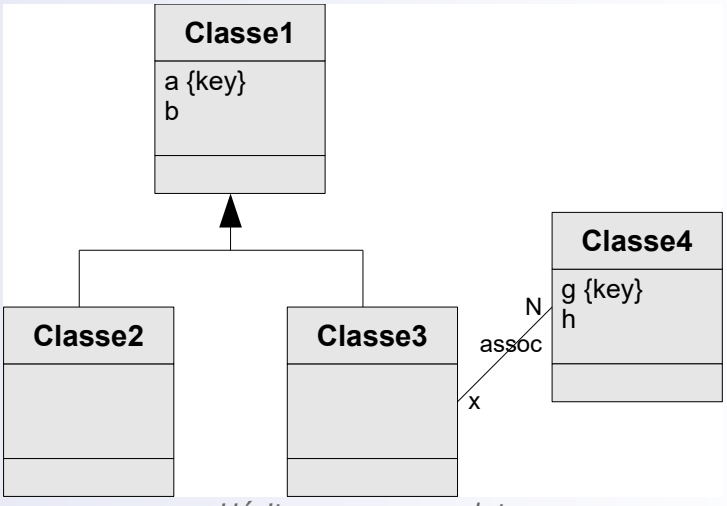
On traite le cas x=1 (1:N), mais le cas M:N ne change rien au problème.

```
Classe2(#c,b,d)
Classe3(#e,b,f)
Classe4(#g,h,fka=>Classe2, fkb=>Classe3)
```

Contrainte : `fka OR fkb`

La solution consiste à ajouter autant de clés étrangères que de classes filles et à gérer le fait que ces clés ne peuvent pas être co-valuées.

Attention : Héritage non complet par la classe mère (association sur une classe fille)



Héritage non complet

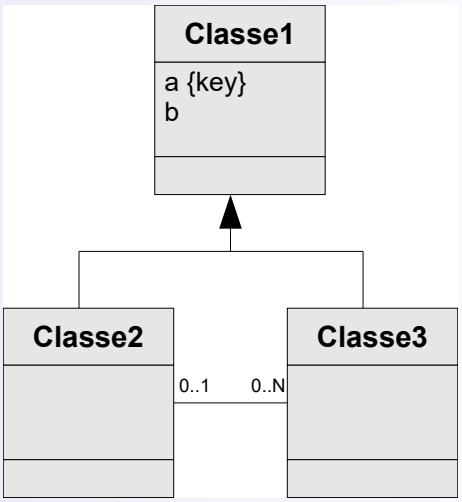
On traite le cas x=1 (1:N).

```
Classe1(#a,b,t:{1,2,3})
Classe4(#g,h,fka=>Classe1)
```

Contraintes : `Classe4.fka ne référence que des enregistrements tels que Classe1.t=3`

On est obligé d'ajouter une contrainte pour limiter la portée de la clé étrangère de Classe4 ; on est sorti ici de ce que l'on sait faire de façon simple en relationnel.

Attention : Héritage non complet par la classe mère (association entre classes filles)



Héritage non complet

`Classe1(#a,b,fka=>Classe1,t:{1,2,3})`

Contraintes : `fka ne référence que des enregistrements tels que t=2 ; si fka alors t=3`

Dans ce cas la solution est encore plus problématique, elle permettra en l'état des associations entre Classe1 et Classe3, et même entre Classe3 et Classe3, on est très loin de la modélisation conceptuelle initiale.

Conseil

Afin de déterminer si un héritage est presque complet ou non, il faut donc surtout regarder les associations, ce sont elles qui poseront le plus de problème un fois en relationnel (à cause de l'intégrité référentielle).

Complément : Héritage non exclusif

L'héritage non exclusif ne doit pas être traité par les classes filles, sous peine d'introduire de la redondance.

Complément : Héritage multiple

L'héritage multiple sera généralement mieux géré avec un héritage par référence.

Complément

Héritage complet

Classes abstraites

Complément : Les cas problématiques obligent à ajouter des contraintes

Contraintes en UML

Liste des contraintes en R

e) Exemple de transformation d'une relation d'héritage

Exemple

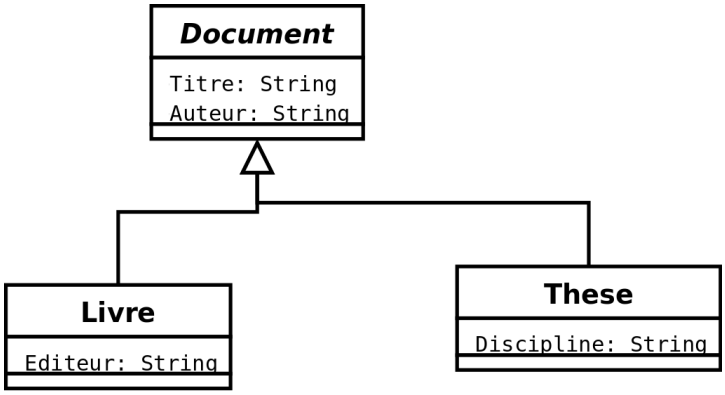


Image 22 Représentation de documents

Remarque : Type d'héritage

1. L'héritage n'est pas complet, puisque l'on observe que les thèses et les livres ont des attributs qui ne sont pas communs (la discipline pour la thèse et l'éditeur pour le livre). Mais l'on peut considérer qu'il est **presque complet**, car il n'y a qu'un attribut par classe fille de différence et pas d'association associée aux classes filles.
2. La classe mère est **abstraite**, on ne gère que des livre ou des thèses, pas d'autres documents ; il n'y a aucune clé candidate identifiée.

La meilleure solution est ici un **héritage par les classes filles** ; un héritage par la classe mère est le second choix (contraintes simples) ; en revanche l'héritage par référence est un mauvais choix (contraintes complexes).

Observons les avantages et inconvénients de chacune des trois solutions.

Héritage absorbé par les classes filles

```
1  These(#id:int, titre:text, discipline:text, auteur:text)
2
3  Livre(#id:int, titre:text, editeur:text, auteur:text)
```

Héritage représenté par une référence

```
1  Document(#id:int, titre:text, auteur:text) avec titre et auteur NOT
   NULL
2
3  These(#id=>Document, discipline:text)
4
5  Livre(#id=>Document, editeur:text)
6
7  Contraites compliquées :
8  - Intersection (PROJECTION(These, id), Projection(Livre, id)) = {}
9  - Projection(Document,id) = Union (Projection(These,id),
   Projection(Livre,id))
10
11 Vues :
12 vThese = JointureNaturelle (Document, These)
13 vLivre = JointureNaturelle (Document, Livre)
```

Héritage absorbé par la classe mère

```
1  Document(#id:int, titre:text, discipline:text, editeur:text,
   auteur:text, type:{These|Livre})
2
3  Contraintes simples :
4  - (discipline NOT NULL AND type=These) OR (editeur NOT NULL AND
   type=Livre)
5  - NOT (discipline NOT NULL AND type=Livre)
6  - NOT (editeur NOT NULL AND type=These)
7
8  Vues :
9  vThese = Projection (Restriction (Document, type=These), id, titre,
   discipline, auteur)
10 vLivre = Projection (Restriction (Document, type=Livre), id, titre,
   editeur, auteur)
```

B. Exercices

1. Lab II+

[45 min]

Description du problème

Un laboratoire souhaite gérer les médicaments qu'il conçoit.

- Un médicament est décrit par un nom, qui permet de l'identifier. En effet il n'existe pas deux médicaments avec le même nom. Un médicament comporte une description courte en français, ainsi qu'une description longue en latin. On gère aussi le conditionnement du médicament, c'est à dire le nombre de pilules par boîte (qui est un nombre entier).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.
- Tout médicament possède au moins un composant, souvent plusieurs. Un composant est identifié par un code unique et possède un intitulé. Tout composant peut intervenir dans la fabrication de plusieurs médicaments. Il existe des composants qui ne sont pas utilisés pour fabriquer des médicaments et que l'on veut quand même gérer.
- Il existe des composants naturels et des composants artificiels. Pour les composants naturels, on gère l'espèce végétale qui porte le composant. Pour les composants artificiels, on gère le nom de la société qui le fabrique.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte « *Médicament contre la chute des choux* » et pour description longue « *Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare.* ». Il est conditionné en boîte de 13.

  Ses contre-indications sont :
  - CI1 : Ne jamais prendre après minuit.

- CI2 : Ne jamais mettre en contact avec de l'eau.

Ses composants sont le **HG79** et le **SN50**.

- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :
  - CI3 : Garder à l'abri de la lumière du soleil

  Son unique composant est le **HG79**.
- Les composants existants sont :
  - **HG79** : "Vif-argent allégé" ; il s'agit d'un composant naturel extrait de l'**edelweiss**.
  - **HG81** : "Vif-argent alourdi" ; il s'agit aussi d'un composant naturel extrait de l'**edelweiss**.
  - **SN50** : "Pur étain" ; il s'agit d'un composant artificiel fabriqué par **Lavoisier et fils SA**.

Question 1

Réaliser le modèle conceptuel de données en UML du problème.

Question 2

En mobilisant les règles adéquates, proposer un modèle logique de données correspondant en relationnel. Le repérage des domaines et des clés est obligatoire.

Question 3

Dessiner des tableaux remplis avec les données fournies en exemple, afin de montrer que le modèle fonctionne selon le besoin exprimé initialement. On pourra mettre le premier mot seulement des descriptions pour gagner du temps.

2. Parc informatique

[30 minutes]

Vous avez en charge la réalisation d'un modèle de base de données pour la gestion d'un parc informatique.

L'analyse des besoins révèlent les informations suivantes : tout matériel informatique est identifié de façon unique par un numéro de série et est décrit par une désignation. Il existe trois types de matériel informatique distincts : les PC, les serveurs et les imprimantes. Pour les PC, les informations que l'on veut gérer sont la taille de la mémoire vive et la cadence du micro-processeur, pour les serveurs on veut gérer leur volume de disque dur et pour les imprimantes leur résolution maximale d'impression.

On veut également gérer les connexions au réseau sachant que tout PC peut être relié à un ou plusieurs serveurs et que chaque serveur sert bien entendu plusieurs PC ; et qu'un PC peut être relié à une imprimante, qui est également utilisée par plusieurs PC.

Question 1

Réaliser le modèle conceptuel UML de ce problème.

Question 2

Réalisez le passage au modèle logique relationnel.

3. Literie

[20 min]

Un revendeur de matelas et de sommier a besoin de créer une base de données, afin de générer son catalogue des produits. Pour cela, il fait appel à vous, afin de concevoir une base de données qui puisse prendre en compte ses besoins. Voici des extraits du document qui retrace les besoins, rédigé par le client :

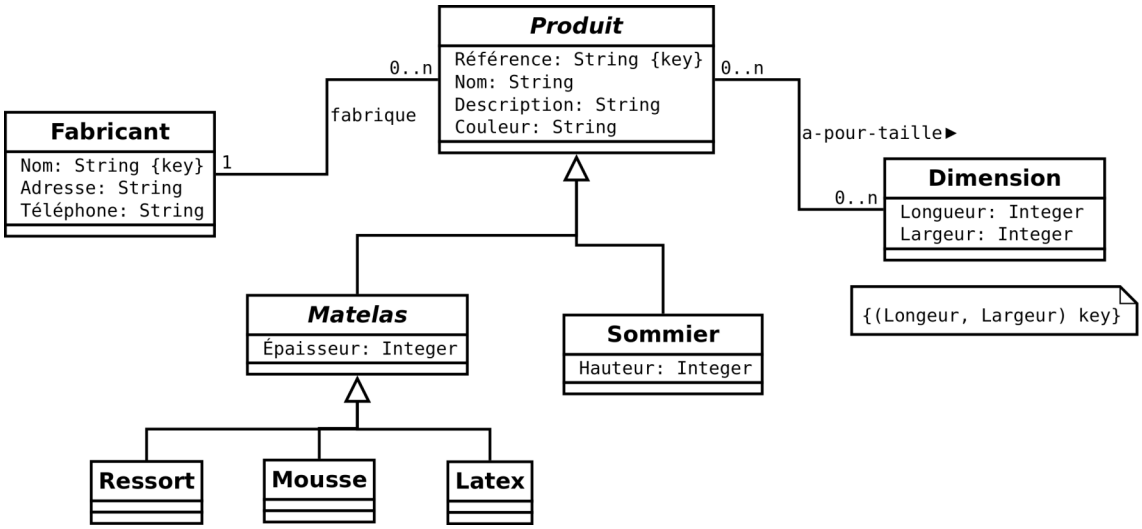
« Le fonctionnement actuel est relativement simple, nous souhaitons juste l'automatiser pour éviter les erreurs et les pertes de temps. Nous avons des types de produits, qui sont des matelas ou des sommiers. Chaque type de produit est caractérisé par des références uniques que nous créons en plus du nom posé par le fabricant (ce nom reprend parfois un plus ancien d'un concurrent) ; nous ajoutons souvent une description textuelle et précisons toujours la couleur. »

« Un type de produit est toujours lié à un fabricant, caractérisé par son nom (sa marque), et nous avons toujours au minimum une adresse et un numéro de téléphone. Un type de matelas a en plus une épaisseur, nécessaire pour l'ajouter dans les catalogues et les publicités. Un matelas est aussi catégorisé en fonction du matériau utilisé : ressort, latex ou mousse. Un type de sommier possède toujours une hauteur. »

« Chaque type de produit est proposé en une ou plusieurs dimensions (longueur et largeur, qui sont souvent des dimensions standardisées). »

Diagramme UML

Un stagiaire a déjà proposé le diagramme UML ci-après.



Literie

Question 1

Vérifiez que le diagramme est correct par rapport à l'énoncé.

Question 2

Proposez un modèle relationnel à partir de l'UML de la question précédente

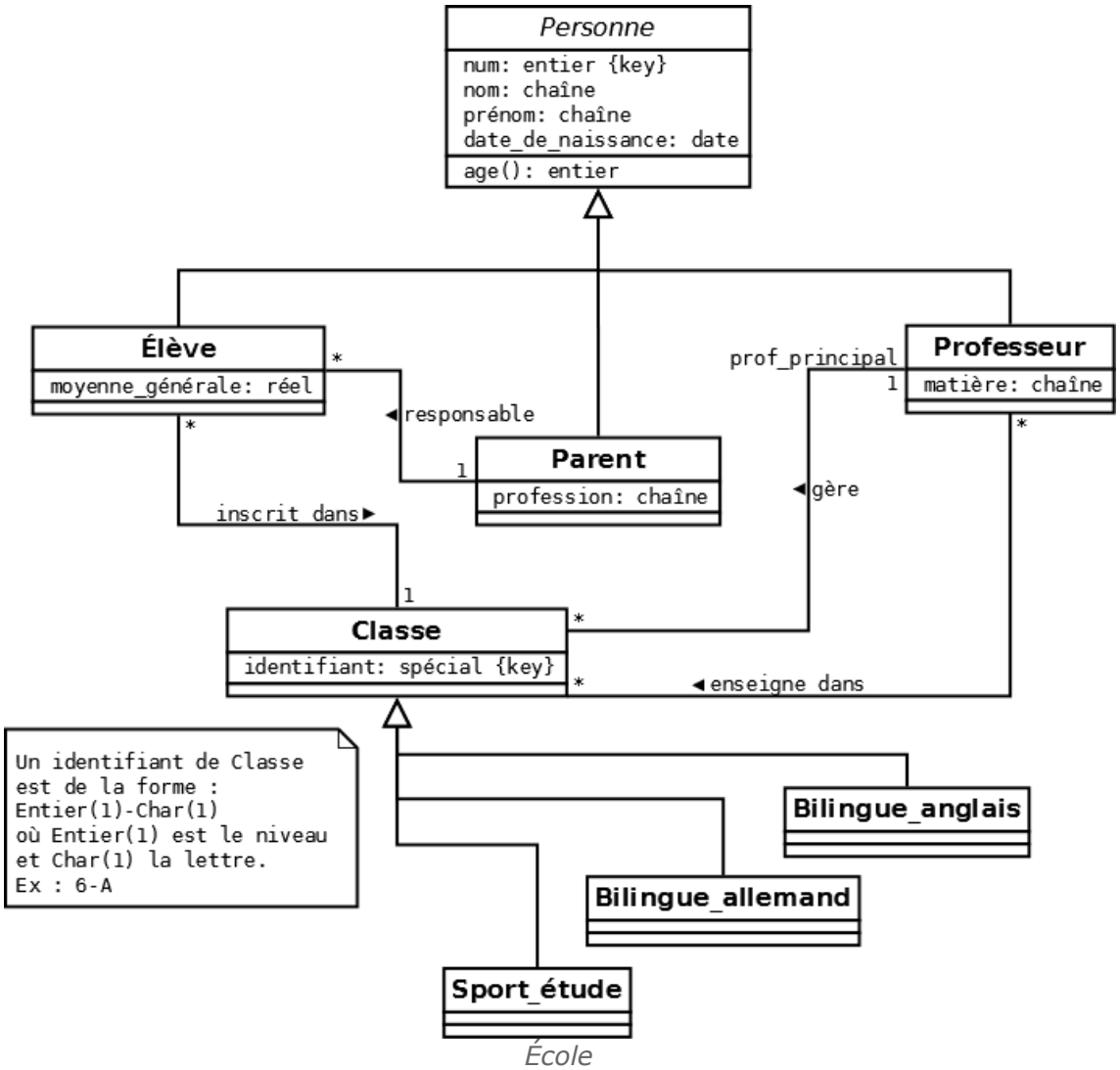
4. À l'école

[20 min]

Une école souhaite se doter d'une base de données pour suivre ses effectifs (élèves et professeurs) ainsi que les classes auxquelles ils sont rattachés. L'analyse des besoins est la suivante :

- Les classes sont identifiées par un niveau et une lettre (6-A, 6-B, 5-A, ...).
- Les classes peuvent avoir un programme adapté suivant un thème unique (sport-étude, bilingue anglais ou bilingue allemand).
- Les classes ont un professeur principal et plusieurs professeurs intervenant dans leur spécialité.
- Les classes accueillent des élèves.
- Les élèves sont sous la responsabilité d'un parent (qui peut aussi être professeur, voire lui même élève).

On a élaboré le diagramme UML ci-après.



École

Question

Proposer un schéma relationnel. Justifier les transformation d'héritage.

Index