



IT 240

# INTRODUCTION AUX BASES DE DONNÉES – UML/SQL

---

AYENA Adébayer

# QUI SUIS — JE ?

[AYENA Adébayor](mailto:ayena.adebayor@gmail.com) ([bayorayena@gmail.com](mailto:ayena.adebayor@gmail.com) / 00228 92 07 79 45)

- ✓ Première année de Licence à la Faculté des Sciences (FDS) de l'Université de Lomé, parcours *Mathématiques* (2013 – 2014) ;
- ✓ Licence Professionnelle au Centre Informatique et de Calcul (CIC) de l'Université de Lomé, Option *Génie Logiciel* (2014 - 2017) ;
- ✓ Oracle Certified Associate, Java SE 8 Programmer (Juin 2019) ;
- ✓ Oracle Certified Professional, Java SE 8 Programmer (Septembre 2019).

# OBJECTIFS DU COURS

- ❑ Ce cours présentera aux étudiants la conception, la mise en œuvre et l'utilisation de bases de données.
- ❑ Les principaux sujets comprennent :
  - la modélisation à l'aide de diagrammes ER ;
  - la création et la maintenance d'une base de données sur PC ;

# OBJECTIFS DU COURS

- La composition et l'utilisation de requêtes dans Structured Query Language ;
- la création et la personnalisation de formulaires et de rapports et l'intégration de bases de données.

# OBJECTIFS DU COURS

□ À la fin de ce cours, les étudiants seront en mesure de :

- Identifier l'utilisation omniprésente des bases de données dans la vie quotidienne ;
- Démontrer les effets de la redondance, des incohérences et du manque d'intégrité lorsque les bases de données sont mal conçues ;

# OBJECTIFS DU COURS

- Appliquer la logique booléenne, la théorie des ensembles et les concepts d'algèbre relationnelle dans la construction de requêtes SQL (Structured Query Language) pour la récupération de données ;
- Prédire et confirmer les résultats en utilisant des preuves empiriques provenant de données accessibles au public ;

# OBJECTIFS DU COURS

- Créer un diagramme de relation d'entité pour modéliser graphiquement les exigences de base de données utilisateur ;
- Construire et maintenir une base de données dans un domaine qui implémente les contraintes du modèle de base de données relationnelle.

# PLAN

Partie 1 : Modélisation Objet avec UML

Partie 2 : Création et maintenance d'une base de données à partir d'une modélisation UML

Partie 3 Utilisation du langage SQL pour administrer une base de données



# NOTIONS FONDAMENTALES EN MATIÈRE DE GESTION DE DONNÉES

## 1. Donnée ou « Data » en anglais

□ Représentation d'un élément d'information, tel qu'un chiffre ou un fait, codé dans un format permettant son stockage et son traitement par ordinateur (Data).

# NOTIONS FONDAMENTALES EN MATIÈRE DE GESTION DE DONNÉES

## **2. Type de donnée ou « Data type » en anglais**

- ☐ Nature du codage utilisé pour représenter une donnée élémentaire et les opérations applicables à cette donnée.
- ☐ Les types les plus courants sont: entier, réel, texte, date, image, etc.

# NOTIONS FONDAMENTALES EN MATIÈRE DE GESTION DE DONNÉES

## 3. Information ou « Information » en anglais

□ Une information est une donnée ou un ensemble de données qui a ou ont été interprétée(s).

# NOTIONS FONDAMENTALES EN MATIÈRE DE GESTION DE DONNÉES

## 4. Bases de données (BD) ou « Database » en anglais (DB)

- ❑ Une base de données est un *ensemble d'informations* qui est organisé de manière à être facilement accessible, géré et mis à jour.
- ❑ Elle est utilisée par les organisations comme méthode de stockage, de gestion et de récupération de l'informations.

# NOTIONS FONDAMENTALES EN MATIÈRE DE GESTION DE DONNÉES

## **5. Système de Gestion de Bases de Données (SGBD) ou « Database Management System » en anglais (DBMS)**

- ❑ Un système de gestion de base de données (SGBD) est un logiciel système conçu pour créer et gérer des bases de données.
- ❑ Les SGBD sont les logiciels intermédiaires entre les utilisateurs et les bases de données.

# NOTIONS FONDAMENTALES EN MATIÈRE DE GESTION DE DONNÉES

## 6. Entrepôt de données ou « Data warehouse » en anglais

❑ Base de données spécialisée dans laquelle est centralisé un volume important de données consolidées à partir des différentes sources de renseignement d'une entreprise (notamment les bases de données internes) et qui est conçue de manière à ce que les personnes intéressées aient accès rapidement à l'information stratégique dont elles ont besoin.

# NOTIONS FONDAMENTALES EN MATIÈRE DE GESTION DE DONNÉES

## **7. Système d'information (SI) ou « Management information system » en anglais**

□ Système constitué des ressources humaines, des ressources informatiques (équipement, logiciel, données) et des procédures permettant d'acquérir, de stocker, de traiter et de diffuser les éléments d'information pertinents au fonctionnement d'une entreprise ou d'une organisation.

# LES FONCTIONS D'UN SYSTÈME D'INFORMATION

□ Le SI possède 4 fonctions essentielles :

- La **saisie** ou **collecte** de l'information ;
- La **mémorisation** de l'information à l'aide de fichier ou de base de données ;
- Le **traitement** de l'information afin de mieux l'exploiter (consultation, organisation, mise à jour, calculs pour obtenir de nouvelles données, ...) ;
- La **diffusion** de l'information.



# EXEMPLES D'UTILISATION D'UNE BASE DE DONNEES

## Exemple 1 :

□ Une institution universitaire pourrait par exemple exploiter une seule base de données permettant de gérer l'admission des candidats, d'assurer l'offre de cours à chaque session, d'inscrire les étudiants, de percevoir les frais d'inscription, de compiler les résultats et d'émettre les bulletins de notes.

# EXEMPLES D'UTILISATION D'UNE BASE DE DONNEES

## Exemple 2 :

❑ Un supermarché pourrait utiliser une base de données pour gérer ses produits en stocks, enregistrer les clients, enregistrer les achats, générer des factures aux clients. Utiliser cette même base de données pour éditer l'état des ventes effectuées au cours d'une journée, d'un mois, d'un trimestre, d'un semestre, d'une année, etc. Afficher le produit le plus acheté dans un mois, etc.

# POURQUOI UTILISER UNE BASE DE DONNEES ?

- ❑ La capacité de stocker, de récupérer et de trier l'information peut rendre les entreprises plus efficaces, les aidant ainsi à être concurrentiel sur le marché.
- ❑ Même les plus petites entreprises peuvent créer des bases de données qui contribuent à la croissance de l'entreprise, car les gestionnaires utilisent les données pour prendre des décisions.
- ❑ Les gestionnaires peuvent également utiliser les données pour repérer les tendances et planifier l'avenir.

# 7 BONNES RAISONS D'UTILISER UNE BASE DE DONNÉES

## 1. Un système centralisé

- ❑ Si votre entreprise est en pleine croissance, il peut être difficile de suivre la quantité croissante de données.
- ❑ De bons systèmes de bases de données peuvent vous aider à gérer toutes les données critiques de votre entreprise de manière centralisée, sûre et sécurisée pour augmenter vos chances de succès.

# 7 BONNES RAISONS D'UTILISER UNE BASE DE DONNÉES

## 2. Une meilleure organisation de travail

- ❑ Plus vous avez de clients, plus il vous sera difficile de prendre des notes manuscrites et de les conserver dans un tableur.
- ❑ Ce problème est facile à résoudre avec une bonne base de données.
- ❑ Presque tous les logiciels de base de données vous permettent d'entrer des notes dans le dossier en ligne d'un client.

# 7 BONNES RAISONS D'UTILISER UNE BASE DE DONNÉES

## 2. Une meilleure organisation de travail

- ❑ Vous pourrez également les retrouver rapidement grâce à une recherche rapide dans la base de données.

# 7 BONNES RAISONS D'UTILISER UNE BASE DE DONNÉES

## 3. Récupération rapide des informations

- ❑ Votre capacité à trouver des données sur les transactions, les impôts, les dépenses et les actifs peut vous aider à prendre des décisions vitales pour votre entreprise.
- ❑ Par exemple, vous pouvez savoir combien vous dépensez en fournitures, en salaires, en énergie et en frais d'expédition, pour ne citer que quelques exemples.

# 7 BONNES RAISONS D'UTILISER UNE BASE DE DONNÉES

## 4. Suivi de la clientèle

- ❑ Votre base de données peut vous aider à garder vos clients en vous permettant d'offrir un meilleur service à la clientèle.
- ❑ Vous pouvez utiliser la base de données pour savoir quels sont les clients qui vous ont le plus acheté, ceux qui achètent le plus souvent et ceux qui ne vous ont pas acheté récemment.



# 7 BONNES RAISONS D'UTILISER UNE BASE DE DONNÉES

## 4. Suivi de la clientèle

❑ Loin d'être impersonnelle, une base de données donne les informations dont vous avez besoin pour maintenir un contact personnel avec des clients importants.

# 7 BONNES RAISONS D'UTILISER UNE BASE DE DONNÉES

## 5. Meilleure gestion des ressources humaines

- ❑ L'utilisation d'une base de données RH pour gérer les dossiers du personnel peut faire économiser du temps et de l'argent.
- ❑ Il peut rationaliser la plupart des tâches RH, automatiser les tâches routinières et accélérer le traitement des données telles que les heures de travail du personnel, les congés, les avantages sociaux, la paie, etc.

# 7 BONNES RAISONS D'UTILISER UNE BASE DE DONNÉES

## 5. Meilleure gestion des ressources humaines

- ❑ Cela peut laisser plus de temps pour se concentrer sur la croissance de votre entreprise.

# 7 BONNES RAISONS D'UTILISER UNE BASE DE DONNÉES

## 6. Suivi efficace des stocks

- ❑ Bien gérer l'inventaire peut parfois sembler un exercice d'équilibriste. Il est facile d'en avoir trop et faire face au gaspillage, ou trop peu et décevoir vos clients et de nuire à votre réputation.
- ❑ Si vous suivez votre inventaire manuellement, il est également facile de faire des erreurs de saisie de données ou d'égarer les feuilles de calcul et les notes.

# 7 BONNES RAISONS D'UTILISER UNE BASE DE DONNÉES

## 6. Suivi efficace des stocks

❑ En utilisant une base de données de suivi des stocks, vous pouvez éviter ces risques et minimiser les pertes de ventes tout en maximisant vos opportunités de croissance.

# 7 BONNES RAISONS D'UTILISER UNE BASE DE DONNÉES

## 7. Planification de la croissance

- ❑ La plupart des bases de données d'entreprise disposent d'une forme ou d'une autre de capacités de reporting, d'analyse des données d'entrée et du suivi de la productivité à l'anticipation des tendances futures et des besoins des clients.
- ❑ Si vous planifiez une stratégie de croissance, un système de base de données robuste peut être la ressource la plus précieuse de votre entreprise.

# **Partie 1 : MODÉLISATION OBJET AVEC UML**

## Chap. 1 : PRÉSENTATION DE UML

- ❑ « *Une image vaut mieux qu'un long discours* » ; ce proverbe résume l'origine de la création de la schématisation en langage de modélisation unifié (UML).
- ❑ Son objectif : créer un langage visuel commun dans le monde complexe du développement de logiciels, un langage qui serait également compris par les utilisateurs professionnels et tous ceux qui veulent comprendre un système



## Chap. 1 : PRÉSENTATION DE UML

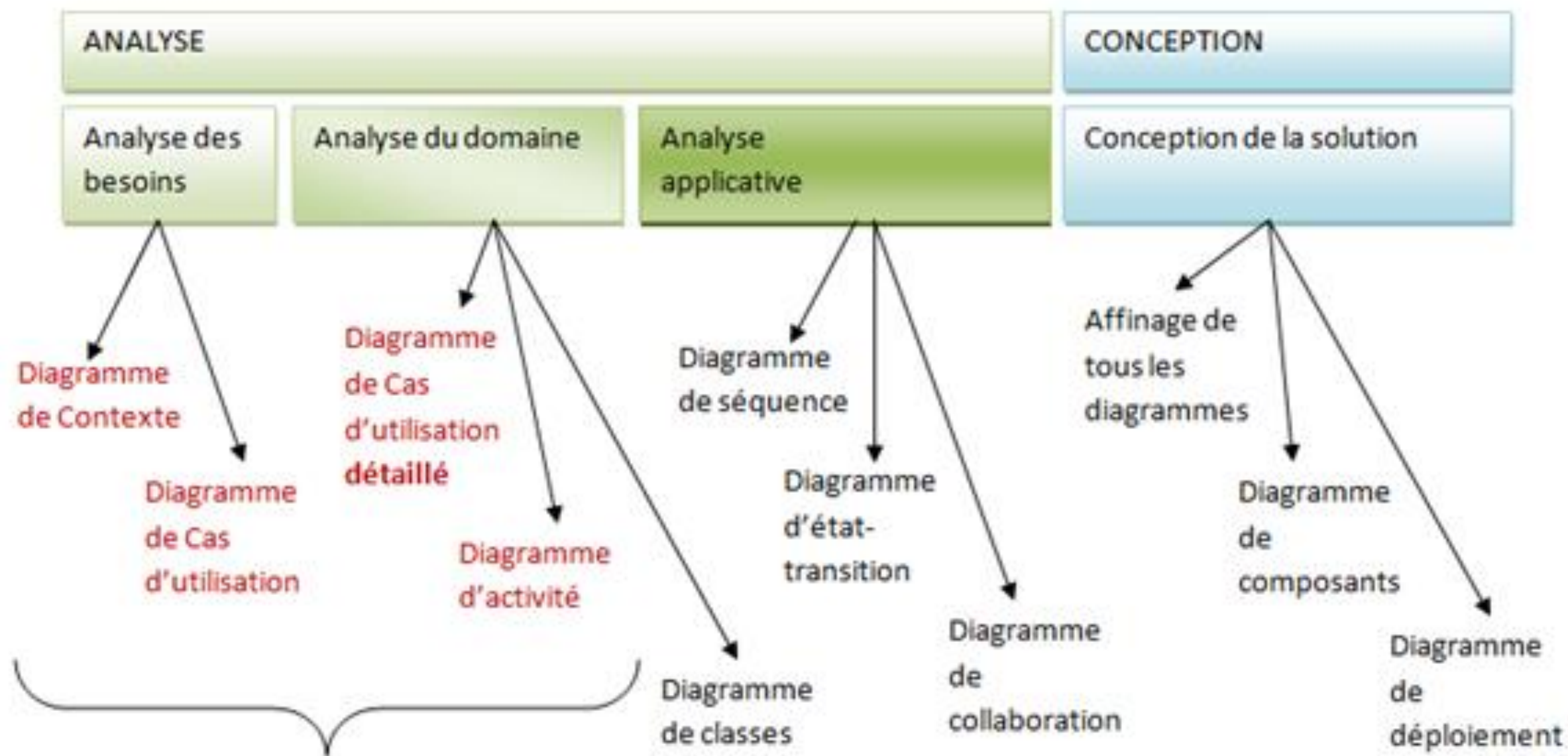
- ❑ Comme n'importe quel type de projet, un projet informatique nécessite une ***phase d'analyse***, suivi d'une ***phase de conception***.
- ❑ Dans la ***phase d'analyse***, on cherche d'abord à bien comprendre et à décrire de façon précise les besoins des utilisateurs ou des clients.
  - Que souhaitent-ils faire avec le logiciel ?
  - Quels sont les personnes qui vont interagir avec le logiciel (les acteurs) ?

# Chap. 1 : PRÉSENTATION DE UML

- Quelles fonctionnalités veulent-ils ?
  - Pour quel usage ?
  - Comment l'action devrait-elle fonctionner ?
- C'est ce qu'on appelle « *l'analyse des besoins* ».
- Après validation de notre compréhension du besoin, nous imaginons ensuite la solution. C'est la partie « *analyse de la solution* ».

## Chap. 1 : PRÉSENTATION DE UML

□ Dans la ***phase de conception***, on apporte plus de détails à la solution et on cherche à clarifier des aspects techniques, tels que l'installation des différentes parties logicielles à installer sur du matériel.



Partie couverte par ce premier cours.

## Chap. 1 : PRÉSENTATION DE UML

- ❑ Pour réaliser ces deux phases dans un projet informatique, nous utilisons des méthodes, des conventions et des notations. **UML** fait partie des notations les plus utilisées aujourd'hui.
- ❑ Nous allons dans cette partie du cours, définir le langage UML et ses outils : les diagrammes.
- ❑ Nous verrons ensuite comment ce langage peut contribuer à la phase d'analyse des besoins et du domaine d'un projet informatique.

## 1.1. DÉFINITION DE UML

- ❑ **UML**, c'est l'acronyme anglais pour « *Unified Modeling Language* ».
- ❑ On le traduit par « *Langage de modélisation unifié* ».
- ❑ La notation **UML** est un langage visuel constitué d'un ensemble de *schémas*, appelés des *diagrammes*, qui donnent chacun une vision différente du projet à traiter.

## 1.1. DÉFINITION DE UML

- ❑ UML nous fournit donc des diagrammes pour représenter le logiciel à développer : *son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel*, etc.
- ❑ Réaliser ces diagrammes revient donc à *modéliser les besoins du logiciel* à développer.

## 1.1. DÉFINITION DE UML

- ❑ Le langage UML ne préconise aucune démarche, ce n'est donc pas une méthode.
- ❑ Chacun est libre d'utiliser les types de diagramme qu'il souhaite, dans l'ordre qu'il veut.
- ❑ Il suffit que les diagrammes réalisés soient cohérents entre eux, avant de passer à la réalisation du logiciel.



## 1.2. POURQUOI MODÉLISER ?

- ❑ Modéliser, c'est décrire de manière **visuelle** et **graphique** les besoins et, les solutions fonctionnelles et techniques de votre projet logiciel.
- ❑ UML nous aide à faire une **description de façon graphique** et devient alors un excellent moyen pour « **visualiser** » le(s) futur(s) logiciel(s).
- ❑ Un logiciel qui a été réalisé sans analyse et sans conception (étapes où l'on modélise le futur logiciel) risque lui aussi de ne pas répondre aux besoins, de comporter des anomalies et d'être très difficile à maintenir.

## 1.2. POURQUOI MODÉLISER ?

- ❑ Tout comme la construction d'une maison nécessite des plans à différents niveaux (vision extérieure, plan des différents étages, plans techniques...), la réalisation d'un logiciel ou d'un ensemble de logiciels a besoin d'un certain nombre de diagrammes.
- ❑ Le temps que prend la modélisation ne devrait pas être un frein. Cette étape nous permet de ne pas perdre davantage de temps ultérieurement.

## 1.3. LES DIAGRAMMES UML

- ❑ Les diagrammes sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie.
- ❑ Il en existe quatorze depuis UML 2.3 qu'on peut classer en 3 catégories :
  - Diagrammes de structure ou diagrammes statiques ;
  - Diagrammes de comportement ;
  - Diagrammes d'interaction ou diagrammes dynamiques.

## 1.3. LES DIAGRAMMES UML

### 1. Diagrammes de structure ou diagrammes statiques

Les diagrammes de structure (structure diagrams) ou diagrammes statiques (static diagrams) rassemblent :

1. **Diagramme de classes (class diagram)** : représentation des classes intervenant dans le système ;

## 1.3. LES DIAGRAMMES UML

**2. Diagramme d'objets (*object diagram*)** : représentation des instances de classes (objets) utilisées dans le système ;

**3. Diagramme de composants (*component diagram*)** : représentation des composants du système d'un point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données...) ;

## 1.3. LES DIAGRAMMES UML

**4. *Diagramme de déploiement (deployment diagram)*** : représentation des éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage...) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux ;

## 1.3. LES DIAGRAMMES UML

**5. *Diagramme des paquets (package diagram)*** : représentation des dépendances entre les paquets (un paquet étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML), c'est-à-dire entre les ensembles de définitions ;

## 1.3. LES DIAGRAMMES UML

**6. Diagramme de structure composite (composite structure diagram) :** représentation sous forme de boîte blanche les relations entre composants d'une classe (depuis UML 2.x) ;

**7. Diagramme de profils (profile diagram) :** spécialisation et personnalisation pour un domaine particulier d'un meta-modèle de référence d'UML (depuis UML 2.2).



## 1.3. LES DIAGRAMMES UML

### II. Diagrammes de comportement

Les diagrammes de comportement (behavior diagrams) rassemblent :

**8. *Diagramme des cas d'utilisation (use-case diagram)*** : représentation des possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire de toutes les fonctionnalités que doit fournir le système ;

## 1.3. LES DIAGRAMMES UML

**9. Diagramme états-transitions (*state machine diagram*)** : représentation sous forme de machine à états finis le comportement du système ou de ses composants ;

**10. Diagramme d'activité (*activity diagram*)** : représentation sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants.

## 1.3. LES DIAGRAMMES UML

### **III. Diagrammes d'interaction ou diagrammes dynamiques**

Les diagrammes d'interaction (interaction diagrams) ou diagrammes dynamiques (dynamic diagrams) rassemblent :

**11. Le diagramme de séquence (sequence diagram) :** représentation de façon séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs ;

## 1.3. LES DIAGRAMMES UML

**12.** Le diagramme de communication (communication diagram) : représentation de façon simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets (depuis UML 2.x) ;

## 1.3. LES DIAGRAMMES UML

**13. Le diagramme global d'interaction (*interaction overview diagram*) :** représentation des enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences (variante du diagramme d'activité) (depuis UML 2.x) ;

**14. Le diagramme de temps (*timing diagram*) :** représentation des variations d'une donnée au cours du temps (depuis UML 2.3).

## 1.3. LES DIAGRAMMES UML

□ Les 4+1 vues d'un système (Recherches...).

## Chap. 2 : QUELLE DÉMARCHE SUIVRE ?

- ❑ Je vous rappelle qu'UML ne préconise aucune démarche spécifique.
- ❑ Si vous voulez commencer votre phase d'analyse en réalisant un **diagramme de classes**, libre à vous.
- ❑ Toutefois, il y a un minimum d'analyse du besoin à faire avant de se ruer sur le diagramme de classes.

## Chap. 2 : QUELLE DÉMARCHE SUIVRE ?

□ Nous allons voir dans ce chapitre, les principales étapes de développement d'un logiciel et quelques démarches à suivre.



## 2.1. LES ÉTAPES DE DÉVELOPPEMENT LOGICIEL

□ Le processus de développement logiciel contient un certain nombre d'étapes :

1. Définition des besoins et des exigences du client et des utilisateurs ;
2. Analyse du système ;
3. Conception du système ;
4. Programmation du logiciel ;

## 2.1. LES ÉTAPES DE DÉVELOPPEMENT LOGICIEL

5. Test du logiciel ;

6. Déploiement ;

7. Maintenance du système.



Les étapes d'un projet

## 2.2. DESCRIPTION DE CHAQUE ÉTAPE

### 1. Définition des besoins et des exigences du client et des utilisateurs

□ La définition des besoins et des exigences correspond à l'étape dans laquelle nous discutons avec le client et les futurs utilisateurs afin de comprendre de quoi ils ont besoin : ***QUI doit pouvoir faire QUOI ?***

□ Lors de cette étape, nous définissons également les demandes précises, telles que le respect de certaines normes graphiques, les temps de réponse, le matériel sur lesquels le logiciel devrait fonctionner, etc.

## 2.2. DESCRIPTION DE CHAQUE ÉTAPE

### 2. Analyse du système

- ❑ L'analyse du système permet d'affiner ce qui a été défini dans l'étape précédente.
- ❑ On y détaille davantage le fonctionnement interne du futur logiciel (**COMMENT cela doit-il fonctionner ?**).

## 2.2. DESCRIPTION DE CHAQUE ÉTAPE

### 3. Conception du système

- ❑ La conception du système correspond à la définition de choix techniques.

## 2.2. DESCRIPTION DE CHAQUE ÉTAPE

### 4. Programmation du logiciel

- ❑ La programmation est l'étape dans laquelle les informaticiens se donnent à cœur joie !
- ❑ Ils réalisent le logiciel à l'aide de langages de programmation, de systèmes de gestion de bases de données, etc.

## 2.2. DESCRIPTION DE CHAQUE ÉTAPE

### 5. Test du logiciel

- ❑ Durant les tests, les informaticiens vérifient que le logiciel fonctionne et répond aux besoins définis en début du projet.
- ❑ Cette phase de tests peut intégrer des validations du logiciel avec le client et/ou les utilisateurs. C'est même plus que souhaité.

## 2.2. DESCRIPTION DE CHAQUE ÉTAPE

### 6. Déploiement

- Lors du déploiement, les informaticiens installent le logiciel sur le matériel et réalisent des ajustements pour faire fonctionner le logiciel dans l'environnement de travail des utilisateurs.



## 2.2. DESCRIPTION DE CHAQUE ÉTAPE

### 7. Maintenance du système

- La maintenance correspond à la période qui suit l'installation et pendant laquelle les anomalies et problèmes doivent être corrigés.

## 2.3. LA DÉMARCHE

## 2.3.1. LES ÉTAPES DE L'ANALYSE DES BESOINS DU CLIENT

☐ **Par quoi on commence ?**

☐ Comme dit précédemment, il ne sert à rien de commencer à coder sans avoir compris les besoins des utilisateurs au préalable.

☐ Sinon, on risque de créer un logiciel qui n'est pas utile, qui est bancal ou en tout cas non satisfaisant pour les utilisateurs.

## 2.3.1. LES ÉTAPES DE L'ANALYSE DES BESOINS DU CLIENT

- ❑ Analyser les besoins, c'est découvrir des éléments de plus en plus précis.
- ❑ Voici les étapes :

## 2.3.1. LES ÉTAPES DE L'ANALYSE DES BESOINS DU CLIENT

### Étape 1

- ❑ On commence par décrire le contexte du logiciel à créer.
- ❑ Qu'est-ce que c'est le contexte ? Eh bien, c'est l'environnement direct du logiciel. Il s'agit là de décrire **QUI** devra utiliser le logiciel.
- ❑ Il faut donc se poser la question : **à qui le logiciel devra servir ?**

## 2.3.1. LES ÉTAPES DE L'ANALYSE DES BESOINS DU CLIENT

### Étape 2

- ❑ On décompose ensuite le logiciel en plusieurs parties distinctes, histoire de ne pas avoir à analyser quelque chose de trop énorme d'un coup.
- ❑ On appelle ça la **décomposition en packages**.
- ❑ La décomposition peut se faire en réfléchissant à des « **familles** » de **fonctionnalités** qui seraient nécessaires.

## 2.3.1. LES ÉTAPES DE L'ANALYSE DES BESOINS DU CLIENT

### Étape 2

□ Les principales questions à se poser sont les suivantes :

- Quelles sont les grandes parties qui doivent composer le logiciel ?
- Pour une partie précise, qui, parmi les acteurs identifiés (ou utilisateurs) l'utilisera ?

## 2.3.1. LES ÉTAPES DE L'ANALYSE DES BESOINS DU CLIENT

### Étape 3

- ❑ Chaque partie est alors analysée séparément, en précisant **QUI** devra pouvoir faire **QUOI** grâce à cette partie du logiciel.
- ❑ C'est ce qu'on appelle définir *les cas d'utilisation*.
- ❑ Par exemple, demandez-vous, « *dans la partie X, qui devra faire quoi avec le logiciel ?* ».



## 2.3.2. LES OUTILS NÉCESSAIRES

□ Pour illustrer chacune des étapes citées ci-dessus, on utilise un diagramme précis, que nous détaillerons au fur et à mesure dans cette partie du cours :

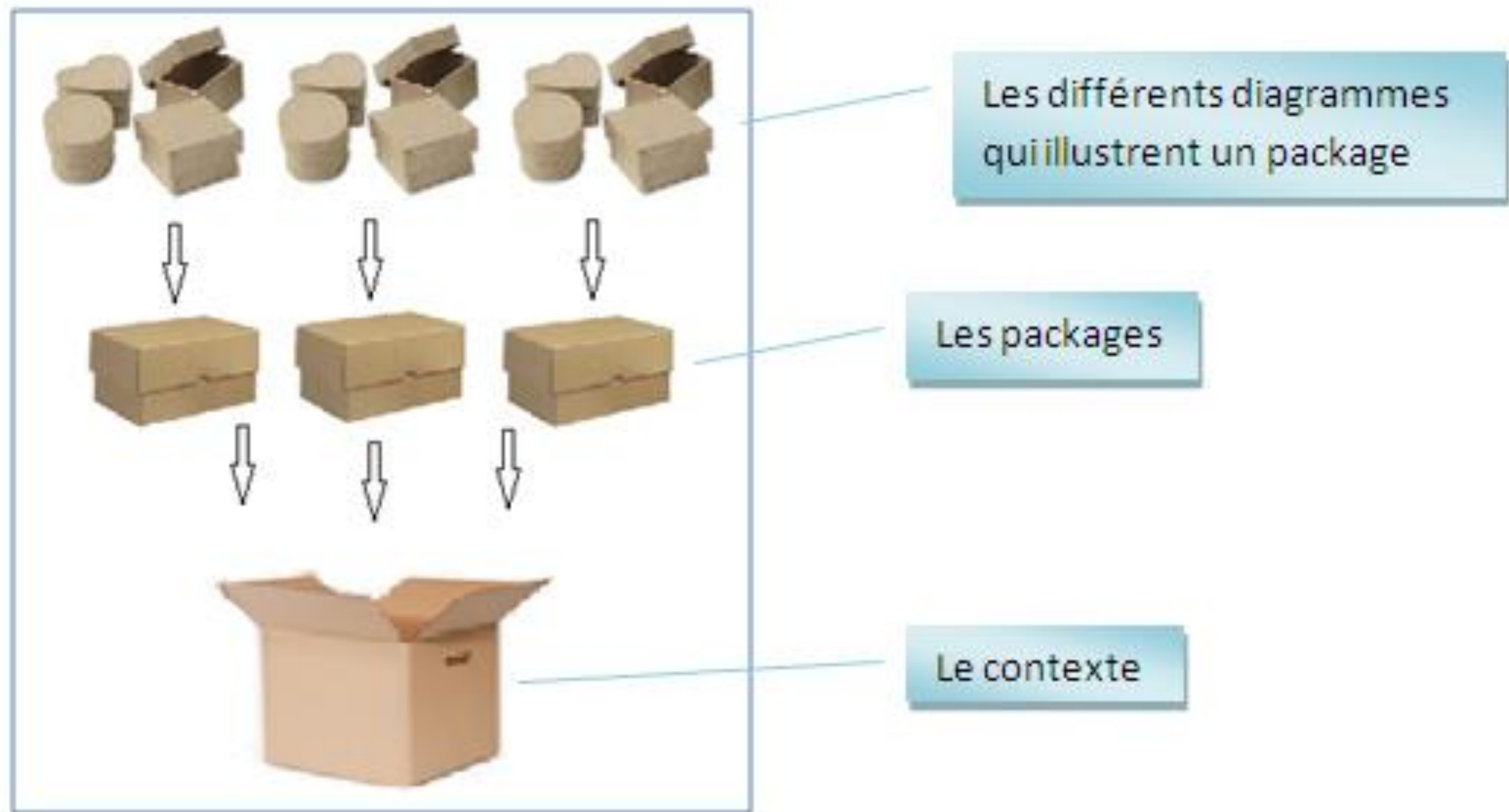
- Pour décrire le contexte, on réalise ***un diagramme de contexte*** dans lequel on indiquera qui aura une utilité du futur logiciel.
- Pour expliquer la décomposition du logiciel en parties distinctes, on se sert ***d'un diagramme de packages***. Celui-ci nous permettra d'indiquer qui aura besoin de chacune des parties.

## 2.3.2. LES OUTILS NÉCESSAIRES

- Enfin, pour illustrer ce que le logiciel doit permettre de faire, on utilise ***un diagramme des cas d'utilisation.***

- En fait, chaque diagramme est une précision du précédent. C'est comme si en ouvrant une grande boîte en carton, on découvrirait d'autres boîtes.

- D'ailleurs, notre découverte ne s'arrêtera pas au diagramme des cas d'utilisation, mais là j'avance un peu sur les parties à venir...



Décomposition de l'analyse

## 2.3.2. LES OUTILS NÉCESSAIRES

□ Dans la suite de ce cours, les diagrammes que nous réaliseront sont les suivants :

- Diagramme de contexte ;
- Diagramme de package ;
- Diagramme de cas d'utilisation (suivi de la description textuelle des différents d'utilisation) ;

## 2.3.2. LES OUTILS NÉCESSAIRES

- Diagramme d'activités ;
- Diagramme de séquences ;
- Diagramme de classes.

## Chap. 3 : DIAGRAMME DE CONTEXTE

- ❑ Au départ, le logiciel peut nous sembler un peu « nébuleux ».
- ❑ On a souvent l'impression de ne pas savoir par quel bout commencer.
- ❑ Rassurez-vous, cela est tout à fait normal.
- ❑ Après avoir parcouru tous ces diagrammes, vous serez en mesure de savoir par où débiter avec vos projets.

## Chap. 3 : DIAGRAMME DE CONTEXTE

- ❑ Le diagramme de contexte sert à délimiter le contour d'un système, à définir clairement ses frontières et les acteurs avec lesquels il communique.

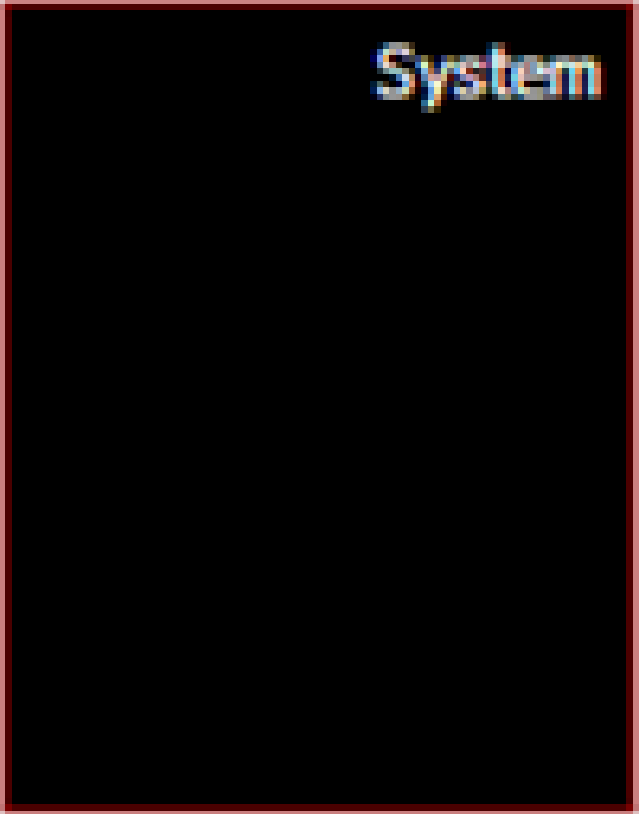
## 3.1. LA BOÎTE NOIRE

- ❑ On commencera d'ailleurs par considérer que le futur logiciel correspond à une boîte noire qui doit fournir des services à son environnement.
- ❑ Par environnement, on entend les utilisateurs qui ont besoin de ce logiciel.
- ❑ Dans UML, on appelle ce qu'on doit analyser, concevoir et réaliser : ***le système.***



## 3.1. LA BOÎTE NOIRE

❑ Ici, le système est donc par exemple une ***application de gestion d'une boutique*** ou une ***application de gestion de stocks...***



System

---

Le système, style « boîte noire »

## 3.1. LA BOÎTE NOIRE

- ❑ Cette « **boîte noire** » sera donc utile à un ou plusieurs utilisateurs.
- ❑ D'ailleurs, on devrait parler d'**acteurs** et non d'**utilisateurs**.

## 3.2. LES ACTEURS

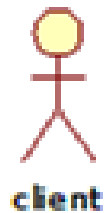
- ❑ Un **acteur** correspond à une entité (humain ou non) qui aura une interaction avec le système.
- ❑ Parmi les acteurs, nous distinguons :
  - les **acteurs principaux** agissent directement sur le système. Il s'agit d'entités qui ont des besoins d'utilisation du système. On peut donc considérer que les futurs utilisateurs du logiciel sont les acteurs principaux.

## 3.2. LES ACTEURS

- les **acteurs secondaires** n'ont pas de besoin direct d'utilisation. Ils peuvent être soit consultés par le système à développer, soit récepteur d'informations de la part du système. Cela est généralement un autre système (logiciel) avec lequel le nôtre doit échanger des informations.

## 3.2. LES ACTEURS

□ Certains acteurs sont de type *humain*. Ils sont alors représentés par un bonhomme en fil de fer (parfois appelé « stick man » en anglais) et on indique leur rôle en dessous.

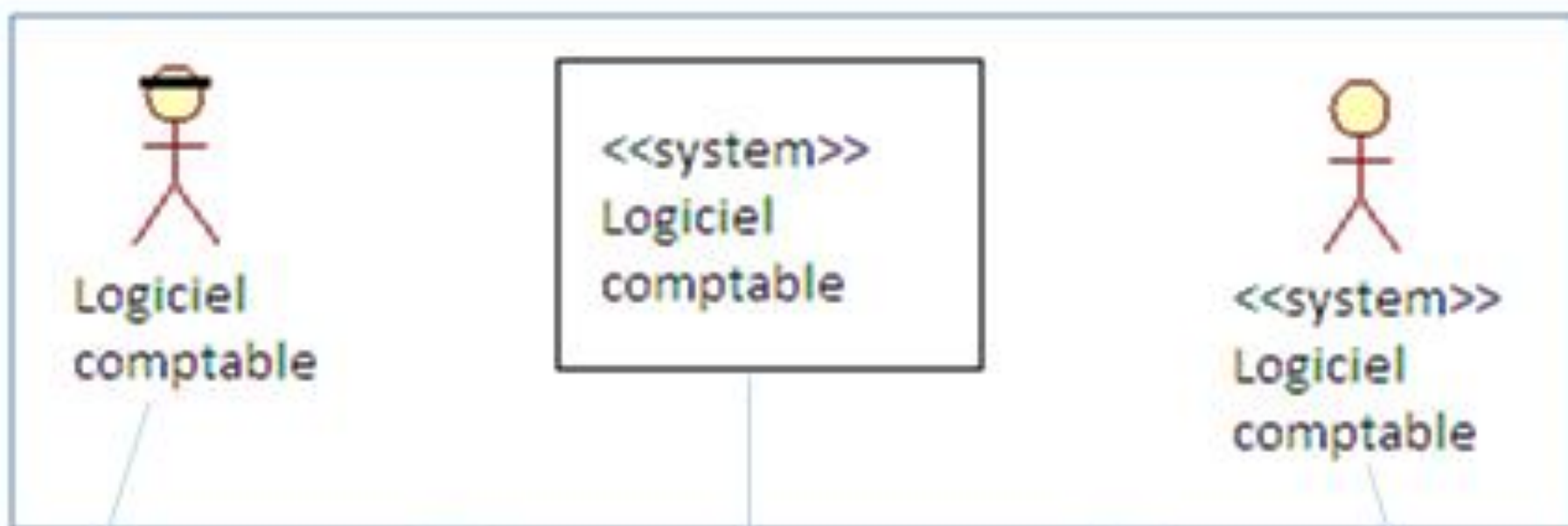


---

La représentation d'un acteur de type humain

## 3.2. LES ACTEURS

□ Pour représenter un acteur de type ***non-humain***, on peut utiliser une représentation graphique différente et/ou ajouter une indication supplémentaire (appelé le ***stéréotype***). Vous pouvez voir les différentes représentations du même acteur dans le schéma suivant.



Acteur non humain sous forme de bonhomme en fil de fer avec une grande barre noire sur la tête.

Acteur non humain sous forme de boîte avec indication du stéréotype `<<système>>`

Acteur non humain sous forme de bonhomme en fil de fer avec indication du stéréotype `<<système>>`

La représentation d'un acteur de type non-humain



## 3.2. LES ACTEURS

□ Vous l'avez compris, l'environnement du système est composé d'acteurs qui agissent sur le système ou avec lesquels le système aura une interaction.

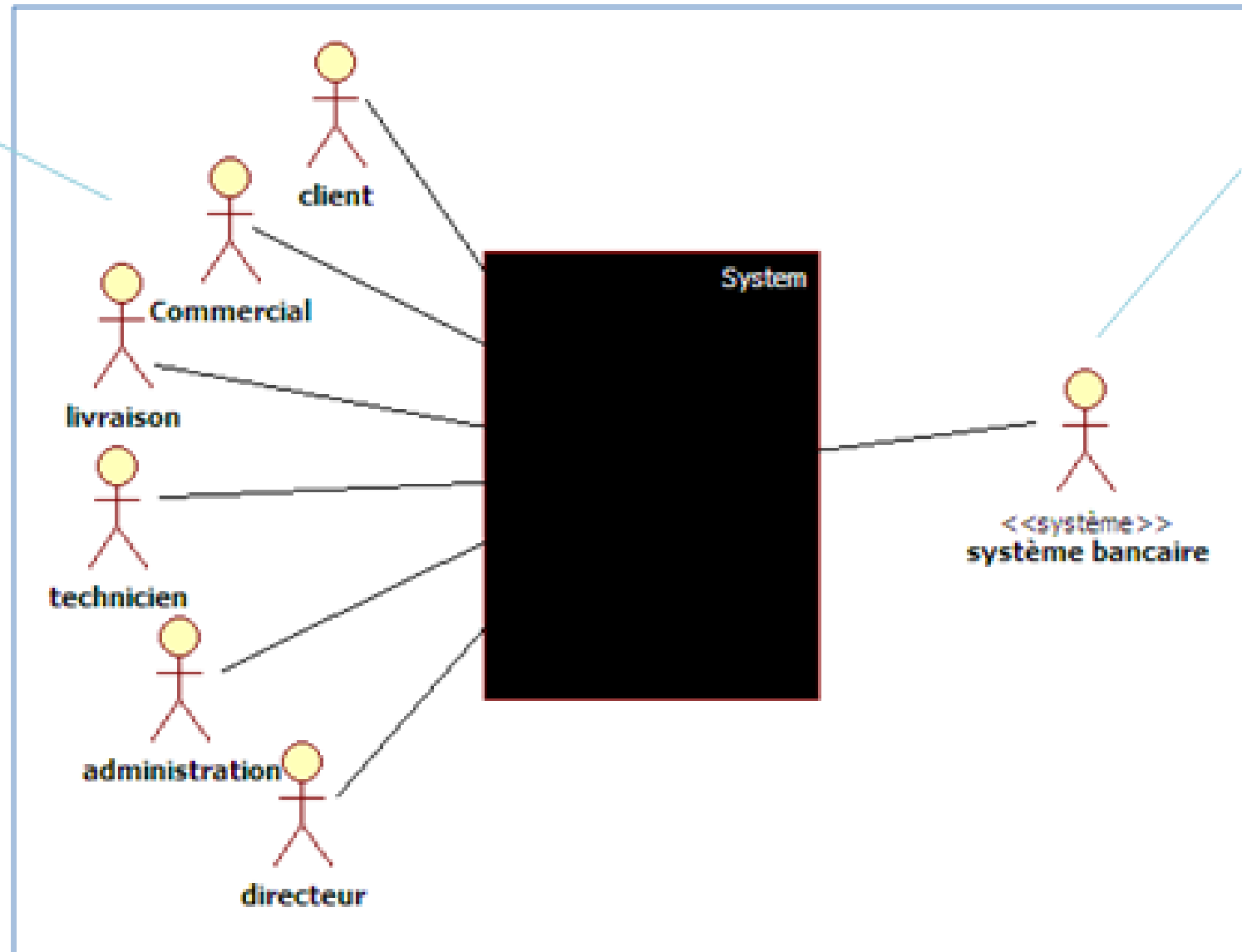
### 3.3. RÉALISATION DU DIAGRAMME DE CONTEXTE

□ Après avoir déterminé tous les acteurs du système à mettre en place, pour réaliser le diagramme de contexte, les acteurs sont placés à côté de la boîte noire, en fonction de leur type :

- on place généralement les acteurs principaux à gauche ;
- et les acteurs secondaires à droite du système

□ Comme illustré à la figure suivante :

Acteurs  
principaux



Acteur  
secondaire

Le diagramme de contexte

## 3.4. DESCRIPTION TEXTUELLE DES ACTEURS

□ Il s'agit juste de réaliser un tableau à deux (2) colonnes listant les différents acteurs dans la première colonne et le rôle de chaque acteur dans la deuxième colonne comme illustré à la figure ci-dessous :

## 3.4. DESCRIPTION TEXTUELLE DES ACTEURS

### Description textuelle des acteurs du système

ACTEURS	DESCRIPTION
Acteur 1	Description du rôle de l'acteur 1

## 3.4. DESCRIPTION TEXTUELLE DES ACTEURS

- ❑ Ça y est, nous venons de créer notre premier diagramme !
- ❑ Continuons maintenant avec la découverte du contenu du système : ***La décomposition en package.***

## Chap. 4 : DIAGRAMME DE PACKAGE

- ❑ Le diagramme de Package correspond à la décomposition du système en parties, appelé « package ».

## 4.1. POURQUOI FAUT-IL DÉCOMPOSER LE SYSTÈME EN PARTIES ?

- ❑ Un proverbe chinois connu dit que « ***un éléphant ne peut pas être avalé d'un coup. La bête doit être coupée en morceaux si on veut en venir à bout, puis les morceaux sont cuisinés séparément avant d'être mangés*** ».
- ❑ Les besoins très différents des acteurs et le nombre de fonctionnalités dont le futur logiciel devra disposer nous semble assez souvent compliqué.



## 4.1. POURQUOI FAUT-IL DÉCOMPOSER LE SYSTÈME EN PARTIES ?

- ❑ Pour y voir clair et pour nous faciliter la tâche, on peut découper le futur logiciel en parties distinctes, en fonction des « **familles** » de fonctionnalités et de façon à pouvoir les analyser séparément.
- ❑ Chacune de ces parties correspond à un **domaine fonctionnel** ou **package**.

## 4.2. DÉCOUPAGE DU SYSTÈME EN PLUSIEURS PACKAGES

- ❑ Pour identifier les parties bien distinctes parmi les fonctionnalités du système à mettre en place, posez-vous cette question : « ***Est-ce que le logiciel comporte des parties différentes qui pourraient être analysées séparément ?*** ».
- ❑ Ne faites pas l'erreur de décomposer le système en fonction des acteurs. Un package (ou une partie) peut être utilisé par plusieurs acteurs !

## 4.3. RÉALISATION DU DIAGRAMME DE PACKAGE

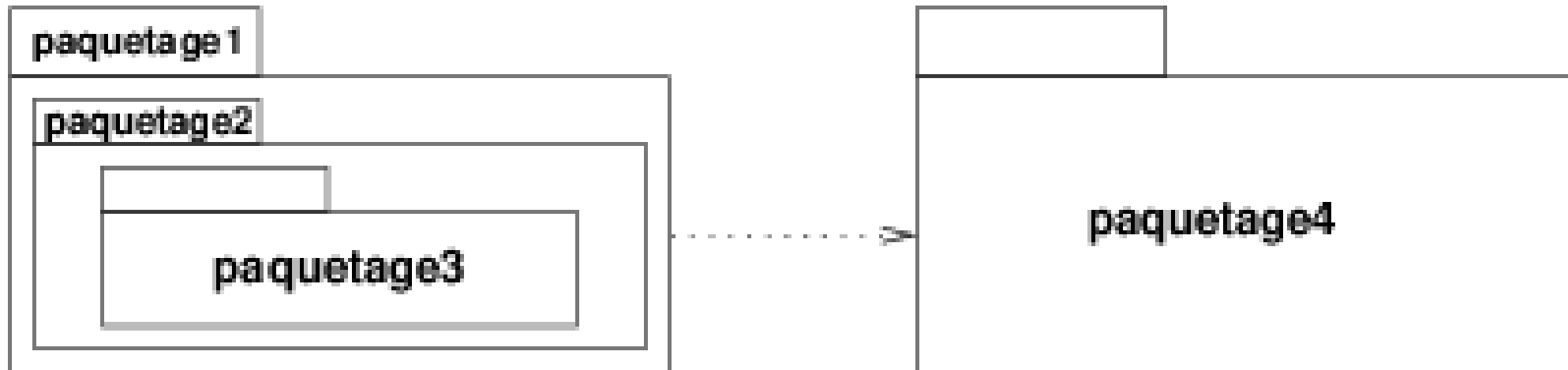
- ❑ Un paquetage est représenté par un rectangle avec une cartouche rectangulaire en haut à gauche.
- ❑ Le nom du paquetage est indiqué dans le rectangle principal ou dans la cartouche en haut à gauche :



## 4.3. RÉALISATION DU DIAGRAMME DE PACKAGE

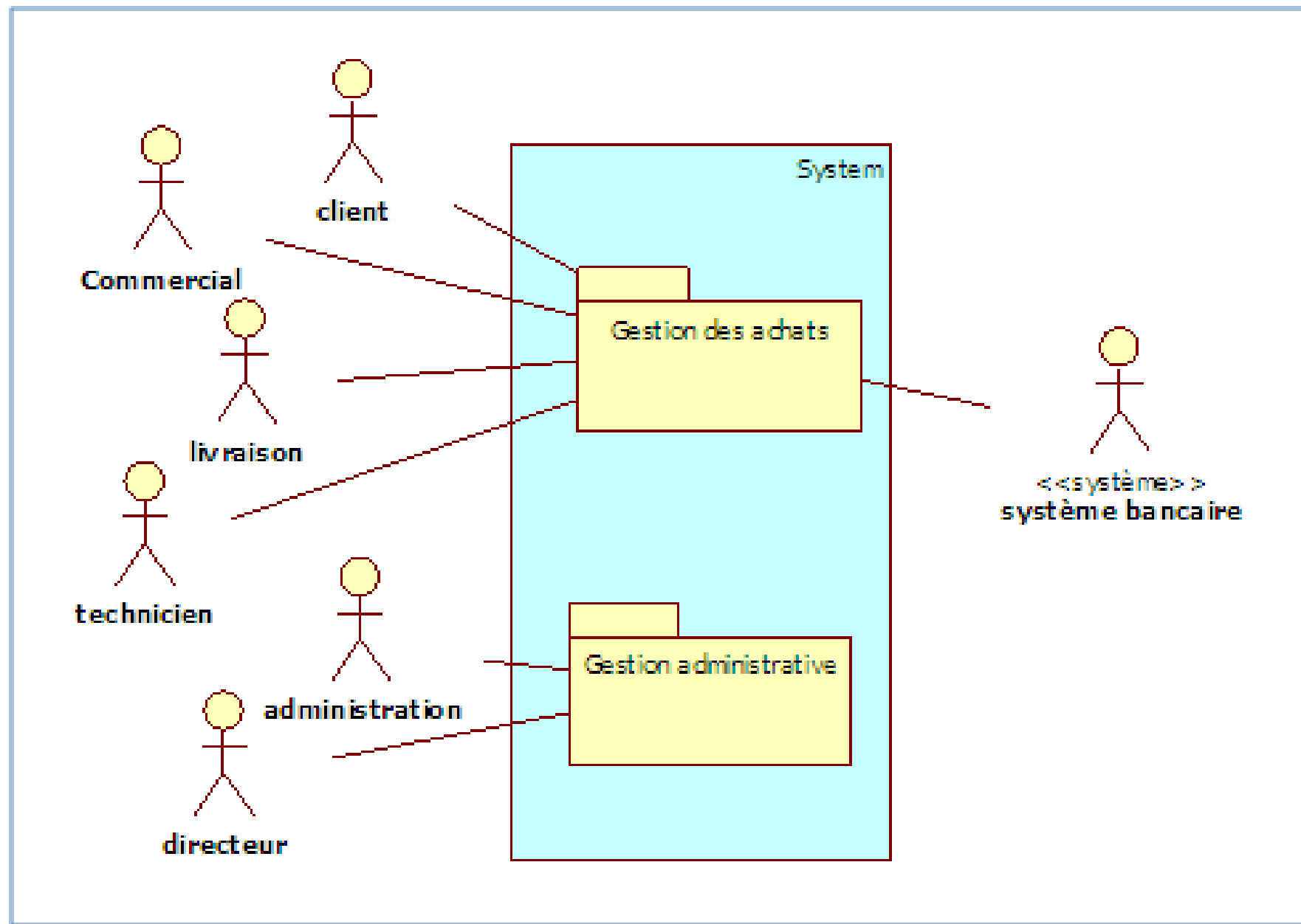
- ❑ Les paquetages peuvent contenir d'autres paquetages pour construire une imbrication d'espaces de nommage.
- ❑ Si un élément d'un paquetage utilise un élément d'un autre paquetage alors le premier paquetage dépend du second.
- ❑ La dépendance est notée par une flèche pointillée allant du paquetage utilisateur vers le paquetage contenant l'élément utilisé.

## 4.3. RÉALISATION DU DIAGRAMME DE PACKAGE



## 4.3. RÉALISATION DU DIAGRAMME DE PACKAGE

□ Certains diagrammes de package sont réalisés en mettant en évidence les acteurs qui interviennent dans chacun de ces packages comme illustré à la figure suivante :



Le diagramme de packages

## 4.4. DESCRIPTION TEXTUELLE DES PACKAGES

❑ Ici aussi, il s'agit de réaliser un tableau à deux (2) colonnes listant les différents packages du système dans la première colonne et la description de des fonctionnalités liées à chaque package dans la deuxième colonne :



## 4.4. DESCRIPTION TEXTUELLE DES PACKAGES

### Description textuelle des packages du système

PACKAGE	DESCRIPTION
Package 1	Description des fonctionnalités du Package 1

## 4.4. DESCRIPTION TEXTUELLE DES PACKAGES

- ❑ Voilà, c'est notre deuxième diagramme ensemble !
- ❑ Continuons maintenant avec l'étape 3 de notre analyse de besoins principaux des utilisateurs : le diagramme de cas d'utilisation.

## Chap. 5 : DIAGRAMME DE CAS D'UTILISATION

- ❑ Nous avons donc identifié les acteurs, et les grandes familles de fonctionnalités (packages).
- ❑ Maintenant, il s'agit de définir plus en détail les besoins de chaque acteur dans ces packages, en répondant à la question : ***QUI devra pouvoir faire QUOI grâce au logiciel ?***
- ❑ Par exemple, ***que souhaite faire le technicien sur le site ? Que souhaite faire le client ? Etc.***

## Chap. 5 : DIAGRAMME DE CAS D'UTILISATION

- ❑ Les diagrammes de cas d'utilisation sont donc utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel.
- ❑ Ils permettent de décrire l'interaction entre les acteurs et le système.

## 5.1. LES CAS D'UTILISATION

- ❑ Le cas d'utilisation représente une fonctionnalité du système à effectuer par un acteur ou plusieurs acteurs.
- ❑ On peut faire un diagramme de cas d'utilisation pour le logiciel entier ou pour chaque package.

## 5.2. RÉALISATION DU DIAGRAMME DE CAS D'UTILISATION

- Un cas d'utilisation se représente par une ellipse contenant le nom du cas d'utilisation (phrase commençant par un verbe à l'infinitif) et optionnellement un stéréotype au dessus du nom :



## 5.2. RÉALISATION DU DIAGRAMME DE CAS D'UTILISATION

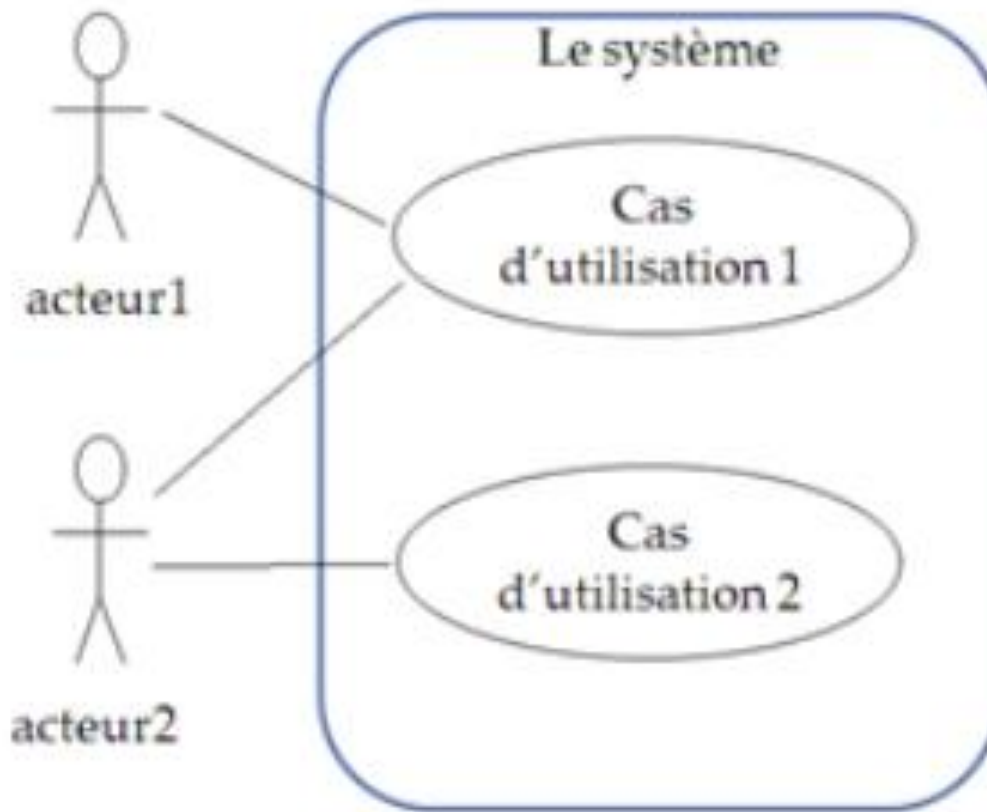
- Les différents cas d'utilisation sont représentés à l'intérieur d'un même rectangle représentant les limites du système ou du package.

## 5.3. RELATION ENTRE ACTEURS ET CAS D'UTILISATION

- ❑ A chaque acteur est associé un ou plusieurs cas d'utilisations, la relation d'association peut aussi être appelée ***relation de communication***.
- ❑ Elle est représentée par un trait reliant l'acteur au cas d'utilisation.



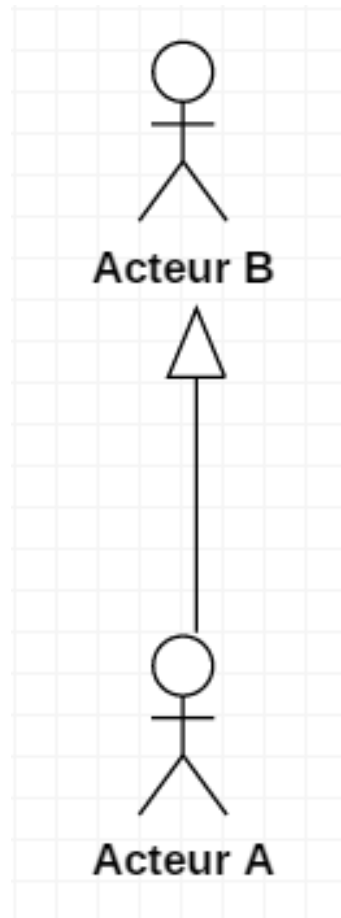
### 5.3. RELATION ENTRE ACTEURS ET CAS D'UTILISATION



## 5.4. RELATIONS ENTRE ACTEURS

- ❑ Il n'y a qu'un seul type de relation possible entre acteurs : ***la relation de généralisation.***
- ❑ Il y a généralisation entre un acteur A et un acteur B lorsqu'on peut dire : ***A est une sorte de B.***
- ❑ L'acteur A peut faire avec le système tout ce que peut faire l'acteur B, plus d'autres actions.

## 5.4. RELATIONS ENTRE ACTEURS



## 5.5. LES RELATIONS ENTRE CAS D'UTILISATION

## 5.5.1. RELATION D'INCLUSION

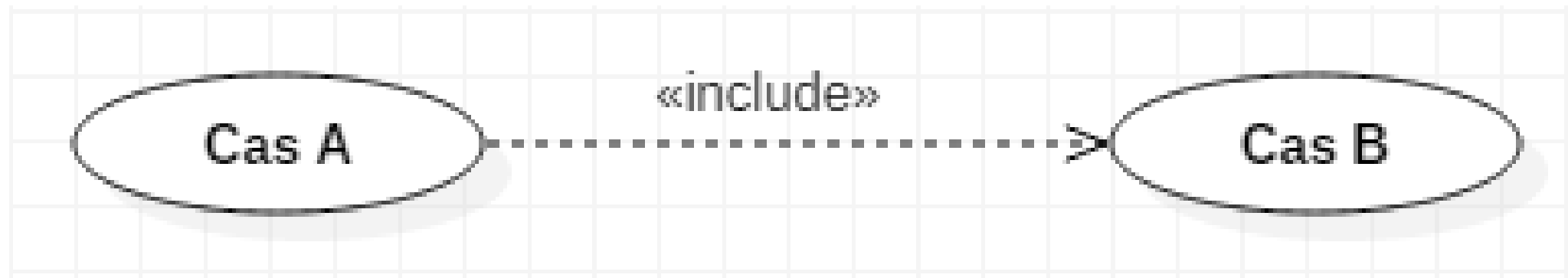
- ❑ La relation d'inclusion sert à enrichir un cas d'utilisation par un autre cas d'utilisation (c'est une sous fonction).
- ❑ Dans un diagramme des cas d'utilisation, cette relation est représentée par une flèche pointillée reliant les 2 cas d'utilisation et munie du stéréotype << *include* >>.
- ❑ L'inclusion permet de :

## 5.5.1. RELATION D'INCLUSION

- ❑ Un cas d'utilisation A inclut un cas d'utilisation B si le comportement décrit par le cas d'utilisation A inclut le comportement du cas d'utilisation B : le cas d'utilisation A dépend de B.
- ❑ Lorsque A est sollicité, B l'est obligatoirement, comme une partie de A.
- ❑ Les inclusions permettent essentiellement de factoriser une partie de la description d'un cas d'utilisation qui serait commune à d'autres cas d'utilisation.

## 5.5.1. RELATION D'INCLUSION

- ❑ Les inclusions permettent également de décomposer un cas complexe en sous-cas plus simples.



## 5.5.2. RELATION D'EXTENSION

- ❑ On dit qu'un cas d'utilisation A étend un cas d'utilisation B lorsque le cas d'utilisation B peut être appelé au cours de l'exécution du cas d'utilisation A.
- ❑ Exécuter A peut éventuellement entraîner l'exécution de B : contrairement à l'inclusion, l'extension est optionnelle.
- ❑ Cette dépendance est symbolisée par le stéréotype << ***extend*** >>

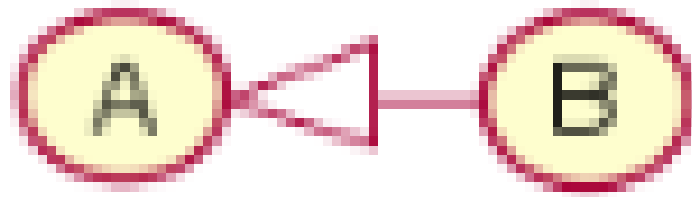


## 5.5.2. RELATION D'EXTENSION



### 5.5.3. LA GÉNÉRALISATION

□ On dit qu'un cas d'utilisation A est une généralisation d'un cas d'utilisation B lorsque le cas d'utilisation B est une sorte du cas d'utilisation A.



## 5.6. DESCRIPTION TEXTUELLE D'UN CAS D'UTILISATION

- Dans cette partie, nous allons découvrir l'utilité de décrire les scénarios des cas d'utilisation, ainsi que les éléments nécessaires à la description du déroulement des d'actions.
- Les diagrammes réalisés jusqu'à maintenant (**diagramme de contexte, diagramme de packages, diagramme de cas d'utilisation**) nous ont permis de découvrir petit à petit les fonctionnalités (appelées aussi des cas d'utilisation) que l'on devrait avoir dans le futur logiciel.

## 5.6. DESCRIPTION TEXTUELLE D'UN CAS D'UTILISATION

- ❑ Nous allons désormais parler de l'interaction entre les acteurs et le système : il s'agit de **décrire** la chronologie des actions qui devront être réalisées par les acteurs et par le système lui-même.
- ❑ On parle d'ailleurs de **scénarios**.

## 5.6. DESCRIPTION TEXTUELLE D'UN CAS D'UTILISATION

□ La description d'un cas d'utilisation permet de :

- clarifier le déroulement de la fonctionnalité ;
- décrire la chronologie des actions qui devront être réalisées ;
- d'identifier les parties redondantes pour en déduire des cas d'utilisation plus précises qui seront utilisées par inclusion, extension ou généralisation/spécialisation. Et oui, dans ce cas nous réaliserons des itérations sur les diagrammes de cas d'utilisation ;

## 5.6. DESCRIPTION TEXTUELLE D'UN CAS D'UTILISATION

- d'indiquer d'éventuelles contraintes déjà connues et dont les développeurs vont devoir tenir compte lors de la réalisation du logiciel.

Ces contraintes peuvent être de nature diverse.

- Les descriptions peuvent aider à découvrir d'autres cas d'utilisation que l'on pourrait ajouter. Il s'agit, dans ce cas, d'une nouvelle itération sur les diagrammes de cas d'utilisation.

## 5.6. DESCRIPTION TEXTUELLE D'UN CAS D'UTILISATION

- ❑ Décrire le déroulement des actions pour un cas d'utilisation peut vous paraître simple, mais c'est un travail qui demande beaucoup de réflexion et de questionnement.
- ❑ On commence souvent par une première description, basée sur les informations que l'on a obtenu auprès du client et/ou des futurs utilisateurs. Lors de la réalisation de cette première description, on découvre souvent des questions auxquelles nous n'avons pas de réponse.

## 5.6. DESCRIPTION TEXTUELLE D'UN CAS D'UTILISATION

- ❑ On réalise alors une **fiche descriptive** pour chaque cas d'utilisation, de façon à raconter l'histoire du déroulement des différentes actions.
- ❑ Cette fiche descriptive doit comporter 4 parties :
  1. L'identification
  2. La description des scénarios
  3. La fin et les post-conditions
  4. Les compléments



## 5.6. DESCRIPTION TEXTUELLE D'UN CAS D'UTILISATION

- ❑ Nous allons nous intéresser à un cas d'utilisation simple de notre *TP* : *Gestion des ventes de matériels en ligne* : « **Consulter le catalogue des produits** ».
- ❑ À partir de la dernière version du diagramme de cas d'utilisation du package « Gestion des achats », nous verrons comment cela devrait se faire en pratique.
- ❑ Réalisons ensemble la fiche descriptive (description textuelle) de ce cas d'utilisation.

## 5.6.1. PARTIE 1 : L'IDENTIFICATION

□ Dans la partie identification, on indique :

- **Le numéro du cas d'utilisation**, de façon aléatoire. Cela permet ensuite de les classer plus facilement ;
- **le nom du cas d'utilisation** (avec indication du package) ;
- **l'acteur (ou les acteurs)** s'il s'agit d'un cas d'utilisation principal ou le nom du cas d'utilisation principal lorsqu'il s'agit d'un cas d'utilisation interne ;  
une description succincte du cas d'utilisation ;

## 5.6.1. PARTIE 1 : L'IDENTIFICATION

- ***la date de rédaction de la fiche et l'auteur***, voire éventuellement les dates de mise à jour et les auteurs successifs ;
- ***les pré-conditions*** : il s'agit des conditions obligatoires au bon déroulement du cas d'utilisation.
- ***les événements à l'origine du démarrage*** du cas d'utilisation ;

## 5.6.1. PARTIE 1 : L'IDENTIFICATION

□ Voici donc comment la fiche descriptive débiterait pour notre cas d'utilisation  
**« Consulter le catalogue des produits » :**

## Cas n° 1

**Nom** : Consulter catalogue produit (package « Gestion des achats »)

**Acteur(s)** : Acheteur (client ou commercial)

**Description** : La consultation du catalogue doit être possible pour un client ainsi que pour les commerciaux de l'entreprise.

**Auteur** : Carina Roels

**Date(s)** : 10/11/2013 (première rédaction)

**Pré-conditions** : L'utilisateur doit être authentifié en tant que client ou commercial (Cas d'utilisation « S'authentifier » – package « Authentification »)

**Démarrage** : L'utilisateur a demandé la page « Consultation catalogue »

## 5.6.2. PARTIE 2 : LA DESCRIPTION DES SCÉNARIOS (OU DIALOGUE)

□ Nous allons maintenant décrire les scénarios qui explicitent la chronologie des actions qui seront réalisées par l'utilisateur et le système.

□ Il existe 3 parties :

- ***Le scénario nominal*** : Il s'agit ici de décrire le déroulement idéal des actions, où tout va pour le mieux.

## 5.6.2. PARTIE 2: LA DESCRIPTION DES SCÉNARIOS (OU DIALOGUE)

- **Les scénarios alternatifs** : Ici, il s'agit de décrire les éventuelles étapes différentes liées aux choix de l'utilisateur, par exemple. C'est le cas des étapes liées à des conditions.
- **Les scénarios d'exception** : On parlera de scénario d'exception lorsque une étape du déroulement pourrait être perturbée à cause d'un événement anormal. Par exemple, lorsqu'une recherche de client ne trouve aucun client correspondant aux critères fournis.

## 5.6.2. PARTIE 2 : LA DESCRIPTION DES SCÉNARIOS (OU DIALOGUE)

- ❑ Les scénarios mettent en évidence les interactions entre les actions de l'utilisateur et le système (le logiciel).
- ❑ Cela peut se faire par une liste numérotée d'actions ou sous forme de tableau qui démontre clairement ce qui est réalisé par l'utilisateur et ce qui est du ressort du système.
- ❑ La représentation de ces scénarios vous est personnelle.



## 5.6.2. PARTIE 2 : LA DESCRIPTION DES SCÉNARIOS (OU DIALOGUE)

❑ Ici, nous avons choisi de proposer une description textuelle, en listant le déroulement du scénario nominal par des chiffres (1, 2, 3...) et les scénarios alternatif et d'exception par des lettres rattachées aux numéros de l'action principale (1 a, 1 b, 2a, 2b...).

❑ Voici la description des scenarios pour notre cas d'utilisation « **Consulter le catalogue des produits** » :

## Le scénario nominal

1. **Le système** affiche une page contenant la liste les catégories de produits.
2. *L'utilisateur* sélectionne une des catégories.
3. **Le système** recherche les produits qui appartiennent à cette catégorie.
4. **Le système** affiche une description et une photo pour chaque produit trouvé.
5. *L'utilisateur* peut sélectionner un produit parmi ceux affichés.
6. **Le système** affiche les informations détaillées du produit choisi.
7. *L'utilisateur* peut ensuite quitter cette description détaillée.
8. **Le système** retourne à l'affichage des produits de la catégorie (retour à l'étape 4)

## Les scénarios alternatifs

- 2.a *L'utilisateur* décide de quitter la consultation de la catégorie de produits choisie.
- 2.b *L'utilisateur* décide de quitter la consultation du catalogue.
- 5.a *L'utilisateur* décide de quitter la consultation de la catégorie de produits choisie.
- 5.b *L'utilisateur* décide de quitter la consultation du catalogue.
- 7.a *L'utilisateur* décide de quitter la consultation de la catégorie de produits choisie.
- 7.b *L'utilisateur* décide de quitter la consultation du catalogue.

### 5.6.3. PARTIE 3 : LA FIN ET LES POST-CONDITIONS

□ La troisième partie d'une description détaillée d'un cas d'utilisation concerne :

- la fin du cas d'utilisation ;
- les post-conditions.

### 5.6.3. PARTIE 3 : LA FIN ET LES POST-CONDITIONS

#### La fin

- ❑ La fin permet de récapituler toutes les situations d'arrêt du cas d'utilisation. Cela permet parfois de s'apercevoir que l'on n'a pas vu tous les cas de figure qui peuvent se présenter.
- ❑ Par exemple, notre cas d'utilisation peut s'arrêter aux étapes 2, 5 et 7 car l'utilisateur peut décider de quitter la consultation du catalogue pour revenir à l'accueil par exemple.

## 5.6.3. PARTIE 3 : LA FIN ET LES POST-CONDITIONS

### Les post-conditions

- ❑ Les post-conditions nous indiquent un résultat tangible qui est vérifiable après l'arrêt du cas d'utilisation et qui pourra témoigner du bon fonctionnement. Cela pourrait être une information qui a été enregistrée dans une base de données ou dans un fichier, ou encore un message envoyé par mail, etc.
- ❑ Par exemple, ici il n'y en a pas car la consultation du catalogue ne donne pas lieu à l'enregistrement d'informations dans un fichier ou une base de données.

### 5.6.3. PARTIE 3 : LA FIN ET LES POST-CONDITIONS

□ Voici la partie 3 de notre cas d'utilisation « *Consulter le catalogue des produits* » :

**Fin :** Scénario nominal : aux étapes 2, 5 ou 7, sur décision de l'utilisateur

**Post-conditions :** Aucun

## 5.6.4. PARTIE 4 : LES COMPLÉMENTS

□ Les compléments peuvent porter sur des sujets variés :

- l'ergonomie ;
- la performance attendue ;
- des contraintes (techniques ou non) à respecter ;
- des problèmes non résolus (ou questions à poser au client et aux futurs utilisateurs).



## 5.6.4. PARTIE 4 : LES COMPLÉMENTS

□ La figure suivante présente un exemple des compléments sur notre cas d'utilisation :

## 5.6.4. PARTIE 4 : LES COMPLÉMENTS

### **Ergonomie**

L'affichage des produits d'une catégorie devra se faire par groupe de 15 produits. Toutefois, afin d'éviter à l'utilisateur d'avoir à demander trop de pages, il devra être possible de choisir des pages avec 30, 45 ou 60 produits.

### **Performance attendue**

La recherche des produits, après sélection de la catégorie, doit se faire de façon à afficher la page des produits en moins de 10 secondes.

## 5.6.4. PARTIE 4 : LES COMPLÉMENTS

### Problèmes non résolus

Nous avons fait la description basée sur l'information que les produits appartiennent à une catégorie. Est-ce qu'il existe des sous-catégories ?

Si tel est le cas, la description devra être revue.

Est-ce que la consultation du catalogue doit être possible uniquement par catégorie ou est-ce qu'on doit prévoir d'autres critères de recherche de produits ?

Doit-on prévoir un affichage trié sur des critères choisis par l'utilisateur (par exemple : par tranche de prix, par disponibilité, etc) ?

## 5.6.4. PARTIE 4 : LES COMPLÉMENTS

- ❑ Nous n'avons pas vu tous les cas de figure qui peuvent se présenter.
- ❑ Chaque cas d'utilisation sera particulier et vous devez faire preuve d'imagination, de propositions et surtout d'écoute pour arriver à des descriptions qui reflètent le réel besoin des utilisateurs.

## 5.6.4. PARTIE 4 : LES COMPLÉMENTS

□ Voici la synthèse de notre deuxième fiche :

## Cas n°1

**Nom :** Consulter catalogue produit (package « Gestion des achats »)

**Acteur(s) :** Acheteur (client ou commercial)

**Description :** La consultation du catalogue doit être possible pour un client ainsi que pour les commerciaux de l'entreprise.

**Auteur :** Carina Roels

**Date(s) :** 10/11/2013 (première rédaction)

**Pré-conditions :** L'utilisateur doit être authentifié en tant que client ou commercial (Cas d'utilisation « S'authentifier » – package « Authentification »)

**Démarrage :** L'utilisateur a demandé la page « Consultation catalogue »

## DESCRIPTION

### Le scénario nominal :

1. Le système affiche une page contenant la liste des catégories de produits.
2. *L'utilisateur* sélectionne une des catégories.
3. **Le système** recherche les produits qui appartiennent à cette catégorie.
4. **Le système** affiche une description et une photo pour chaque produit trouvé.
5. *L'utilisateur* peut sélectionner un produit parmi ceux affichés.
6. **Le système** affiche les informations détaillées du produit choisi.
7. *L'utilisateur* peut ensuite quitter cette description détaillée.
8. **Le système** retourne à l'affichage des produits de la catégorie (retour à l'étape 4)

## Les scénarios alternatifs

2.a *L'utilisateur* décide de quitter la consultation de la catégorie de produits choisie.

2.b *L'utilisateur* décide de quitter la consultation du catalogue.

5.a *L'utilisateur* décide de quitter la consultation de la catégorie de produits choisie.

5.b *L'utilisateur* décide de quitter la consultation du catalogue.

7.a *L'utilisateur* décide de quitter la consultation de la catégorie de produits choisie.

7.b *L'utilisateur* décide de quitter la consultation du catalogue.

**Fin :** Scénario nominal : aux étapes 2, 5 ou 7, sur décision de l'utilisateur

**Post-conditions :** Aucun

## COMPLEMENTS

### Ergonomie

L'affichage des produits d'une catégorie devra se faire par groupe de 15 produits.

Toutefois, afin d'éviter à l'utilisateur d'avoir à demander trop de pages, il devra être possible de choisir des pages avec 30, 45 ou 60 produits.



## Performance attendue

La recherche des produits, après sélection de la catégorie, doit se faire de façon à afficher la page des produits en moins de 10 secondes.

## Problèmes non résolus

Nous avons fait la description basée sur l'information que les produits appartiennent à une catégorie. Est-ce qu'il existe des sous-catégories ?

Si tel est le cas, la description devra être revue.

Est-ce que la consultation du catalogue doit être possible uniquement par catégorie ou est-ce qu'on doit prévoir d'autres critères de recherche de produits ?

Doit-on prévoir un affichage trié sur des critères choisis par l'utilisateur (par exemple : par tranche de prix, par disponibilité, etc) ?

## Chap. 6 : LE DIAGRAMME D'ACTIVITÉ

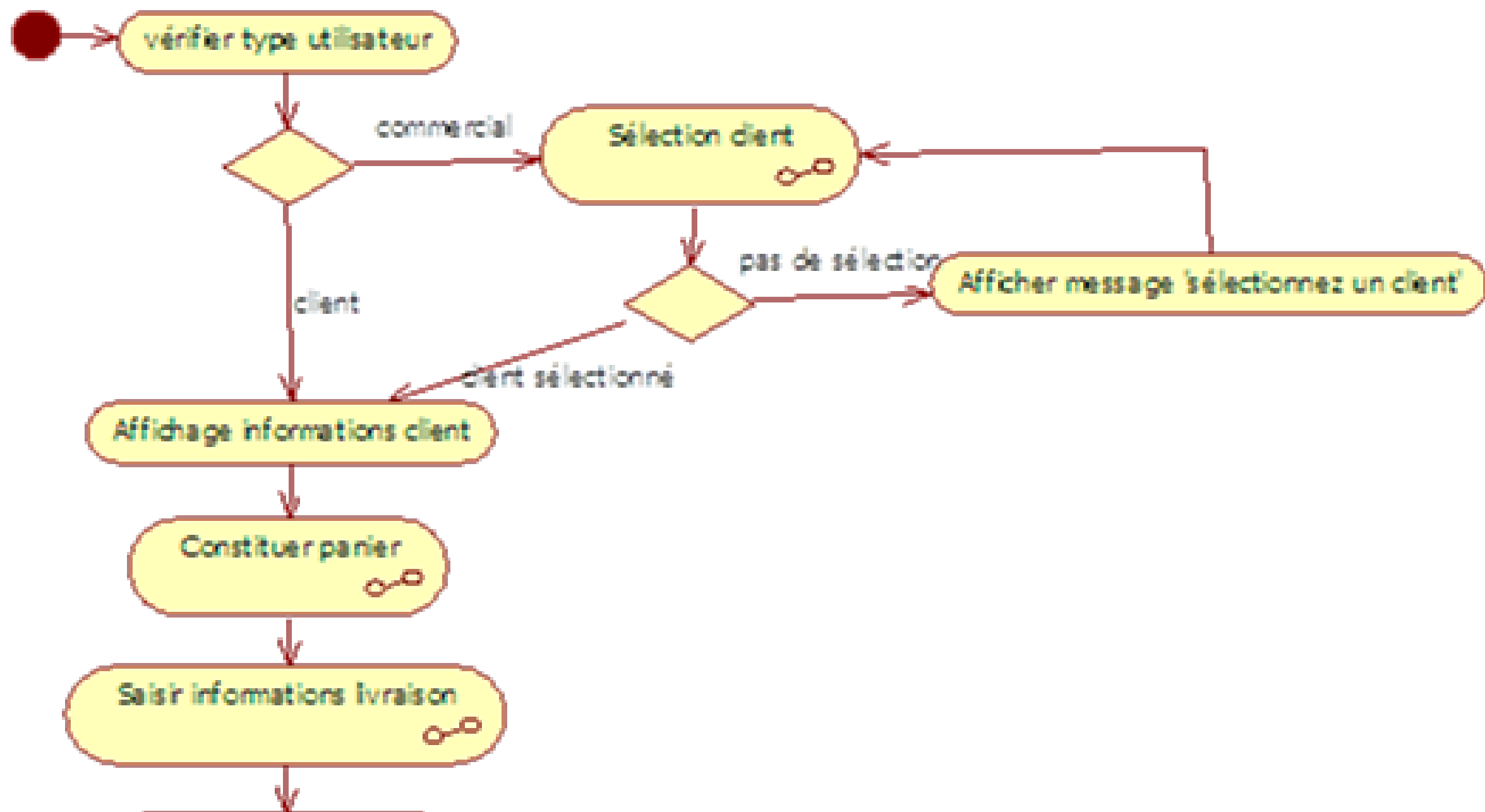
- ❑ Dans ce chapitre, nous allons brièvement voir l'alternative visuelle des descriptions détaillées des cas d'utilisation.
- ❑ Il s'agit du ***diagramme d'activité***.

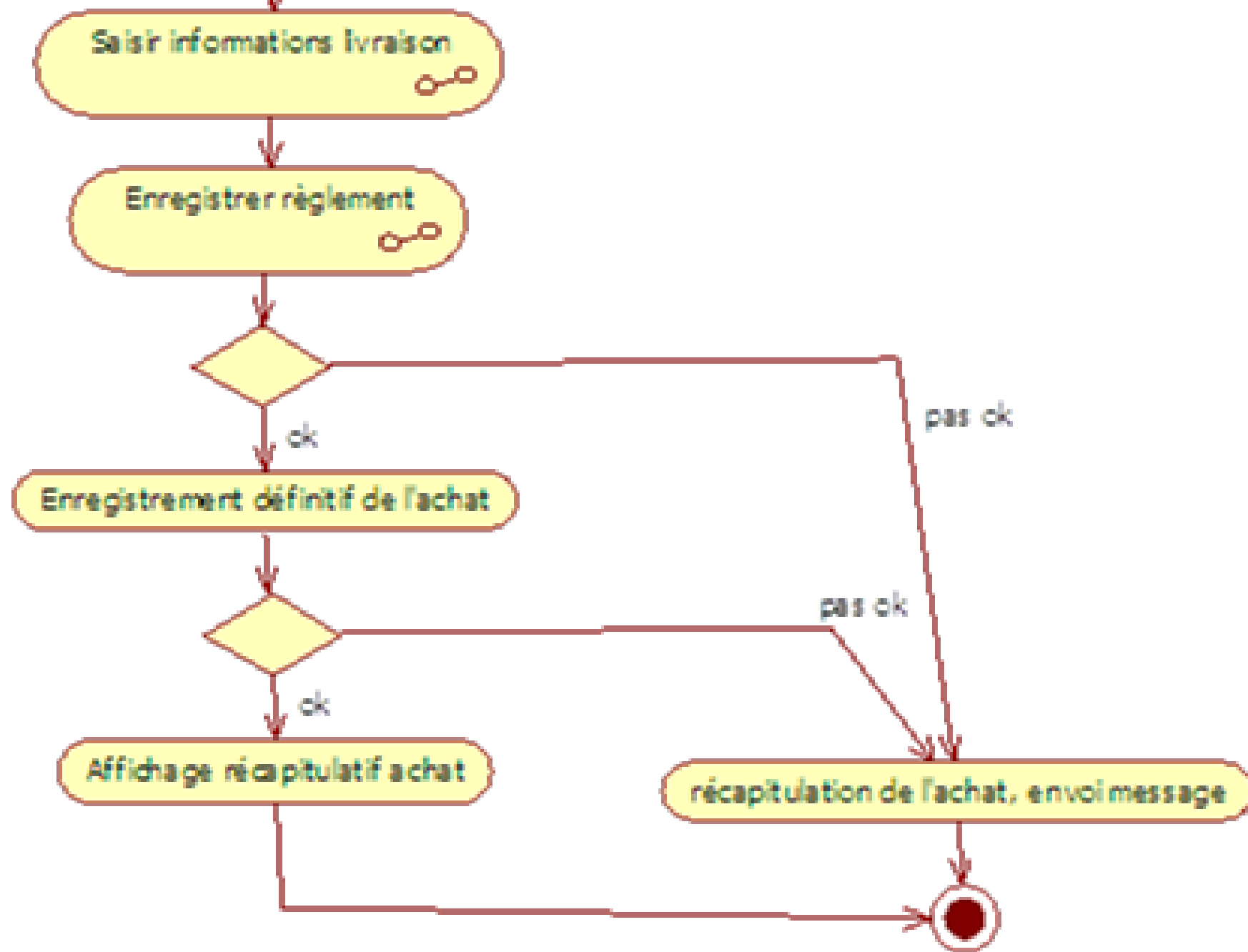
## 6.1. DESCRIPTION

- ❑ La fiche descriptive d'un cas d'utilisation peut contenir plusieurs scénarios alternatifs et/ou d'exception.
- ❑ Il est alors dès fois difficile d'avoir une vision de l'ensemble des actions.
- ❑ Le diagramme d'activité est un moyen graphique pour donner cette vision d'ensemble.

## 6.1. DESCRIPTION

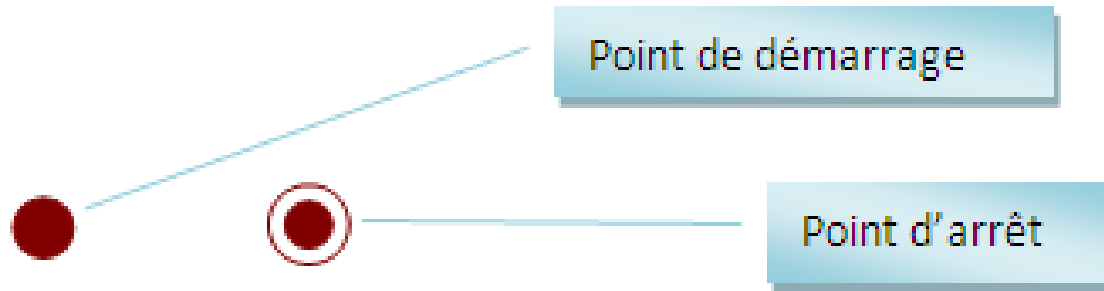
□ Voici un exemple du diagramme d'activité du cas d'utilisation « ***Enregistrer un achat*** » :





## 6.2.1. POINT DE DÉMARRAGE ET D'ARRÊT

□ Tout diagramme d'activité est composé d'un point de démarrage, d'un point arrêté, qui sont représentés par des cercles :



---

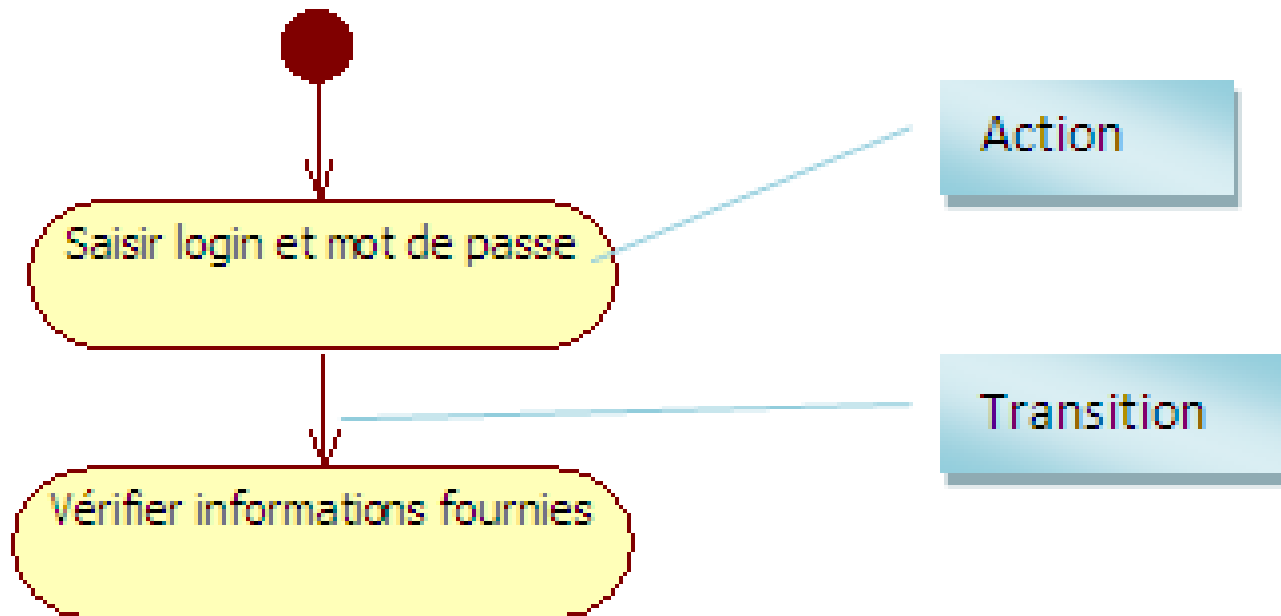
Point d'arrêt et point de démarrage

## 6.2.2. LES ACTIONS ET LES TRANSITIONS

- ❑ Je vous rappelle qu'un diagramme d'activité est une formalisation graphique des actions qui sont réalisées dans un cas d'utilisation.
- ❑ Le diagramme est donc organisé en actions réalisées soit par un acteur, soit par le système, relié par une flèche indiquant l'enchaînement des actions :



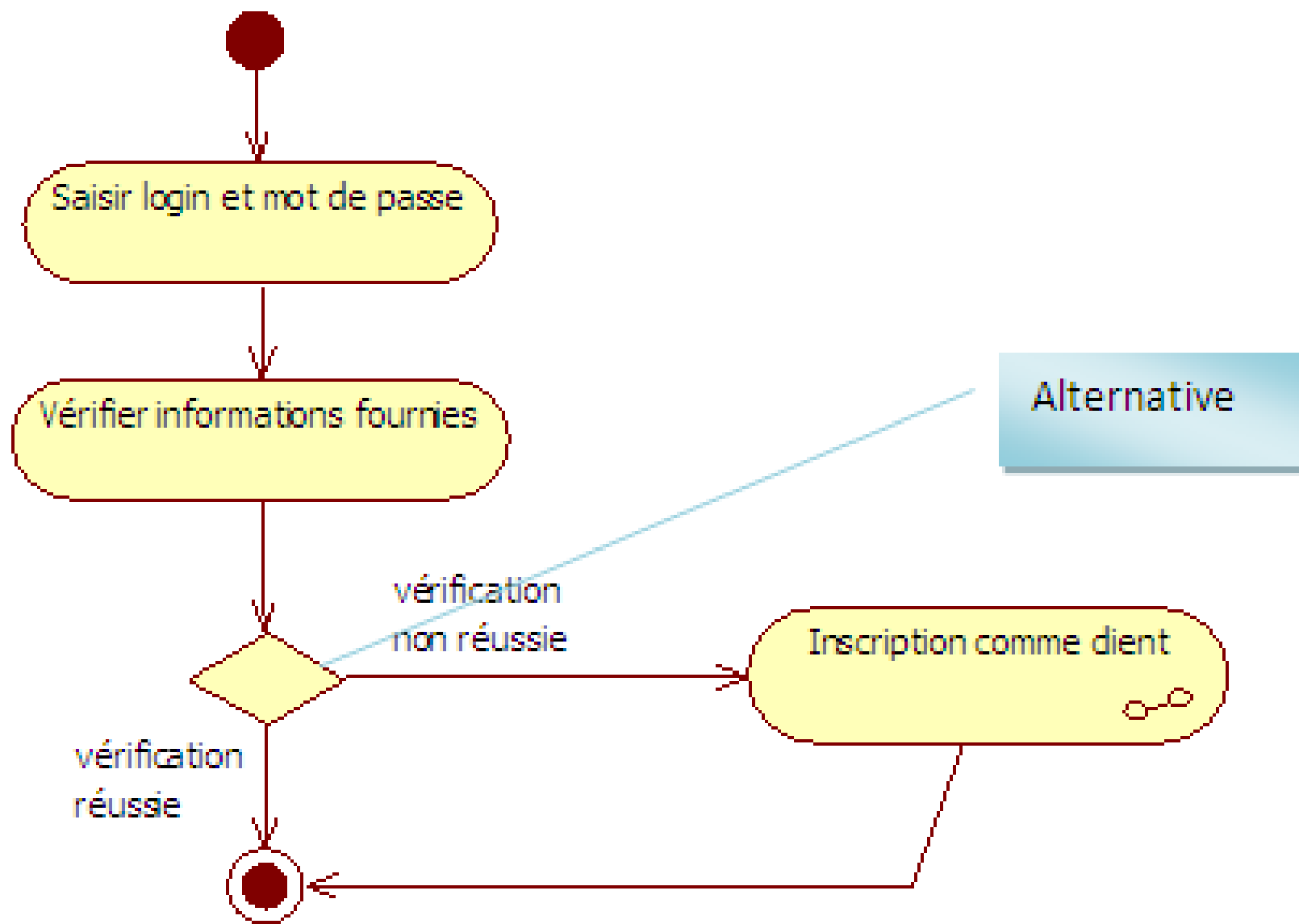
## 6.2.2. LES ACTIONS ET LES TRANSITIONS



Action / transition

### 6.2.3. L'ALTERNATIVE

- ❑ Elle permet d'indiquer les différents scénarios du cas d'utilisation dans un même diagramme.
- ❑ Dans l'exemple, il s'agit de la condition d'après laquelle le cas d'utilisation « Inscription comme client » serait appelé.



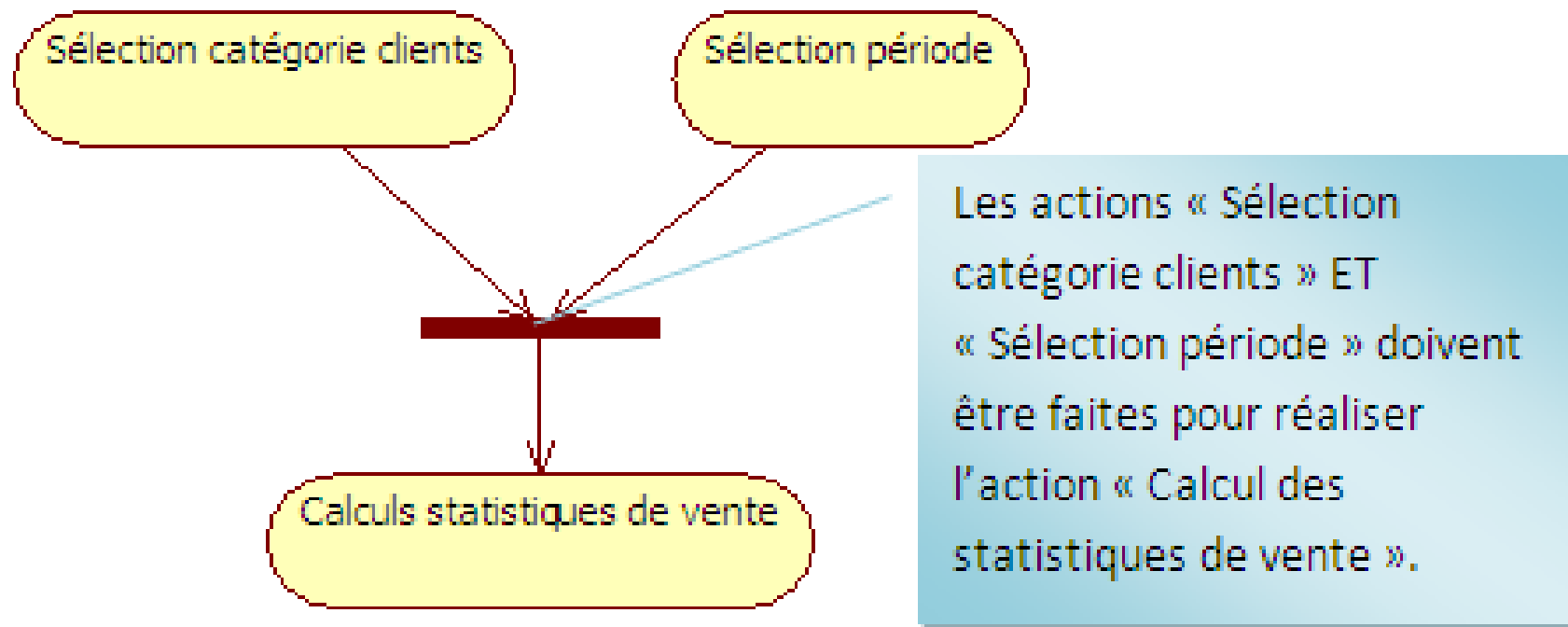
---

L'alternative

## 6.2.4. LA SYNCHRONISATION

- Elle indique qu'il faut avoir réalisé deux actions pour pouvoir réaliser la troisième en-dessous.

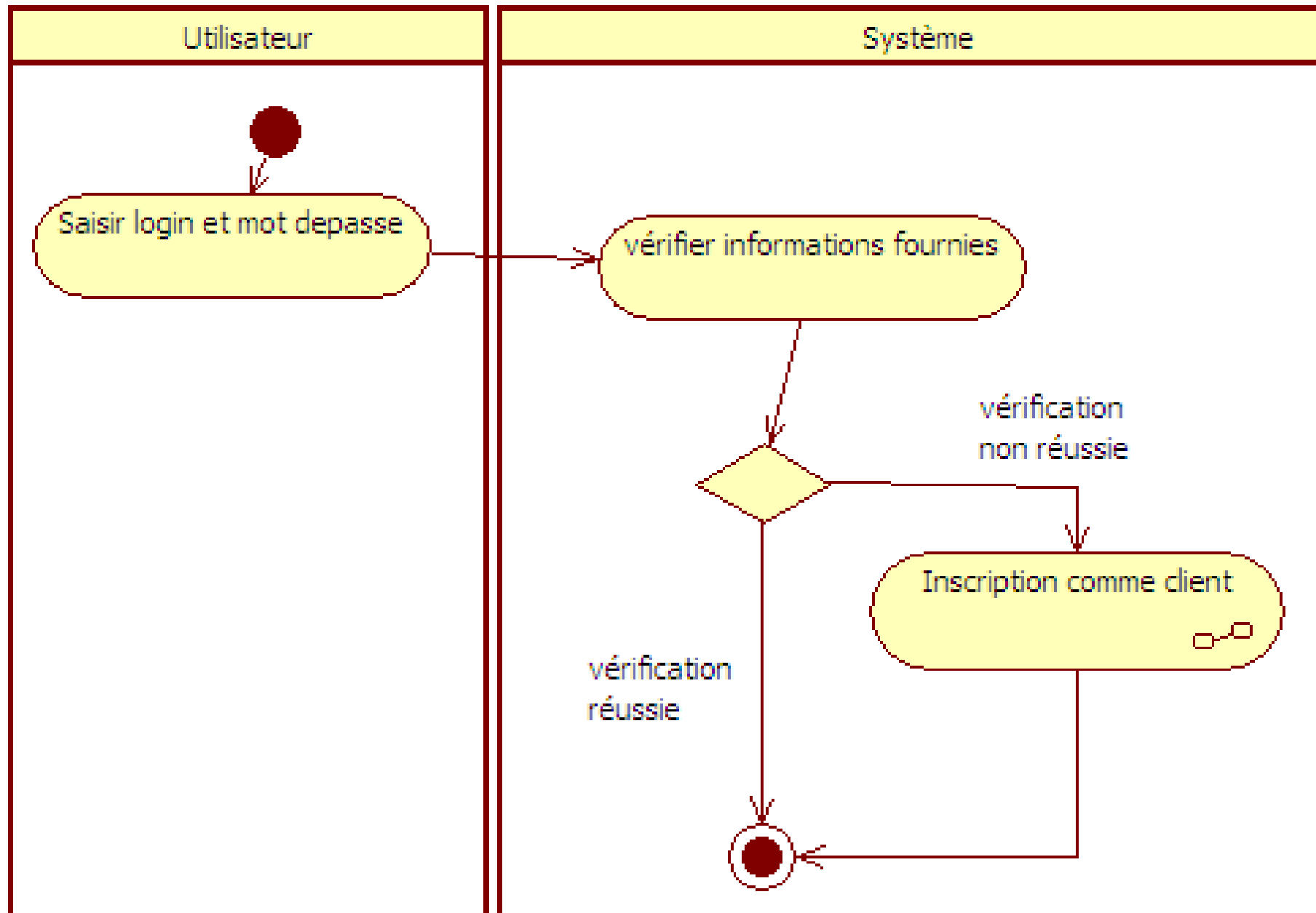
## 6.2.4. LA SYNCHRONISATION



La synchronisation

## 6.2.5. LES COULOIRS (DIT « SWIMLANES » EN ANGLAIS)

□ Ils permettent d'indiquer qui (de l'utilisateur ou du système) réalise les actions.



## 6.3. CONCLUSION

- ❑ Un diagramme d'activité est donc un bon complément à la fiche descriptive d'un cas d'utilisation.
- ❑ Si un cas d'utilisation contient de nombreux scénarios, le diagramme d'activité permet de donner une vision globale de l'ensemble des scénarios possibles.



## Chap. 7 : LE DIAGRAMME DE SEQUENCES

□ Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation ***Unified Modeling Language***.

## Chap. 7 : LE DIAGRAMME DE SEQUENCES

- ❑ **Les diagrammes de cas d'utilisation** modélisent à QUOI sert le système, en organisant les interactions possibles avec les acteurs.
- ❑ **Les diagrammes de classes** permettent de spécifier la structure et les liens entre les objets dont le système est composé : ils spécifient QUI sera à l'oeuvre dans le système pour réaliser les fonctionnalités décrites par les diagrammes de cas d'utilisation.

## Chap. 7 : LE DIAGRAMME DE SEQUENCES

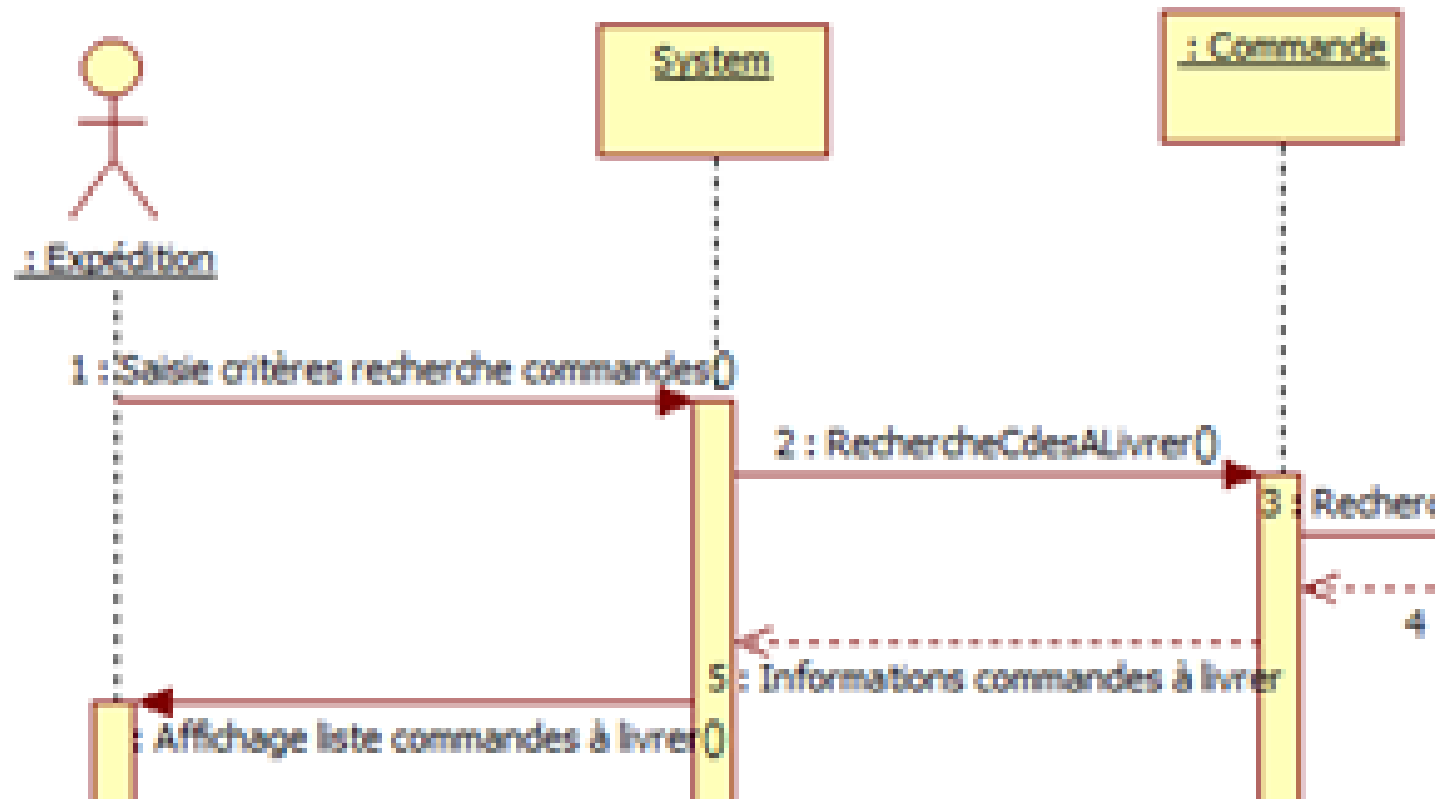
□ **Les *diagrammes de séquences*** permettent de décrire COMMENT les éléments du système interagissent entre eux et avec les acteurs :

- Les objets au coeur d'un système interagissent en s'échangeant des messages.
- Les acteurs interagissent avec le système au moyen d'IHM (Interfaces Homme-Machine).

## 7.1. L'UTILITÉ DU DIAGRAMME DE SÉQUENCE

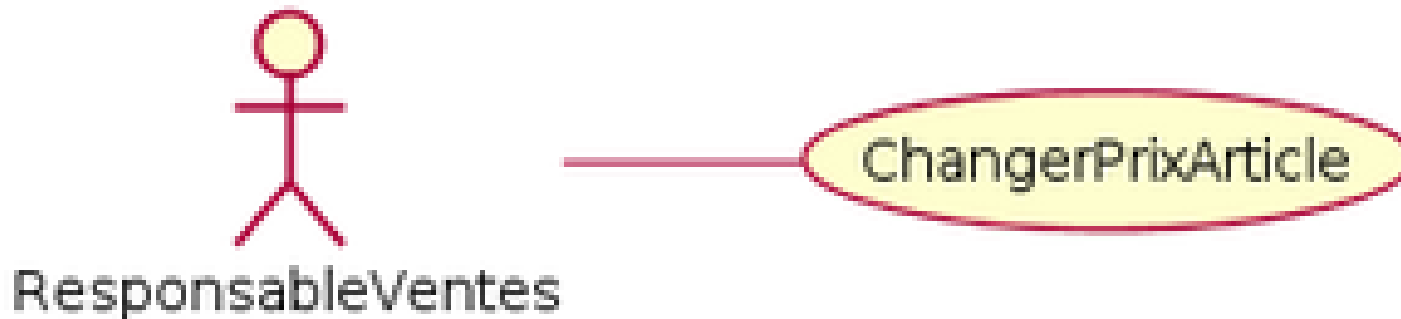
- ❑ Le diagramme de séquence permet de montrer les interactions d'objets dans le cadre d'un scénario d'un Diagramme des cas d'utilisation.
- ❑ Dans un souci de simplification, on représente l'acteur principal à gauche du diagramme, et les acteurs secondaires éventuels à droite du système.
- ❑ Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets.

## 7.1. L'UTILITÉ DU DIAGRAMME DE SÉQUENCE



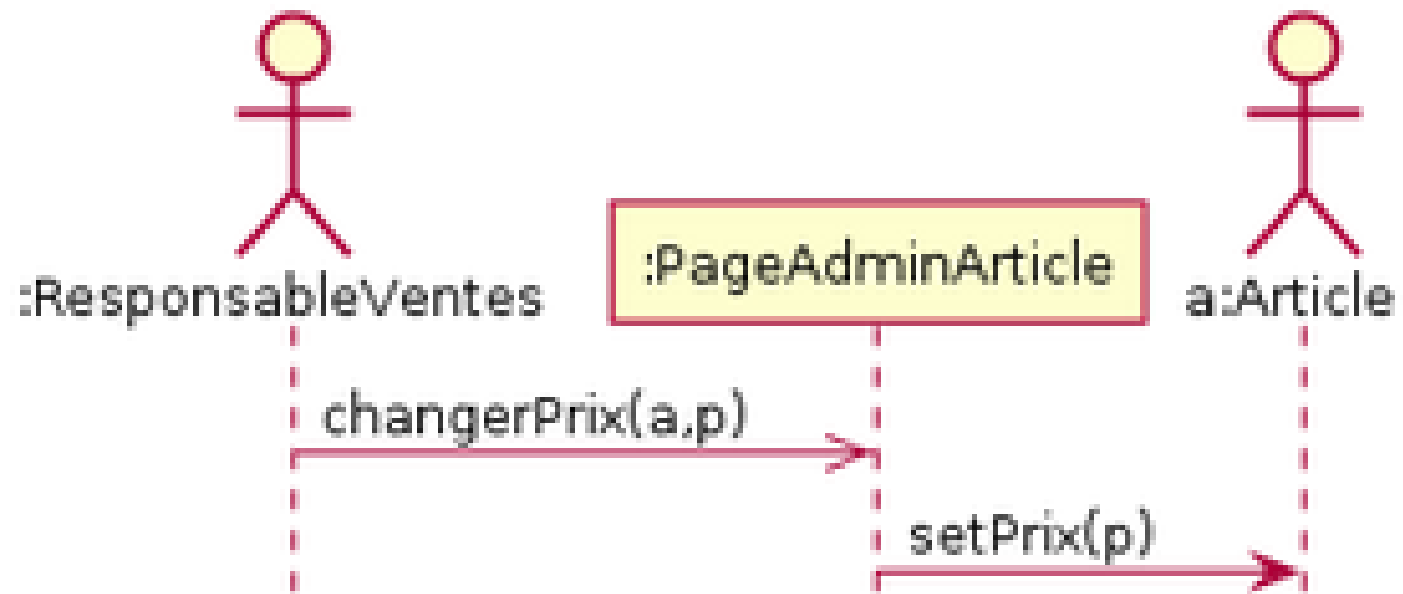
## 7.2. INTERACTIONS

- ❑ Pour être complètement spécifiée, une interaction doit être décrite dans plusieurs diagrammes UML :
- ❑ Cas d'utilisation



## 7.2. INTERACTIONS

### ☐ Séquences



## 7.2. INTERACTIONS

- ❑ Classes pour spécifier les opérations nécessaires

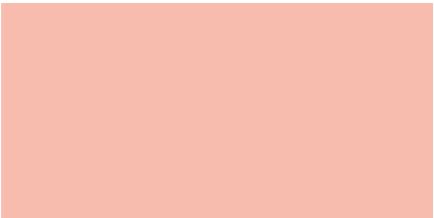




## 7.3. COMPOSANTS ET SYMBOLES ÉLÉMENTAIRES

- ❑ Pour comprendre ce qu'est un diagramme de séquence, vous devez connaître ses symboles et ses composants.
- ❑ Les diagrammes de séquence sont composés des icônes et des éléments suivants :

## 7.3. COMPOSANTS ET SYMBOLES ÉLÉMENTAIRES

Symbole	Nom	Description
	Symbole d'objet	Représente une classe ou un objet en langage UML. Le symbole objet montre comment un objet va se comporter dans le contexte du système. Les attributs de classe ne doivent pas être énumérés dans cette forme.

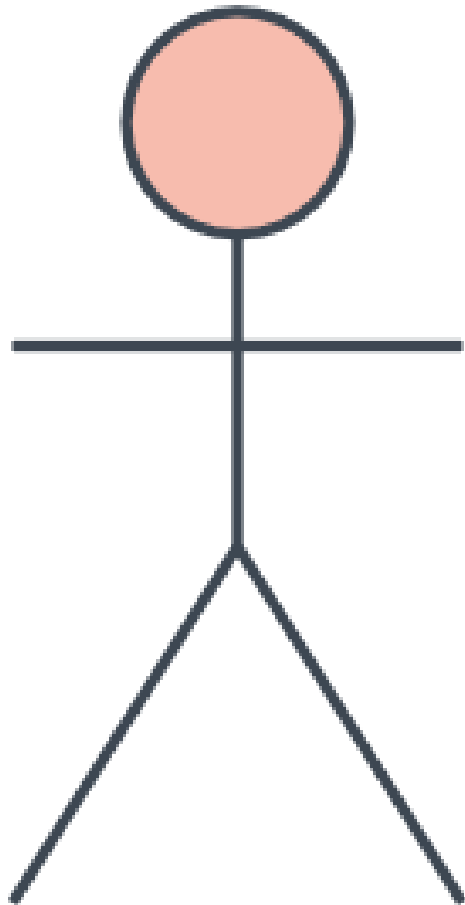
## 7.3. COMPOSANTS ET SYMBOLES ÉLÉMENTAIRES



boîte  
d'activation

Représente le temps  
nécessaire pour qu'un objet  
accomplisse une tâche. Plus la  
tâche nécessite de temps, plus  
la boîte d'activation est longue.

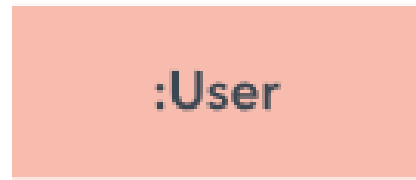
## 7.3. COMPOSANTS ET SYMBOLES ÉLÉMENTAIRES



Symbole  
d'acteur

Montre les entités qui interagissent avec le système ou qui sont extérieures à lui.

## 7.3. COMPOSANTS ET SYMBOLES ÉLÉMENTAIRES




Symbole de  
ligne de vie

Représente le passage du temps qui se prolonge vers le bas. Cette ligne verticale en pointillés montre les événements séquentiels affectant un objet au cours du processus schématisé. Les lignes de vie peuvent commencer par une forme rectangulaire avec un intitulé ou par un symbole d'acteur.

## 7.4. SYMBOLES DE MESSAGES COURANTS

- ❑ Utilisez les flèches et les symboles de messages suivants pour indiquer comment les informations sont transmises entre des objets.
- ❑ Ces symboles peuvent représenter le début et l'exécution d'une opération, ou l'envoi et la réception d'un signal.

## 7.4. SYMBOLES DE MESSAGES COURANTS

Symbole	Nom	Description
	Symbole de messages synchrones	Représentés par une ligne pleine terminée par une pointe de flèche pleine. On utilise ce symbole lorsqu'un expéditeur doit attendre une réponse à un message avant de continuer. Le diagramme doit montrer à la fois l'appel et la réponse.

## 7.4. SYMBOLES DE MESSAGES COURANTS



Symbole de  
messages  
asynchrones

Représentés par une ligne pleine terminée par une pointe de flèche. Les messages asynchrones ne nécessitent pas de réponse avant que l'expéditeur ne continue. Seul l'appel doit être inclus dans le diagramme.



## 7.4. SYMBOLES DE MESSAGES COURANTS



Symbole de  
messages de  
réponse

Représentés par une ligne en pointillés terminée par une pointe de flèche, ces messages sont des réponses aux appels.

## Chap. 8 : LE DIAGRAMME DE CLASSES

- ❑ Les diagrammes de classes sont l'un des types de diagrammes UML les plus utiles, car ils décrivent clairement la structure d'un système particulier en modélisant ses classes, ses attributs, ses opérations et les relations entre ses objets.
- ❑ Très utilisé par les ingénieurs logiciel pour documenter l'architecture des logiciels, les diagrammes de classes sont un type de diagramme de structure, car ils décrivent ce qui doit être présent dans le système modélisé.

## Chap. 8 : LE DIAGRAMME DE CLASSES

- ❑ Le diagramme de classes en UML (Modèle Conceptuel de Données, MCD en MERISE) est une représentation des besoins en matière de données pour un système d'information.
- ❑ Il met en évidence les classes (ou entités), leurs attributs, les relations (ou associations) et contraintes entre ces classes pour un domaine donné.

## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Classe (ou entité) :

- ❑ Objet concret ou abstrait du monde réel au sujet duquel une organisation est susceptible de conserver des données.
- ❑ Une entité concrète possède une existence physique : c'est le cas d'un client, d'un équipement, d'un produit par exemple.

## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Classe (ou entité) :

- ❑ Une entité abstraite a une existence conceptuelle, par exemple une transaction, un tarif, l'annulation d'un vol d'avion.
- ❑ Le client ***Jean Dupond*** est une entité, la commande ***COM0001*** est aussi une entité, la première concrète, l'autre abstraite.

## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Classe (ou entité) :

□ Bien qu'une commande puisse prendre la forme d'un document papier, ce qui intéresse le modélisateur c'est l'aspect conceptuel de la transaction car elle peut ne pas exister sur papier. Elle sera donc considérée avant tout comme un objet abstrait.

## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Classe (ou entité) :

- ❑ Toute entité possède des **propriétés**, appelés **attributs**, et l'ensemble des entités qui ont les mêmes attributs est représenté graphiquement par une entité type, soit un rectangle comportant dans la case au haut le nom de l'entité type et dans la case du bas la liste des attributs de l'entité type.
- ❑ Il s'agit du moule dans lequel toute entité de ce type est formée.

## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Attribut

- ❑ Donnée élémentaire qui sert à caractériser une propriété des entités et des associations dans un modèle conceptuel de données (Attribute).
- ❑ La figure 1-1 montre l'entité type Client qui décrit la structure commune pour l'ensemble des entités Client notamment les attributs que chaque entité de ce type possède : No client, Nom client, Prénom client et Adresse client.



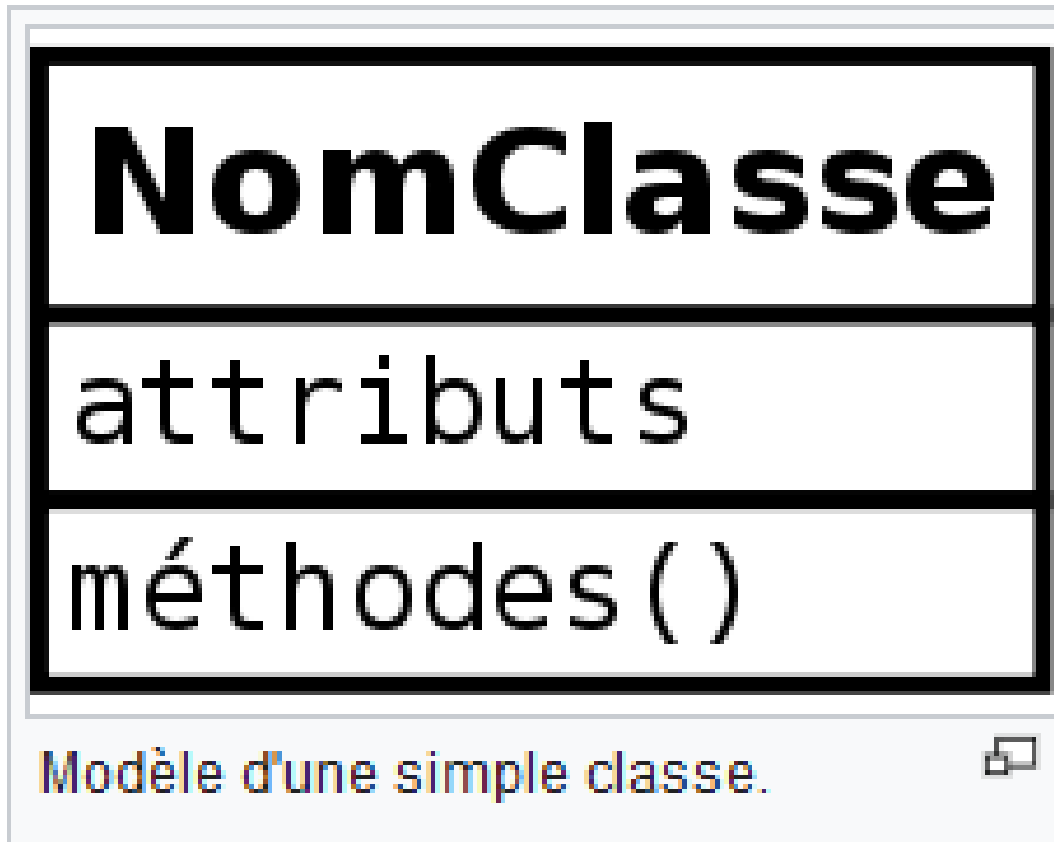
## 8.1. DÉFINITION DES CONCEPTS DE BASE

□ Une classe est représentée par un rectangle séparé en trois parties :

- la première partie contient le nom de la classe
- la seconde contient les attributs de la classe
- la dernière contient les méthodes de la classe

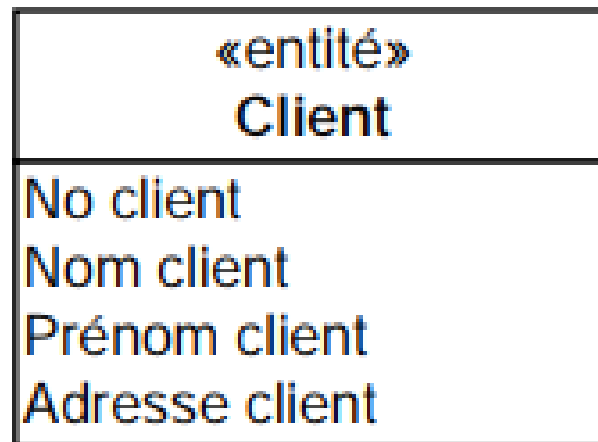
□ La seconde et la dernière représentent ***le comportement*** de la classe.

## 8.1. DÉFINITION DES CONCEPTS DE BASE



## 8.1. DÉFINITION DES CONCEPTS DE BASE

**FIGURE 1-1** Représentation graphique de l'entité type Client



## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Occurrence d'entité

- ❑ Élément particulier d'une entité type, identifiable de façon unique (Instance).
- ❑ Prenons l'exemple d'une entité **Client**, on dira que l'entité dont le **No client** est **CLI0002** est une occurrence de cette entité.

## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Association

- ❑ Lien sémantique qui existe entre deux entités ou plus.
- ❑ Elle représente souvent la mémoire d'un événement qui a permis d'établir un lien logique entre ces entités (Relationship).

## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Association

- ❑ La figure 1-2 présente un modèle où les entités **Client** et **Commande** sont liées par une association portant le nom **Effectue**.
- ❑ On notera que l'association est identifiée à l'aide d'un verbe d'action, ce qui permet de faire une lecture de l'association comme s'il s'agissait d'une phrase comportant un sujet, un verbe et un complément.

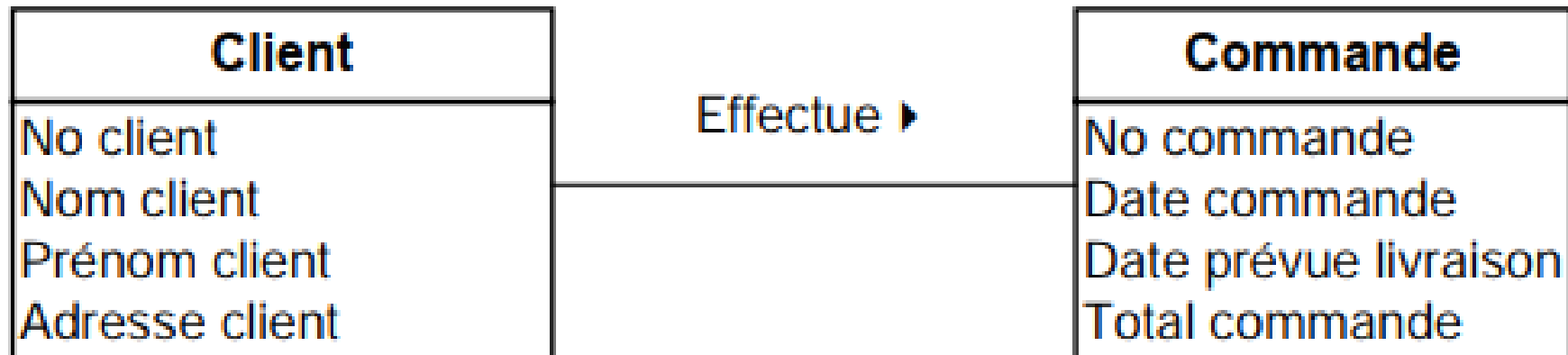
## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Association

- ❑ Cette association est dite binaire car elle met en cause deux entités.
- ❑ Comme nous le verrons plus loin à la section portant sur les concepts avancés, il existe aussi des associations sur plus de deux entités, dites de degré supérieur, qui sont beaucoup moins fréquentes mais dont l'importance mérite que l'on s'y attarde.

## 8.1. DÉFINITION DES CONCEPTS DE BASE

**FIGURE 1-2** Association entre l'entité Client et l'entité Commande





## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Association

- ❑ Le modèle de la figure 1-2 est une représentation abstraite qui, par analogie, pourrait correspondre intuitivement à deux cartons de fiches, le premier portant le nom **Client** et l'autre le nom **Commande**.
- ❑ Certaines fiches du carton **Client** sont reliées à des fiches du carton **Commande** par une ficelle.

## 8.1. DÉFINITION DES CONCEPTS DE BASE

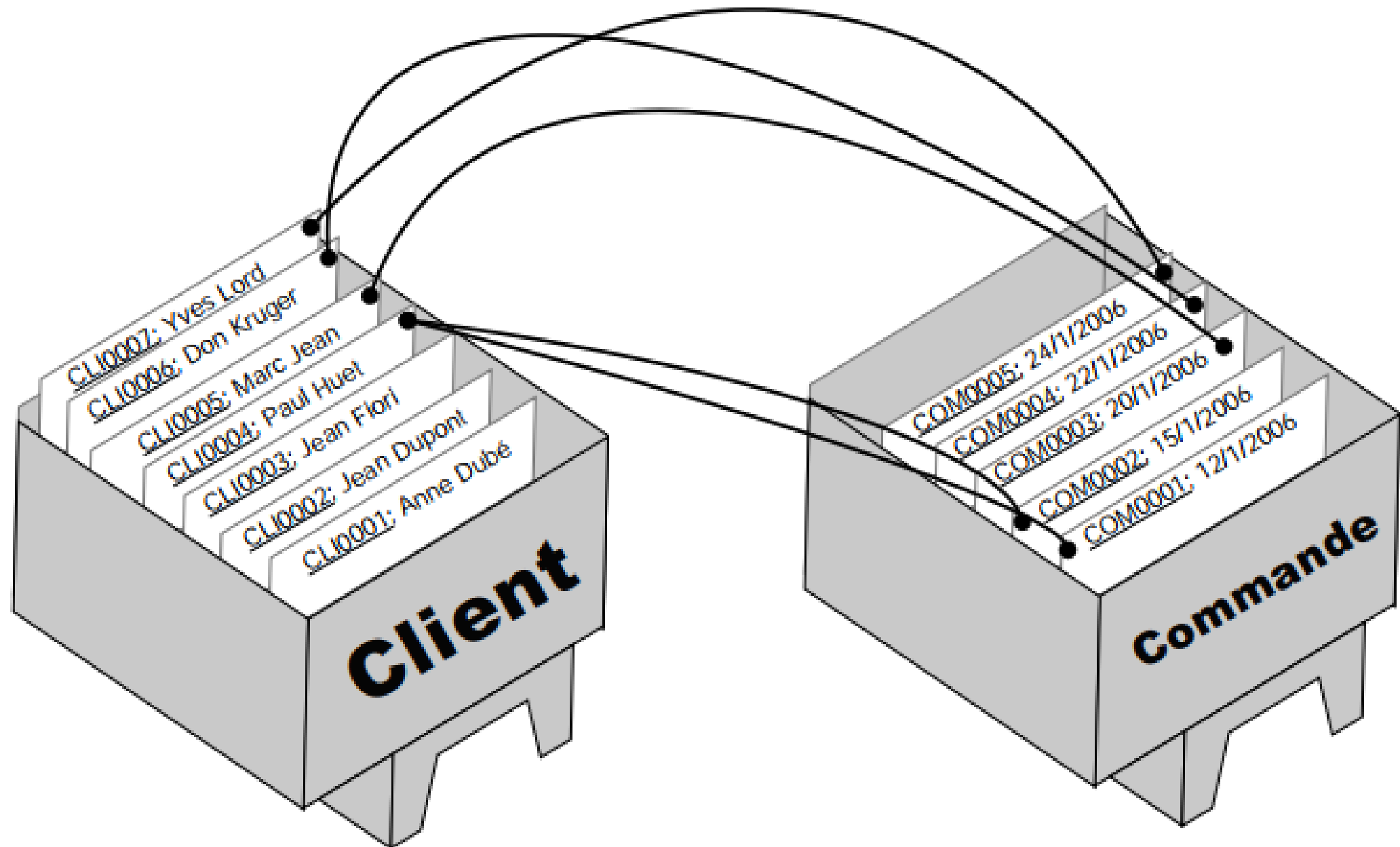
### Association

- ❑ Dans cette analogie les deux cartons correspondent aux deux entités.
- ❑ Les fiches représentent des occurrences de chacune des entités.
- ❑ Les inscriptions sur les fiches sont les valeurs des attributs pour une occurrence d'entité.
- ❑ Enfin, on assimile les ficelles aux associations.

## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Association

- ❑ Cette analogie est illustrée à la figure 1-3 où, pour des raisons de simplification, les valeurs pour les attributs des entités ne sont pas toutes montrées.
- ❑ Seules les valeurs des trois premiers attributs le sont : ***No client***, ***Prénom client*** et ***Nom client***.

**FIGURE 1-3****Analogie pour le modèle conceptuel de la figure 1-2**

## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Association

- ❑ En consultant la figure 1-3, vous comprendrez toute l'importance d'une association dans un modèle conceptuel.
- ❑ Sur le plan sémantique, la présence d'une association entre deux occurrences d'entités (une ficelle) nous assure qu'à partir d'une occurrence il est possible d'avoir accès à l'autre occurrence associée en suivant cette ficelle et vice versa.

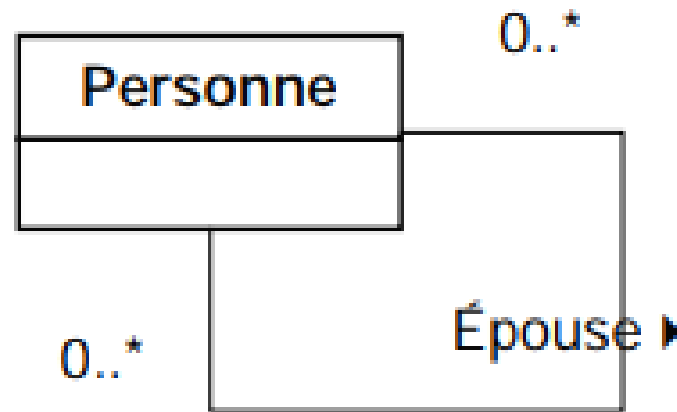
## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Association

- ❑ Une association peut mettre en cause des occurrences de la même entité.
- ❑ On parle alors *d'association réflexive*.
- ❑ Pour illustrer la notion de mariage entre deux personnes, on peut faire appel à une association réflexive comme le montre la figure 1-4.

## 8.1. DÉFINITION DES CONCEPTS DE BASE

**FIGURE 1-4** Une personne est associée à une autre personne lorsqu'elle l'épouse



## 8.1. DÉFINITION DES CONCEPTS DE BASE

### Association

□ Le modèle illustre le fait qu'un carton comportant des fiches sur des personnes pourrait comporter des ficelles qui relient des fiches placées dans le même carton pour mémoriser un événement, soit ici le mariage entre deux personnes.



## 8.2. CONTRAINTES SUR LES ATTRIBUTS ET LES ASSOCIATIONS

- ❑ L'analogie des cartons à fiches met aussi en évidence la nécessité d'identifier chaque fiche dans un carton, donc chaque occurrence d'une entité, pour pouvoir facilement les repérer.
- ❑ Pour ce faire, il doit exister un attribut, ou un groupe d'attributs, qui fournit une identification unique à chaque occurrence d'une entité.
- ❑ Cet attribut ou ce groupe d'attributs est appelé **identifiant** dans le formalisme entité-association.

## 8.2.1. IDENTIFIANT

- ❑ L'identifiant d'une entité est un attribut ou groupe d'attributs permettant d'identifier chaque occurrence de cette entité.
- ❑ Pour ce qui est du modèle conceptuel de la figure 1-2, on peut déduire sans risque de se tromper que l'attribut **No client** serait un bon choix comme identifiant de l'entité **Client** et que le **No commande** pourrait jouer ce rôle pour l'entité **Commande**.

## 8.2.1. IDENTIFIANT

- ❑ On en conclut que chaque numéro de client et chaque numéro de commande est unique, ce qui permet de repérer une fiche dans le bac correspondant si son identifiant est connu.
- ❑ L'importance de l'identifiant implique que chaque fiche possède une valeur unique pour l'identifiant.
- ❑ La plupart des notations utilisent le soulignement pour marquer un identifiant dans le diagramme.

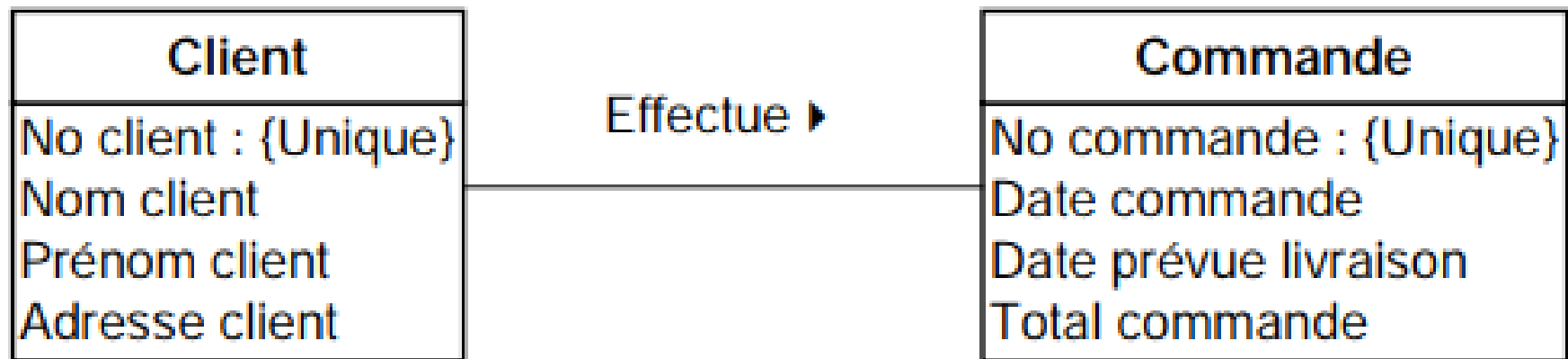
## 8.2.1. IDENTIFIANT

- ❑ Dans la notation UML cependant, il en va autrement.
- ❑ L'attribut est marqué d'une contrainte, placée à la suite du nom de l'attribut, qui indique que la valeur de cet attribut est unique, c'est-à-dire qu'elle ne peut être prise par deux occurrences de la même entité.
- ❑ En UML une contrainte est inscrite entre accolades comme le montre la figure 1-5.

## 8.2.1. IDENTIFIANT

❑ Ici la contrainte {Unique} marque l'attribut agissant comme identifiant.

**FIGURE 1-5** M odèle conceptuel dont les identifiants sont marqués



## 8.2.1. IDENTIFIANT

- ❑ Il s'agit d'un premier type de contrainte applicable à un attribut qui doit obligatoirement avoir une valeur pour chaque occurrence de l'entité.
- ❑ Par ailleurs, les valeurs doivent être toutes différentes d'une occurrence à l'autre.
- ❑ La présence d'une contrainte {Unique} sur un attribut a pour corollaire que la valeur de cet attribut ne peut être inconnu, ou nulle.

## 8.2.1. IDENTIFIANT

- ❑ Cette contrainte n'est pas expressément exprimée dans le modèle sous la forme {Unique ; Non nul} mais elle est implicite pour l'attribut choisi comme identifiant.
- ❑ La présence d'un identifiant marqué de la contrainte {Unique} pour une entité représente une contrainte d'intégrité d'entité.

## 8.2.1. IDENTIFIANT

- ❑ Lorsqu'un identifiant est formé d'un seul attribut on le qualifie ***d'identifiant simple***.
- ❑ S'il s'avère qu'un identifiant doit comporter plusieurs attributs, une contrainte {Unique} sera placée avant tous les attributs de l'entité indiquant en cela quels sont les attributs qui forment l'identifiant.
- ❑ On parle alors d'un ***identifiant composé***.

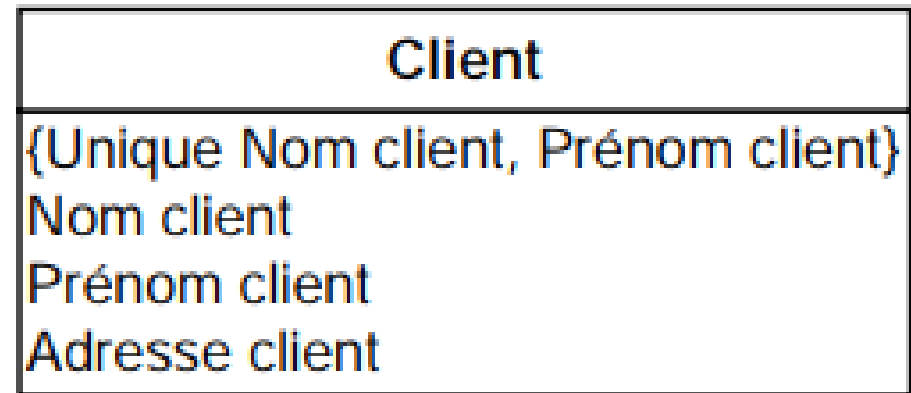


## 8.2.1. IDENTIFIANT

- ❑ La figure 1-6 montre une entité avec identifiant composé.
- ❑ Comme **No client** n'est pas présent dans l'entité **Client**, la combinaison du **Nom** et du **Prénom du client** a été choisi comme **identifiant** dans la mesure où on peut être assuré que deux clients n'auraient jamais à la fois le même nom et le même prénom.
- ❑ Si on ne peut être assuré de cela, il vaut mieux envisager un nouvel attribut, **No client**, comme le montre l'entité **Client** de la figure 1-5.

## 8.2.1. IDENTIFIANT

**FIGURE 1-6** Modèle conceptuel avec identifiant composé



## 8.2.2. RÈGLE D'IDENTITÉ

- ❑ Toute entité présente dans un modèle conceptuel de données doit comporter obligatoirement un identifiant.
- ❑ Chaque occurrence de l'entité doit posséder une valeur pour cet attribut.
- ❑ La valeur de l'attribut identifiant devra être stable, c'est-à-dire qu'au cours de la vie d'une occurrence de l'entité cette valeur ne pourra changer.

## 8.2.2. RÈGLE D'IDENTITÉ

- ❑ Deux occurrences de l'entité ne pourraient avoir la même valeur pour leur identifiant.
- ❑ La même valeur accordée plus d'une fois à un identifiant représente un doublon.
- ❑ Les doublons sont inacceptables pour un identifiant.

## 8.2.2. RÈGLE D'IDENTITÉ

- ❑ La règle d'identité prescrit qu'un identifiant est obligatoire, stable et sans doublon.
- ❑ Dans le cas de l'entité **Client**, au delà du fait que le choix d'un identifiant composé n'est pas recommandé, l'utilisation du **nom** et du **prénom** du client ne peut garantir la stabilité de l'identifiant, car une personne pourrait changer de nom.

## 8.2.2. RÈGLE D'IDENTITÉ

- ❑ Un autre type de contrainte doit être ajouté au modèle conceptuel dès lors que les identifiants des entités associées sont établis.
- ❑ Il s'agit des **contraintes de multiplicité** qui précisent à la fois que la participation d'une entité à l'association est obligatoire ou non, et le cas échéant, le nombre d'occurrences d'une association auxquelles elle peut participer.

### 8.2.3. MULTIPLICITÉ (ASSOCIATION BINAIRE)

□ Contrainte inscrite à chaque extrémité d'une association binaire comportant un couple de valeurs (minimum–maximum) qui établit, pour chaque entité de l'association, les nombres minimum et maximum d'occurrences de l'autre entité qui peuvent lui être associées.

### 8.2.3. MULTIPLICITÉ (ASSOCIATION BINAIRE)

□ Il est fondamental que le modélisateur et l'ensemble des utilisateurs du système aient une compréhension commune de ces règles pour que les contraintes de multiplicité soient valables et s'avèrent en conséquence pertinentes.



### 8.2.3. MULTIPLICITÉ (ASSOCIATION BINAIRE)

- ❑ Le tableau 1-1 donne les quatre combinaisons de base du couple de valeurs.
- ❑ Le premier chiffre (**0 ou 1**), indique qu'il s'agit d'une association optionnelle (**0**) ou d'une association obligatoire (**1**).
- ❑ Le deuxième chiffre indique le nombre maximum d'occurrences des associations auxquelles une entité participe : **une (1)** ou strictement plus d'une, c'est-à-dire **plusieurs (\*)**. L'astérisque représente en effet la valeur **plusieurs**.

## 8.2.3. MULTIPLICITÉ (ASSOCIATION BINAIRE)

**TABLEAU 1-1** Codification des multiplicités

Multiplicité UML	Signification
0..1	Au plus un
1..1 (ou 1)	Un seul
0..* (ou *)	Un nombre indéterminé
1..*	Au moins un

### 8.2.3. MULTIPLICITÉ (ASSOCIATION BINAIRE)

□ Si nous reprenons le modèle de la figure 1-4 et tentons d'établir les multiplicités de l'association binaire **Effective**, il suffit de consulter l'analogie de la figure 1-3 pour arriver à les déduire.

### 8.2.3. MULTIPLICITÉ (ASSOCIATION BINAIRE)

- ❑ Considérons les occurrences de **Client**, soit par analogie les fiches dans le carton **Client**.
- ❑ Nous notons que certains clients ne sont liés à aucune commande, c'est le cas d'**Anne Dubé**.
- ❑ L'association avec **Commande** est donc optionnelle (0) pour **Client**.

### 8.2.3. MULTIPLICITÉ (ASSOCIATION BINAIRE)

- ❑ Par ailleurs, certains clients sont liés à plusieurs commandes (\*), c'est le cas de **Paul Huet**.
- ❑ On en déduit donc la multiplicité **0..\*** qui sera placée sur l'association du côté de **Commande**.

### 8.2.3. MULTIPLICITÉ (ASSOCIATION BINAIRE)

- ❑ En consultant les occurrences de **Commande**, soit les fiches du carton **Commande**, on constate qu'elles sont toutes liées à un client.
- ❑ L'association avec **Client** est donc obligatoire (1).
- ❑ De plus une commande n'est jamais liée à plus d'un client (1).
- ❑ On en déduit la multiplicité **1..1** qui sera placée sur l'association du côté de **Client**, car une commande ne peut être liée qu'à un seul **Client**.

### 8.2.3. MULTIPLICITÉ (ASSOCIATION BINAIRE)

- ❑ Le modélisateur expérimenté ne fait pas nécessairement la réflexion sur les multiplicités à l'aide de l'analogie des cartons.
- ❑ Il va établir assez aisément, en analysant le fonctionnement de l'organisation, qu'un client peut effectuer un nombre indéterminé de commandes (**0..\***) et qu'une commande ne peut être liée qu'à un seul client (**1..1**).
- ❑ En toute logique, une commande ne peut en effet provenir de deux clients.

### 8.2.3. MULTIPLICITÉ (ASSOCIATION BINAIRE)

- ❑ La figure 1-7 montre un modèle conceptuel maintenant complet avec les deux contraintes de multiplicité placées de part et d'autre de l'association.
- ❑ Toute association binaire comporte deux multiplicités car une association se lit dans les deux directions avec des multiplicités qui peuvent être différentes.



### 8.2.3. MULTIPLICITÉ (ASSOCIATION BINAIRE)

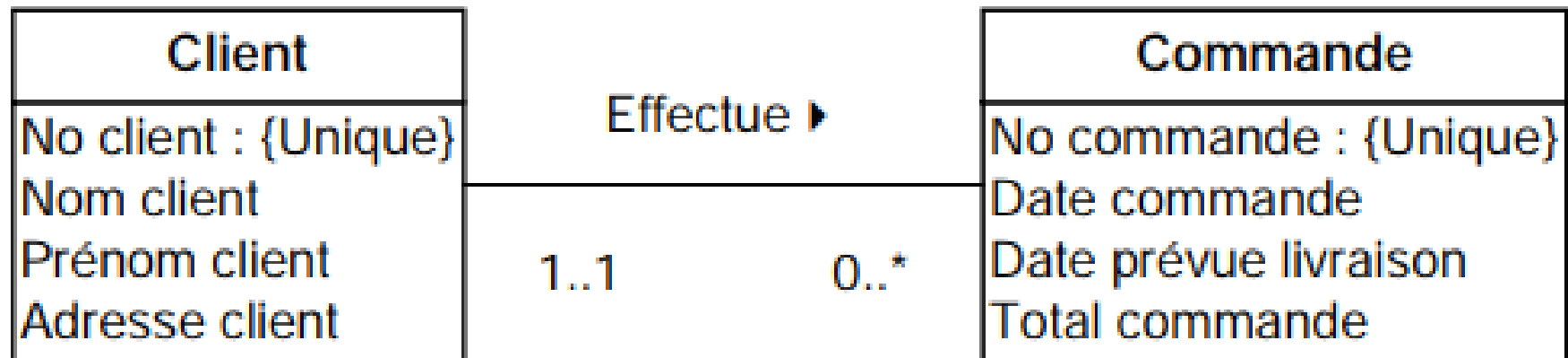
□ Dans le modèle de la figure 1-7 le modélisateur a été tenu de répondre à deux questions qui ont conduit à deux réponses différentes :

1. « Un client est-il obligatoirement lié à une commande et à combien au maximum ? »

2. « Une commande est-elle obligatoirement liée à un client et à combien au maximum ? »

## 8.2.3. MULTIPLICITÉ (ASSOCIATION BINAIRE)

**FIGURE 1-7** Modèle conceptuel avec multiplicités



## Partie 2 : CRÉATION ET MAINTENANCE D'UNE BASE DE DONNÉES À PARTIR D'UNE MODÉLISATION UML

## Chap. 9 : INTRODUCTION AUX BASES DE DONNEES

- ❑ Vous avez de nombreuses données à traiter (via une application par exemple) et vous voulez les organiser correctement, avec un outil adapté ?
- ❑ Les bases de données ont été créées pour vous !
- ❑ Les bases de données sont utilisées par de nombreuses entreprises dans toutes les industries.

## Chap. 9 : INTRODUCTION AUX BASES DE DONNEES

- ❑ Les secteurs utilisant les bases de données sont multiples (finance, assurances, administration publique, médias, écoles...) et à des fins diverses (inventaires, gestion de ressources humaines, contrôle de production, comptabilité, facturation...).
- ❑ En somme, les bases de données apparaissent comme des outils désormais indispensables pour la généralisation des informations ainsi que la facilité, la rapidité et la simultanéité de leur utilisation.

## Chap. 9 : INTRODUCTION AUX BASES DE DONNEES

□ Ces sortes de conteneurs de données sont une source inépuisable d'information grâce aux outils informatiques permettant leur utilisation.

## 9.1. DEFINITION

- ❑ Une base de données est un ensemble structuré et organisé permettant le stockage de grandes quantités d'informations afin d'en faciliter l'exploitation (ajout, mise à jour, recherche de données).
- ❑ Ces informations sont en rapport avec une activité donnée et peuvent être utilisées par des programmes ou des utilisateurs communs, d'où la nécessité de leur mise en commun.

## 9.1. DEFINITION

- ❑ Peu importe le support utilisé pour rassembler et stocker les données (papier, fichiers, etc.), dès lors que des données sont rassemblées et stockées d'une manière organisée dans un but spécifique, on parle de base de données.
- ❑ Une base de données informatisée est une base de données accessible par un matériel informatique (ordinateur, etc.).



## 9.1. DEFINITION

- ❑ La base de données est donc la pièce centrale des dispositifs informatiques servant à la collecte, au stockage et à l'utilisation des informations recueillies.
- ❑ Ces dispositifs comportent un système de gestion de base de données (SGBD) qui est une sorte de logiciel moteur pour l'accès et la manipulation de la base de données.
- ❑ L'intérêt d'une base de données est de permettre la consultation d'un grand nombre de données à des utilisateurs qui s'en sont vu accordé le droit.

## 9.1. DEFINITION

□ L'avantage majeur d'une base de données hormis de regrouper un grand nombre d'informations demeure ***la possibilité d'y accéder par plusieurs utilisateurs simultanément.***

## 9.2. GESTION DES BASES DE DONNEES (LES SGBD)

- ❑ La gestion et l'accès à une base de données sont assurés par un ensemble de programmes qui constituent le Système de gestion de base de données (SGBD).
- ❑ Les données sont décrites sous la forme d'un modèle, grâce à un « **Langage de Définition des Données (LDD)** » ou « **Data Definition Language (DDL)** » en anglais.
- ❑ Cette description est appelée *schéma*.

## 9.2. GESTION DES BASES DE DONNEES (LES SGBD)

- ❑ Une fois la base de données spécifiée, on peut y *insérer des données*, les *récupérer*, les *modifier* et les *supprimer* : c'est ce qu'on appelle *manipuler les données*.
- ❑ Les données peuvent être manipulées par un « *Langage de Manipulation des Données (LMD)* » ou « *Data Manipulation Language (DML)* » en anglais.

## 9.2. GESTION DES BASES DE DONNEES (LES SGBD)

- ❑ Le langage permettant de contrôler l'accès aux données d'une base de données est le « **langage de contrôle de données (LCD)** » ou « **Data Control Language (DCL)** » en anglais.
- ❑ Celui utilisé pour le contrôle transactionnel dans une base de données, c'est-à-dire les caractéristiques des transactions, la validation et l'annulation des modifications est le « **langage de contrôle des transactions (LCT)** » ou « **Transaction Control Language (TCL)** » en anglais.

## 9.2. GESTION DES BASES DE DONNEES (LES SGBD)

- ❑ Pour permettre une utilisation optimale d'une base de données, il faut mettre en place un système de gestion, d'où l'intérêt des **SGBD** (en anglais **DBMS** : *Database Management System*).
- ❑ Le système de gestion de base de données est une suite de programmes qui manipule la structure de la base de données et dirige l'accès aux données qui y sont stockées.

## 9.2. GESTION DES BASES DE DONNEES (LES SGBD)

- ❑ Une base de données est composée d'une collection de fichiers ; on y accède par le SGBD qui reçoit des demandes de manipulation du contenu et effectue les opérations nécessaires sur les fichiers.
- ❑ Il cache la complexité des opérations et offre une vue synthétique sur le contenu.
- ❑ Le SGBD permet à plusieurs usagers de manipuler simultanément le contenu, et peut offrir différentes vues sur un même ensemble de données.

## 9.2. GESTION DES BASES DE DONNEES (LES SGBD)

- ❑ Pour permettre une utilisation optimale d'une base de données, il faut donc mettre en place un système de gestion, d'où l'intérêt des SGBD.
- ❑ Le SGBD est donc un ensemble de services permettant :
  - L'accès aux données de façon simple ;
  - D'autoriser l'accès aux informations à de multiples utilisateurs ;



## 9.2. GESTION DES BASES DE DONNEES (LES SGBD)

- La manipulation des données présentes dans la base (insertion, suppression, modification, etc.).

## 9.2. GESTION DES BASES DE DONNEES (LES SGBD)

- ❑ Un SGBD héberge généralement plusieurs bases de données, qui sont destinées à des logiciels ou des thématiques différents.
- ❑ Actuellement, la plupart des SGBD fonctionnent selon un mode **client/serveur**.
- ❑ Le **serveur** (sous-entendu la machine qui stocke les données) reçoit des requêtes de plusieurs **clients** et ceci de manière concurrente.
- ❑ Le **serveur** analyse la requête, la traite et retourne le résultat au **client**.

## 9.2. GESTION DES BASES DE DONNEES (LES SGBD)

- ❑ C'est-à-dire que la base de données se trouve sur un « **serveur** » qui ne sert qu'à ça, et pour interagir avec cette base de données, il faut utiliser un logiciel « **client** » qui va interroger le serveur et transmettre la réponse que le serveur lui aura donnée.
- ❑ Le serveur peut être installé sur une machine différente du client ; c'est souvent le cas lorsque les bases de données sont importantes ; ce n'est cependant pas obligatoire.

## 9.3. MODELE DE BASES DE DONNEES

 Recherches...

## 9.4. OBJECTIFS DES SGBD

- ❑ Des objectifs principaux ont été fixés aux SGBD dès l'origine de ceux-ci, et ce, afin de résoudre les problèmes causés par la démarche classique.
- ❑ Ces objectifs sont les suivants :

## 9.4. OBJECTIFS DES SGBD

### 1. Indépendance physique :

- ❑ La façon dont les données sont définies doit être indépendante des structures de stockage utilisées.

## 9.4. OBJECTIFS DES SGBD

### 2. Indépendance logique :

- ❑ Un même ensemble de données peut être vu différemment par des utilisateurs différents. Toutes ces visions personnelles des données doivent être intégrées dans une vision globale.

## 9.4. OBJECTIFS DES SGBD

### 3. Accès aux données :

- ❑ L'accès aux données se fait par l'intermédiaire d'un Langage de Manipulation de Données (LMD).
- ❑ Il est crucial que ce langage permette d'obtenir des réponses aux requêtes en un temps « raisonnable ».



## 9.4. OBJECTIFS DES SGBD

### 3. Accès aux données :

- ❑ Le LMD doit donc être optimisé, minimiser le nombre d'accès disques, et tout cela de façon totalement transparente pour l'utilisateur.

## 9.4. OBJECTIFS DES SGBD

### 4. Administration centralisée des données (intégration) :

- ☐ Toutes les données doivent être centralisées dans un réservoir unique commun à toutes les applications.
- ☐ En effet, des visions différentes des données (entre autres) se résolvent plus facilement si les données sont administrées de façon centralisée.

## 9.4. OBJECTIFS DES SGBD

### 5. Non-redondance des données :

- ❑ Afin d'éviter les problèmes lors des mises à jour, chaque donnée ne doit être présente qu'une seule fois dans la base.

## 9.4. OBJECTIFS DES SGBD

### 6. Cohérence des données :

- ❑ Les données sont soumises à un certain nombre de contraintes d'intégrité qui définissent un état cohérent de la base.
- ❑ Elles doivent pouvoir être exprimées simplement et vérifiées automatiquement à chaque insertion, modification ou suppression des données.

## 9.4. OBJECTIFS DES SGBD

### 6. Cohérence des données :

- ☐ Les contraintes d'intégrité sont décrites dans le Langage de Description de Données (LDD).

## 9.4. OBJECTIFS DES SGBD

### 7. Partage des données :

- ❑ Il s'agit de permettre à plusieurs utilisateurs d'accéder aux mêmes données au même moment de manière transparente.

## 9.4. OBJECTIFS DES SGBD

### 7. Partage des données :

□ Si ce problème est simple à résoudre quand il s'agit uniquement d'interrogations, cela ne l'est plus quand il s'agit de modifications dans un contexte multiutilisateur, car il faut : permettre à deux (ou plus) utilisateurs de modifier la même donnée « en même temps » et assurer un résultat d'interrogation cohérent pour un utilisateur consultant une table pendant qu'un autre la modifie.

## 9.4. OBJECTIFS DES SGBD

### 8. Sécurité des données :

- ☐ Les données doivent pouvoir être protégées contre les accès non autorisés.
- ☐ Pour cela, il faut pouvoir associer à chaque utilisateur des droits d'accès aux données.



## 9.4. OBJECTIFS DES SGBD

### 9. Résistance aux pannes :

- ❑ Que se passe-t-il si une panne survient au milieu d'une modification, si certains fichiers contenant les données deviennent illisibles ? Il faut pouvoir récupérer une base dans un état « sain ».
- ❑ Ainsi, après une panne intervenant au milieu d'une modification deux solutions sont possibles : soit récupérer les données dans l'état dans lequel elles étaient avant la modification, soit terminer l'opération interrompue.

## 9.5. LES SGBD LES PLUS CONNUS ET UTILISES

### Exercice de maison

- ❑ Recherches des principaux SGBD, les avantages et les inconvénients de chaque SGBD.

## 9.6. LE LANGAGE SQL

- ❑ Le **SQL** (***Structured Query Language***) est un langage informatique qui permet d'interagir avec des bases de données.
- ❑ C'est le langage pour base de données le plus répandu, et c'est bien sûr celui utilisé par **MySQL**.
- ❑ C'est donc le langage que nous allons utiliser pour dire au client MySQL d'effectuer des opérations sur la base de données stockée sur le serveur MySQL.

## 9.6. LE LANGAGE SQL

- ❑ Il a été créé dans les années **1970** et c'est devenu standard en **1986** (pour la norme ANSI - **1987** en ce qui concerne la norme ISO).
- ❑ Il est encore régulièrement amélioré.

## 9.7. PRÉSENTATION SUCCINCTE DE MYSQL

- ❑ MySQL est donc un Système de Gestion de Bases de Données Relationnelles, qui utilise le langage SQL.
- ❑ C'est un des SGBDR les plus utilisés.
- ❑ Sa popularité est due en grande partie au fait qu'il s'agit d'un logiciel **Open Source**, ce qui signifie que son code source est librement disponible et que quiconque qui en ressent l'envie et/ou le besoin peut modifier MySQL pour l'améliorer ou l'adapter à ses besoins.

## 9.7. PRÉSENTATION SUCCINCTE DE MYSQL

- ❑ Une version gratuite de MySQL est par conséquent disponible.
- ❑ À noter qu'une version commerciale payante existe également.



## 9.8. DESCRIPTION DE LA STRUCTURE D'UNE BASE DE DONNEES

□ De manière simple, une **base de données** est un ensemble de « **tables** » et chaque table est constituée de « **colonnes** ».

## 9.8. DESCRIPTION DE LA STRUCTURE D'UNE BASE DE DONNEES

colonne de la table



table "animal"

id	espece	sexe	date_naissance	nom	commentaires
1	chien	M	2010-04-05 13:43:00	Rox	Mordille beaucoup
2	chat	NULL	2010-03-24 02:23:00	Roucky	NULL
3	chat	F	2010-09-13 15:02:00	Schtroumpfette	NULL
4	tortue	F	2009-08-03 05:12:00	NULL	NULL
5	chat	NULL	2010-10-03 16:44:00	Choupi	Né sans oreille gauche
6	tortue	F	2009-06-13 08:17:00	Bobosse	Carapace bizarre



occurrence de la table



## 9.8. MISE EN PLACE DE L'ENVIRONNEMENT DE DEVELOPPEMENT

### Pratique

- ❑ Installation des outils suivants :
- ❑ **Wamp** (ou Xampp) : Serveur local (**Apache**) + Serveurs de base de données (**MySQL** et **MariaDB**) + outils d'administration de bases de données (**phpMyAdmin**) + etc.

## Chap. 10 : TRANSFORMATION DU DIAGRAMME DE CLASSES EN MODÈLE RELATIONNEL

## 10.1. RÈGLE 1 : L'ASSOCIATION UN À UN (ONE-TO-ONE)

### Règle 1 : L'association un à un (one-to-one)

- ❑ L'association **un à un (one-to-one)** est une association qui possède **1** comme cardinalité **maximum** de part et d'autre de celle-ci.
- ❑ En d'autres termes, il y a la présence de la cardinalité **(?..1)** des deux côtés de l'association.

## 10.1. RÈGLE 1 : L'ASSOCIATION UN À UN (ONE-TO-ONE)

### Règle 1 : L'association un à un (one-to-one)

1. Chaque classe se transforme en une table ;
2. Chaque attribut de classe se transforme en un champs de table ;
3. On fait le choix de migrer l'attribut identifiant d'une classe vers l'autre classe (en général on prend le sens privilégié dans le domaine fonctionnel).

## 10.2. RÈGLE 2 : L'ASSOCIATION UN À PLUSIEURS (ONE-TO-MANY)

### Règle 2 : L'association un à plusieurs (one-to-many)

- ❑ L'association *un à plusieurs* (*one-to-many*) ou *plusieurs à un* (*many-to-one*) est une association qui possède **1** comme cardinalité maximum d'un côté et une cardinalité maximum supérieure à **1** (cardinalité multiple) de l'autre côté de celle-ci.
- ❑ En d'autres termes, il y a la présence de la cardinalité (**?..1**) d'un côté de l'association et de la cardinalité (**?..\***) de l'autre côté.

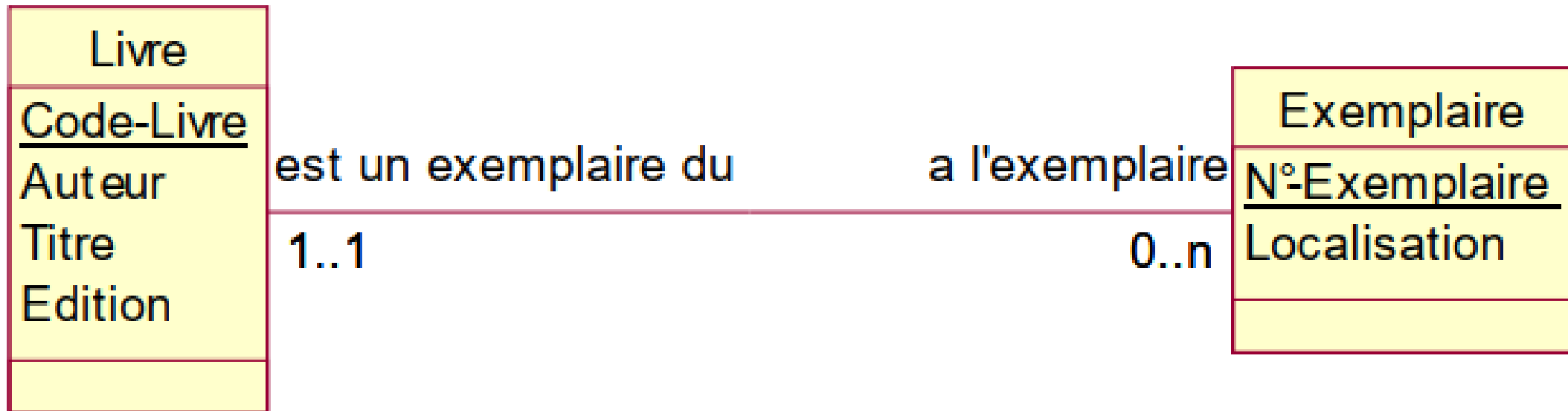
## 10.2. RÈGLE 2 : L'ASSOCIATION UN À PLUSIEURS (ONE-TO-MANY)

### Règle 2 : L'association un à plusieurs (one-to-many)

1. Chaque classe se transforme en une table ;
2. Chaque attribut de classe se transforme en un champs de table ;
3. L'identifiant de la classe qui est associée à la cardinalité (**?..1**) devient le clé étrangère de la classe ayant la cardinalité (**?..\***).

## 10.2. RÈGLE 2 : L'ASSOCIATION UN À PLUSIEURS (ONE-TO-MANY)

**Règle 2 : L'association un à plusieurs (one-to-many)**



## 10.2. RÈGLE 2 : L'ASSOCIATION UN À PLUSIEURS (ONE-TO-MANY)

### Règle 2 : L'association un à plusieurs (one-to-many)

□ Dans cet exemple, la clé primaire de l'entité **Livre** devient clé étrangère de l'entité **Exemplaire**.



## 10.3. RÈGLE 3 : L'ASSOCIATION PLUSIEURS À PLUSIEURS (MANY-TO-MANY)

### Règle 3 : L'association plusieurs à plusieurs (*many-to-many*)

- ❑ L'association *plusieurs à plusieurs* (*many-to-many*) est une association qui possède une **cardinalité maximum supérieure à 1** (cardinalité multiple) de part et d'autre de celle-ci.
- ❑ En d'autres termes, il y a la présence de la cardinalité (?..n) des deux côtés de l'association.

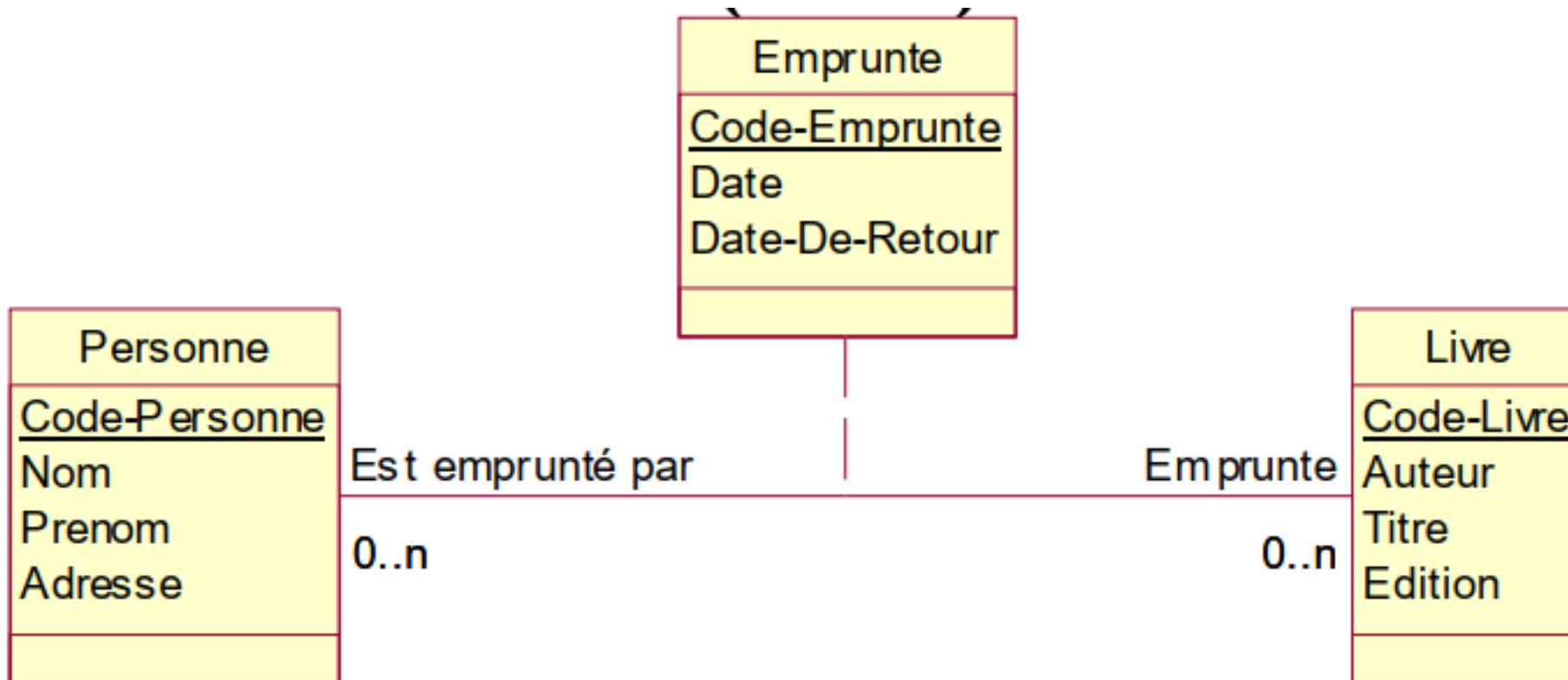
## 10.3. RÈGLE 3 : L'ASSOCIATION PLUSIEURS À PLUSIEURS (MANY-TO-MANY)

### Règle 3 : L'association plusieurs à plusieurs (many-to-many)

1. Chaque classe se transforme en une table ;
2. Chaque attribut de classe se transforme en un champs de table ;
3. L'association se transforme en une table. Cette table a comme champs l'identifiant de chacune des deux classes, plus d'éventuels autres attributs.

## 10.3. RÈGLE 3 : L'ASSOCIATION PLUSIEURS À PLUSIEURS (MANY-TO-MANY)

Règle 3 : L'association plusieurs à plusieurs (many-to-many)

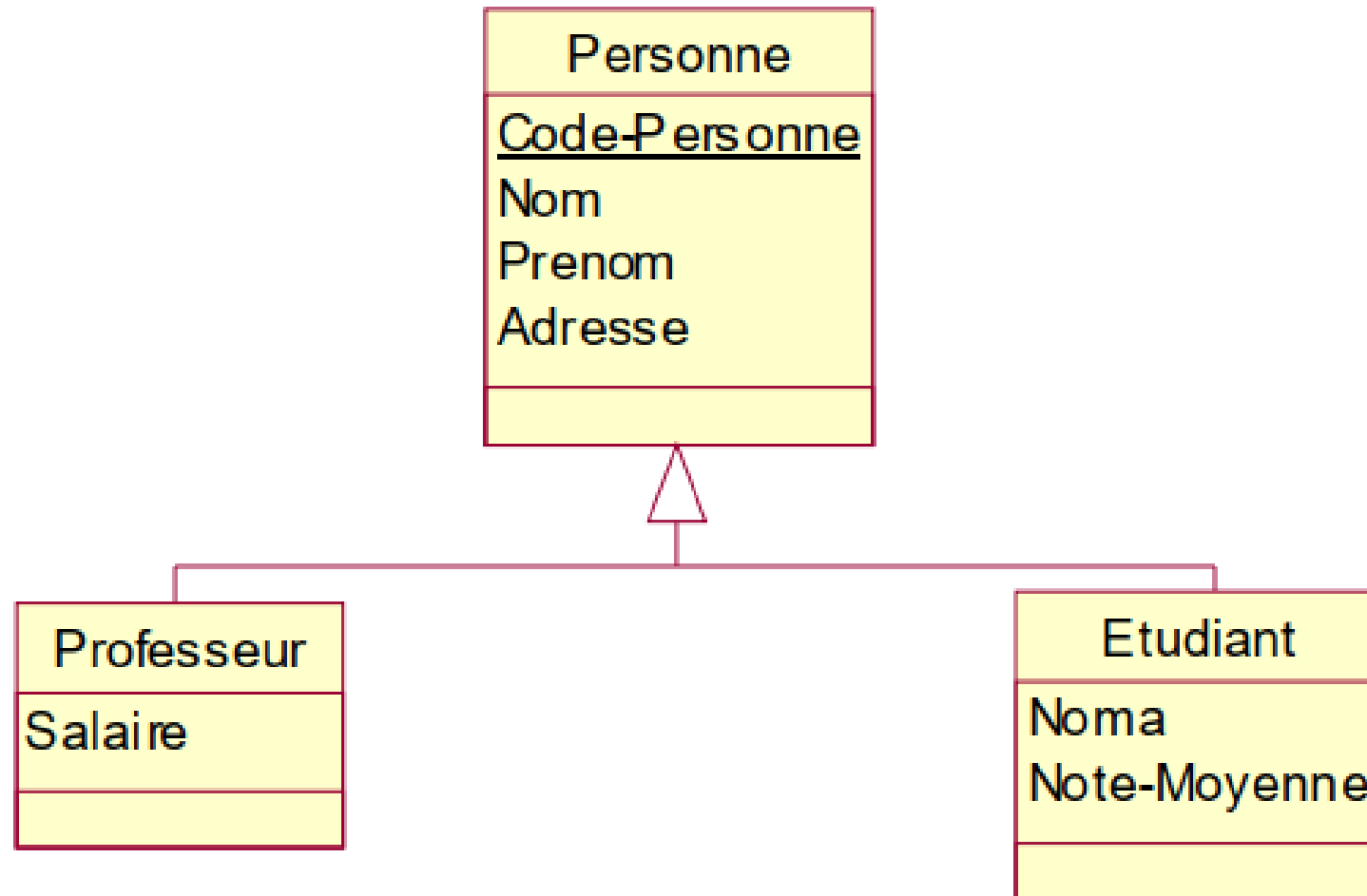


## 10.4. RÈGLE 4 : PRÉSENCE D'UNE GÉNÉRALISATION

**Règle 4 : présence d'une généralisation**

☐ Recherches...

## 10.4. RÈGLE 4 : PRÉSENCE D'UNE GÉNÉRALISATION



## Chap. 11 : INSTALLATION ET CONNEXION A MYSQL

❑ Maintenant qu'on sait à peu près de quoi on parle, il est temps d'installer MySQL sur nos ordinateurs, et de commencer à l'utiliser.

❑ Au programme de ce chapitre :

- Installation de MySQL ;
- Connexion et déconnexion au client MySQL ;
- Création d'un utilisateur ;
- Bases de la syntaxe du langage SQL.

## 11.1. UTILISATION DE MYSQL

- ❑ Il existe plusieurs manières d'utiliser MySQL.
- ❑ La première, est l'utilisation en ligne de commande.

## 11.1.1. LIGNE DE COMMANDE

- ❑ Il s'agit d'une fenêtre toute simple, dans laquelle toutes les instructions sont tapées à la main.
- ❑ Pas de bouton, pas de zone de saisie ; juste votre clavier.
- ❑ Vous allez le trouver dans **Démarrer > Tous les programmes > Accessoires.**



Administrator: Command Prompt

Microsoft Windows [Version 6.1.7601]  
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\>

## 11.1.2. INTERFACE GRAPHIQUE

- ❑ Si l'on ne veut pas utiliser la ligne de commande, on peut utiliser une interface graphique, qui permet d'exécuter pas mal de choses simples de manière intuitive sur une base de données.
- ❑ Comme interface graphique pour MySQL, on peut citer **MySQL Workbench**, **PhpMyAdmin** (souvent utilisé pour créer un site web en combinant MySQL et PHP) ou **MySQL Front** par exemple.

### 11.1.3. POURQUOI UTILISER LA LIGNE DE COMMANDE ?

☐ Deux raisons :

1. primo, parce qu'elle permet de maîtriser vraiment les commandes.

### 11.1.3. POURQUOI UTILISER LA LIGNE DE COMMANDE ?

2. ensuite, parce qu'il est fort probable que vous désiriez utiliser MySQL en combinaison avec un autre langage de programmation (si ce n'est pas votre but immédiat, ça viendra probablement un jour). Or, dans du code PHP (ou Java, ou Python, etc.), on ne va pas écrire "Ouvre PhpMyAdmin et clique sur le bon bouton pour que je puisse insérer une donnée dans la base". On va devoir écrire en dur les requêtes. Il faut donc que vous sachiez comment faire.

### 11.1.3. POURQUOI UTILISER LA LIGNE DE COMMANDE ?

- ❑ Je vous encourage vivement à commencer par utiliser la ligne de commande, ou au minimum à faire l'effort de décortiquer les requêtes que vous laisserez l'interface graphique construire pour vous.
- ❑ Ceci afin de pouvoir les écrire vous-mêmes le jour où vous en aurez besoin (ce jour viendra, je vous le prédis).

## 11.2. INSTALLATION DU LOGICIEL MYSQL (SUR WINDOWS)

☐ Pour télécharger MySQL, vous pouvez vous rendre sur le site suivant :

<http://dev.mysql.com/downloads/mysql/#downloads>

☐ Sélectionnez l'OS sur lequel vous travaillez (Windows, Mac OS ou Linux).

## 11.2. INSTALLATION DU LOGICIEL MYSQL (SUR WINDOWS)

- ❑ Après avoir téléchargé MySQL, exécuter le fichier téléchargé, l'installateur démarre et vous guide lors de l'installation.
- ❑ Lorsqu'il vous demande de choisir entre trois types d'installation, choisissez **"Typical"**. Cela installera tout ce dont nous pourrions avoir besoin.

## Choose Setup Type

Choose the setup type that best suits your needs



Typical

Installs the most common program features. Recommended for most users.

Custom

Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.

Complete

All program features will be installed. Requires the most disk space.

Back

Next

Cancel



## 11.2. INSTALLATION DU LOGICIEL MYSQL (SUR WINDOWS)

- ☐ L'installation se lance.
- ☐ Une fois qu'elle est terminée, cliquez sur « **Terminer** ».

# MySQL Server 5.5 Setup



Completed the MySQL Server 5.5 Setup Wizard

Click the Finish button to exit the Setup Wizard.



Launch the MySQL Instance Configuration Wizard

Back

Finish

Cancel

## 11.2. INSTALLATION DU LOGICIEL MYSQL (SUR WINDOWS)

☐ Dans l'outil de configuration suivant, choisissez la configuration standard, et à l'étape suivante, cochez l'option "Include Bin Directory in Windows PATH"

## MySQL Server Instance Configuration Wizard



### MySQL Server Instance Configuration

Configure the MySQL Server 5.5 server instance.



Please set the Windows options.

#### ☒ **Install As Windows Service**



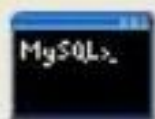
This is the recommended way to run the MySQL server on Windows.

Service Name:

MySQL

☒ Launch the MySQL Server automatically

#### ☒ **Include Bin Directory in Windows PATH**



Check this option to include the directory containing the server / client executables in the Windows PATH variable so they can be called from the command line.

< Back

Next >

Cancel

## 11.2. INSTALLATION DU LOGICIEL MYSQL (SUR WINDOWS)

- ☐ On vous propose alors de définir un nouveau mot de passe pour l'utilisateur « **root** ».
- ☐ Choisissez un mot de passe et confirmez-le.
- ☐ Ne cochez aucune autre option à cette étape.
- ☐ Cliquez ensuite sur « **Execute** » pour lancer la configuration.

## 11.3. CONNEXION À MYSQL

- ❑ Je vous ai dit que MySQL était basé sur un modèle **client - serveur**, comme la plupart des SGBD.
- ❑ Cela implique donc que votre base de données se trouve sur un **serveur** auquel vous n'avez pas accès directement, il faut passer par un **client** qui fera la **liaison entre vous et le serveur**.

## 11.3. CONNEXION À MYSQL

□ Lorsque vous installez MySQL, plusieurs choses sont donc installées sur votre ordinateur :

- un serveur de base de données MySQL ;
- plusieurs logiciels clients qui permettent d'interagir avec le serveur.

## 11.3.1. CONNEXION AU CLIENT

- ❑ Parmi ces clients, celui dont nous allons parler à présent est ***mysql*** (original comme nom ).
- ❑ C'est celui que nous utiliserons tout au long de ce cours pour ***nous connecter à notre base de données et y insérer, consulter et modifier*** des données.
- ❑ La commande pour lancer le client est tout simplement son nom :

Code : Console

```
mysql
```



## 11.3.1. CONNEXION AU CLIENT

❑ Cependant cela ne suffit pas. Il vous faut également préciser un certain nombre de paramètres.

❑ Le client *mysql* a besoin d'au minimum trois paramètres :

- *l'hôte* : c'est-à-dire l'endroit où est localisé le serveur ;
- *le nom d'utilisateur* ;
- *et le mot de passe de l'utilisateur.*

## 11.3.1. CONNEXION AU CLIENT

- ❑ L'hôte et l'utilisateur ont des valeurs par défaut, et ne sont donc pas toujours indispensables.
- ❑ La valeur par défaut de l'hôte est « **localhost** », ce qui signifie que le serveur est sur le même ordinateur que le client.
- ❑ C'est bien notre cas, donc nous n'aurons pas à préciser ce paramètre.

## 11.3.1. CONNEXION AU CLIENT

- ❑ Pour le nom d'utilisateur, la valeur par défaut dépend de votre système.
- ❑ Sous Windows, l'utilisateur courant est « **ODBC** », tandis que pour les systèmes Unix(Mac et Linux), il s'agit de votre **nom d'utilisateur** (le nom qui apparaît dans l'invite de commande).

## 11.3.1. CONNEXION AU CLIENT

❑ Pour votre première connexion à MySQL, il faudra vous connecter avec l'utilisateur « **root** », pour lequel vous avez normalement défini un mot de passe (si vous ne l'avez pas fait, inutile d'utiliser ce paramètre, mais ce n'est pas très sécurisé).

❑ Par la suite, nous créerons un nouvel utilisateur.

## 11.3.1. CONNEXION AU CLIENT

□ Pour chacun des trois paramètres, deux (2) syntaxes sont possibles :

## Code : Console

```
#####  
# Hôte #  
#####  
  
--hote=nom_hote  
  
# ou  
  
-h nom_hote  
  
#####  
# User #  
#####  
  
--user=nom_utilisateur  
  
# ou  
  
-u nom_utilisateur  
  
#####  
# Mot de passe #  
#####  
  
--password=password  
  
# ou  
  
-ppassword
```

## 11.3.1. CONNEXION AU CLIENT

- ❑ Remarquez l'absence d'espace entre -p et le mot de passe.
- ❑ C'est voulu (mais uniquement pour ce paramètre-là), et souvent source d'erreurs.
- ❑ La commande complète pour se connecter est donc :

## 11.3.1. CONNEXION AU CLIENT

### Code : Console

```
mysql -h localhost -u root -pmotdepasstopsecret
```

```
# ou
```

```
mysql --host=localhost --user=root --password=motdepasstopsecret
```

```
# ou un mélange des paramètres courts et longs si ça vous amuse
```

```
mysql -h localhost --user=root -pmotdepasstopsecret
```



## 11.3.1. CONNEXION AU CLIENT

- ❑ Nous utiliserons uniquement les paramètres courts dans la suite du cours.
- ❑ Mais choisissez ce qui vous convient le mieux.
- ❑ Notez que pour le mot de passe, il est possible (et c'est même très conseillé) de préciser uniquement que vous utilisez le paramètre, sans lui donner de valeur :

**Code : Console**

```
mysql -h localhost -u root -p
```

## 11.3.1. CONNEXION AU CLIENT

- ❑ Apparaissent alors dans la console les mots suivants :

**Code : Console**

```
Enter password:
```

## 11.3.1. CONNEXION AU CLIENT

- ❑ Tapez donc votre mot de passe, et là, vous pouvez constater que les lettres que vous tapez ne s'affichent pas.
- ❑ Cela permet simplement de cacher votre mot de passe à d'éventuels curieux qui regarderaient par-dessus votre épaule.

## 11.3.1. CONNEXION AU CLIENT

- ❑ Donc pour résumer, pour me connecter à *mysql*, je tape la commande suivante (le nom de l'hôte est omis) :

### Code : Console

```
mysql -u root -p
```

- ❑ Il ne reste plus qu'à taper mon mot de passe et je suis connecté.

## 11.3.1. CONNEXION AU CLIENT

- ❑ Maintenant que vous savez vous connecter, vous allez enfin pouvoir discuter avec le serveur MySQL (en langage SQL évidemment).
- ❑ Vous pouvez constater que vous êtes connectés grâce au joli (quoiqu'un peu formel) message de bienvenue, ainsi qu'au changement de l'invite de commande.
- ❑ On voit maintenant *mysql/*> .

## 11.3.2. DÉCONNEXION

❑ Pour se déconnecter du client, il suffit d'utiliser la commande ***quit*** ou ***exit***.

## Chap. 12 : LES TYPES DE DONNÉES

- ❑ Nous avons vu dans l'introduction qu'une base de données contenait des **tables** qui, elles-mêmes sont organisées en **colonnes**, dans lesquelles sont **stockées des données**.
- ❑ En SQL (et dans la plupart des langages informatiques), les données sont séparées en **plusieurs types** (par exemple : texte, nombre entier, date...).
- ❑ Lorsque l'on définit une colonne dans une table de la base, il faut donc lui donner un type, et toutes les données stockées dans cette colonne devront correspondre au type de la colonne.

## Chap. 12 : LES TYPES DE DONNÉES

□ Nous allons donc voir les différents types de données existant dans MySQL.



## 12.1. LES TYPES NUMÉRIQUES

□ On peut subdiviser les ***types numériques*** en deux (2) sous-catégories :

- les ***nombre entiers*** ;
- et les ***nombre décimaux***.

## 12.1.1. LES NOMBRES ENTIERS

- ❑ Les types de données qui acceptent des nombres entiers comme valeur sont désignés par le mot-clé ***INT***, et ses déclinaisons ***TINYINT***, ***SMALLINT***, ***MEDIUMINT*** et ***BIGINT***.
- ❑ La différence entre ces types est le nombre d'octets (donc la place en mémoire) réservés à la valeur du champ.
- ❑ Voici un tableau reprenant ces informations, ainsi que l'intervalle dans lequel la valeur peut être comprise pour chaque type.

## 12.1.1. LES NOMBRES ENTIERS

Type	Nombre d'octets	Minimum	Maximum
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647
BIGINT	8	-9223372036854775808	9223372036854775807

## 12.1.1. LES NOMBRES ENTIERS

- ❑ Si vous essayez de stocker une valeur en dehors de l'intervalle permis par le type de votre champ, MySQL stockera la valeur la plus proche.
- ❑ Par exemple, si vous essayez de stocker **12457** dans un **TINYINT**, la valeur stockée sera **127** ; ce qui n'est pas exactement pareil, vous en conviendrez.
- ❑ Réfléchissez donc bien aux types de vos champs.

## 12.1.1.1. L'ATTRIBUT UNSIGNED

- ❑ Vous pouvez également préciser que vos colonnes sont **UNSIGNED**, c'est-à-dire qu'on ne précise pas s'il s'agit d'une valeur positive ou négative (on aura donc toujours une valeur positive).
- ❑ Dans ce cas, la longueur de l'intervalle reste la même, mais les valeurs possibles sont décalées, le minimum valant 0.
- ❑ Pour les **TINYINT**, on pourra par exemple aller de 0 à 255.

## 12.1.1.2. LIMITER LA TAILLE D'AFFICHAGE ET L'ATTRIBUT ZEROFILL

- ❑ Il est possible de préciser le nombre de chiffres minimum à l'affichage d'une colonne de type **INT** (ou un de ses dérivés).
- ❑ Il suffit alors de préciser ce nombre entre parenthèses : **INT(x)**.
- ❑ Notez bien que cela ne change pas les capacités de stockage dans la colonne.
- ❑ Si vous déclarez un **INT(2)**, vous pourrez toujours y stocker **45282** par exemple.

## 12.1.1.2. LIMITER LA TAILLE D'AFFICHAGE ET L'ATTRIBUT ZEROFILL

- ❑ Simplement, si vous stockez un nombre avec un nombre de chiffres inférieur au nombre défini, le caractère par défaut sera ajouté à gauche du chiffre, pour qu'il prenne la bonne taille.
- ❑ Sans précision, le caractère par défaut est l'espace.
- ❑ Cette taille d'affichage est généralement utilisée en combinaison avec l'attribut **ZEROFILL**.

## 12.1.1.2. LIMITER LA TAILLE D'AFFICHAGE ET L'ATTRIBUT ZEROFILL

- ❑ Cet attribut ajoute des zéros à gauche du nombre lors de son affichage, il change donc le caractère par défaut par 0.
- ❑ Donc, si vous déclarez une colonne comme ci-dessous :

**Code : SQL**

```
INT (4) ZEROFILL
```



## 12.1.1.2. LIMITER LA TAILLE D'AFFICHAGE ET L'ATTRIBUT ZEROFILL

☐ Vous aurez l'affichage suivant :

Nombre stocké	Nombre affiché
45	0045
4156	4156
785164	785164

## 12.2. NOMBRES DÉCIMAUX

□ Cinq mots-clés permettent de stocker des nombres décimaux dans une colonne : ***DECIMAL, NUMERIC, FLOAT, REAL*** et ***DOUBLE***.

## 12.2.1. NUMERIC ET DECIMAL

□ **NUMERIC** et **DECIMAL** sont équivalents et acceptent deux paramètres : la *précision* et *l'échelle*.

- **La précision** définit le nombre de chiffres significatifs stockés, donc les 0 à gauche ne comptent pas. En effet 0024 est équivalent à 24. Il n'y a donc que deux chiffres significatifs dans 0024.
- **L'échelle** définit le nombre de chiffres après la virgule.

## 12.2.1. NUMERIC ET DECIMAL

- ❑ Dans un champ **DECIMAL(5,3)**, on peut donc stocker des nombres de **5 chiffres significatifs maximum**, dont **3 chiffres son après la virgule**.
- ❑ Par exemple : **12.354, -54.258, 89.2** ou **-56**.
- ❑ **DECIMAL(4)** équivaut à écrire **DECIMAL(4, 0)**.

## 12.2.1. NUMERIC ET DECIMAL

❑ En SQL pur, on ne peut pas stocker dans un champ DECIMAL(5,3) un nombre supérieur à 99.999, puisque le nombre ne peut avoir que deux chiffres avant la virgule (5 chiffres en tout, dont 3 après la virgule,  $5-3 = 2$  avant). Cependant, MySQL permet en réalité de stocker des nombres allant jusqu'à 999.999. En effet, dans le cas de nombres positifs, MySQL utilise l'octet qui sert à stocker le signe - pour stocker un chiffre supplémentaire.

## 12.2.1. NUMERIC ET DECIMAL

- ❑ En SQL pur, on ne peut pas stocker dans un champ **DECIMAL(5,3)** un nombre supérieur à **99.999**, puisque le nombre ne peut avoir que deux chiffres avant la virgule (5 chiffres en tout, dont 3 après la virgule,  $5-3 = 2$  avant).
- ❑ Cependant, MySQL permet en réalité de stocker des nombres allant jusqu'à **999.999**.
- ❑ En effet, dans le cas de nombres positifs, MySQL utilise l'octet qui sert à stocker le signe - pour stocker un chiffre supplémentaire.

## 12.2.1. NUMERIC ET DECIMAL

- ❑ Comme pour les nombres entiers, si l'on entre un nombre qui n'est pas dans l'intervalle supporté par la colonne, MySQL le remplacera par le plus proche supporté.
- ❑ Donc si la colonne est définie comme un **DECIMAL(5,3)** et que le nombre est trop loin dans les positifs (1012,43 par exemple), **999.999** sera stocké, et -**99.999** si le nombre est trop loin dans les négatifs.

## 12.2.1. NUMERIC ET DECIMAL

- ❑ S'il y a trop de chiffres après la virgule, MySQL arrondira à l'échelle définie.



## 12.2.2. FLOAT, DOUBLE ET REAL

- ❑ Le mot-clé **FLOAT** peut s'utiliser sans paramètre, auquel cas quatre (4) octets sont utilisés pour stocker les valeurs de la colonne.
- ❑ Il est cependant possible de spécifier une précision et une échelle, de la même manière que pour **DECIMAL** et **NUMERIC**.
- ❑ Quant à **REAL** et **DOUBLE**, ils ne supportent pas de paramètres.

## 12.2.2. FLOAT, DOUBLE ET REAL

❑ **DOUBLE** est normalement plus précis que **REAL** (stockage dans 8 octets contre stockage dans 4 octets), mais ce n'est pas le cas avec MySQL qui utilise 8 octets dans les deux cas.

❑ Je vous conseille donc d'utiliser **DOUBLE** pour éviter les surprises en cas de changement de SGBDR.

## 12.2.2. FLOAT, DOUBLE ET REAL

- ❑ **DOUBLE** est normalement plus précis que **REAL** (stockage dans 8 octets contre stockage dans 4 octets), mais ce n'est pas le cas avec MySQL qui utilise 8 octets dans les deux cas.
- ❑ Je vous conseille donc d'utiliser **DOUBLE** pour éviter les surprises en cas de changement de SGBDR.

## 12.2.2. FLOAT, DOUBLE ET REAL

### Valeurs exactes vs. valeurs approchées

- ❑ Les nombres stockés en tant que **NUMERIC** ou **DECIMAL** sont stockés sous forme de chaînes de caractères.
- ❑ Par conséquent, c'est la valeur exacte qui est stockée.
- ❑ Par contre, les types **FLOAT**, **DOUBLE** et **REAL** sont stockés sous forme de nombres, et c'est une valeur approchée qui est stockée.

## 12.2.2. FLOAT, DOUBLE ET REAL

### Valeurs exactes vs. valeurs approchées

- ❑ Cela signifie que si vous stockez par exemple **56,6789** dans une colonne de type **FLOAT**, en réalité, MySQL stockera une valeur qui se rapproche de **56,6789** (par exemple, **56,67890000000000000001**).
- ❑ Cela peut poser problème pour des comparaison notamment (**56,67890000000000000001** n'étant pas égal à **56,6789**).

## 12.2.2. FLOAT, DOUBLE ET REAL

### Valeurs exactes vs. valeurs approchées

- ❑ S'il est nécessaire de conserver la précision exacte de vos données (l'exemple type est celui des données bancaires), il est donc conseillé d'utiliser un type numérique à valeur exacte (**NUMERIC** ou **DECIMAL** donc).
- ❑ *La documentation anglaise de MySQL donne des exemples de problèmes rencontrés avec les valeurs approchées. N'hésitez pas à y faire un tour si vous pensez pouvoir être concernés par ce problème, ou si vous êtes simplement curieux.*

## 12.3. TYPES ALPHANUMÉRIQUES

## 12.3.1. CHAÎNES DE TYPE TEXTE



## 12.3.1.1. CHAR ET VARCHAR

- ❑ Pour stocker un texte relativement court (moins de 255 octets), vous pouvez utiliser les types **CHAR** et **VARCHAR**. Ces deux types s'utilisent avec un paramètre qui précise la taille que peut prendre votre texte (entre 1 et 255).
- ❑ La différence entre **CHAR** et **VARCHAR** est la manière dont ils sont stockés en mémoire.

## 12.3.1.1. CHAR ET VARCHAR

- ❑ Un **CHAR(x)** stockera toujours x octets, en remplissant si nécessaire le texte avec des espaces vides pour le compléter, tandis qu'un **VARCHAR(x)** *stockera* jusqu'à x octets (entre 0 et x), et stockera en plus en mémoire la taille du texte stocké.
- ❑ Si vous entrez un texte plus long que la taille maximale définie pour le champ, celui-ci sera tronqué.

## 12.3.1.2. TEXT

- ❑ Il suffit alors d'utiliser le type **TEXT** pour pouvoir stocker des textes de plus de 255 octets, ou un de ses dérivés **TINYTEXT**, **MEDIUMTEXT** ou **LONGTEXT**.
- ❑ La différence entre ceux-ci étant la place qu'ils permettent d'occuper en mémoire.
- ❑ Petit tableau habituel :

## 12.3.1.2. TEXT

Type	Longueur maximale	Mémoire occupée
TINYTEXT	2 <sup>8</sup> octets	Longueur de la chaîne + 1 octet
TEXT	2 <sup>16</sup> octets	Longueur de la chaîne + 2 octets
MEDIUMTEXT	2 <sup>24</sup> octets	Longueur de la chaîne + 3 octets
LONGTEXT	2 <sup>32</sup> octets	Longueur de la chaîne + 4 octets

### 12.3.1.3. CHÂÎNES DE TYPE BINAIRE

- ❑ Comme les chaînes de type texte que l'on vient de voir, une chaîne binaire n'est rien d'autre qu'une suite de caractères.
- ❑ Cependant, si les textes sont affectés par l'encodage et l'interclassement, ce n'est pas le cas des chaînes binaires.
- ❑ Une chaîne binaire n'est rien d'autre qu'une suite d'octets.
- ❑ Aucune interprétation n'est faite sur ces octets.

### 12.3.1.3. CHAÎNES DE TYPE BINAIRE

□ Ceci a deux conséquences principales :

- Une chaîne binaire traite directement l'octet, et pas le caractère que l'octet représente. Donc par exemple, une recherche sur une chaîne binaire sera toujours sensible à la casse, puisque "A" (code binaire : 01000001) sera toujours différent de "a" (code binaire : 01100001).
- Tous les caractères sont utilisables, y compris les fameux caractères de contrôle non-affichables définis dans la table ASCII.

### 12.3.1.3. CHÂÎNES DE TYPE BINAIRE

- ❑ Par conséquent, les types binaires sont parfaits pour stocker des données « brutes » comme des images par exemple, tandis que les chaînes de texte sont parfaites pour stocker...du texte !
- ❑ Les types binaires sont définis de la même façon que les types de chaînes de texte. **VARBINARY(x)** et **BINARY(x)** permettent de stocker des chaînes binaires de x caractères maximum (avec une gestion de la mémoire identique à **VARCHAR(x)** et **CHAR(x)**).

### 12.3.1.3. CHÂÎNES DE TYPE BINAIRE

□ Pour les chaînes plus longues, il existe les types ***TINYBLOB***, ***BLOB***, ***MEDIUMBLOB*** et ***LOB***, également avec les mêmes limites de stockage que les types ***TEXT***.



## 12.4. SET ET ENUM

- ❑ **SET** et ENUM sont des types propres à MySQL.
- ❑ Ils sont donc à utiliser avec une grande prudence !

## 12.4.1. ENUM

- ❑ Une colonne de type **ENUM** est une colonne pour laquelle on définit un certain nombre de valeurs autorisées, de type « **chaîne de caractère** ».
- ❑ Par exemple, si l'on définit une colonne **espece** (pour une espèce animale) de la manière suivante :

**Code : SQL**

```
espece ENUM('chat', 'chien', 'tortue')
```

## 12.4.1. ENUM

- ❑ La colonne **espece** pourra alors contenir les chaînes « **chat** », « **chien** » ou « **tortue** », mais pas les chaînes « **lapin** » ou « **cheval** ».
- ❑ En plus de « **chat** », « **chien** » et « **tortue** », la colonne **espece** pourrait prendre deux autres valeurs :
  - si vous essayez d'introduire une chaîne non-autorisée, MySQL stockera une chaîne vide dans le champ ;

## 12.4.1. ENUM

- si vous autorisez le champ à ne pas contenir de valeur (vous verrez comment faire ça dans le chapitre sur la création des tables), le champ contiendra NULL, qui correspond à "pas de valeur" en SQL (et dans beaucoup de langages informatiques).

□ Un **ENUM** peut avoir maximum 65535 valeurs possibles

## 12.4.2. SET

- ❑ SET est fort semblable à ENUM.
- ❑ Une colonne SET est en effet une colonne qui permet de stocker une chaîne de caractères dont les valeurs possibles sont prédéfinies par l'utilisateur.
- ❑ La différence avec ENUM, c'est qu'on peut stocker dans la colonne entre 0 et x valeur(s), x étant le nombre de valeurs autorisées.

## 12.4.2. SET

❑ Donc, si l'on définit une colonne de type SET de la manière suivante :

**Code : SQL**

```
espece SET('chat', 'chien', 'tortue')
```

## 12.4.2. SET

□ On pourra stocker dans cette colonne :

- "" (chaîne vide) ;
- 'chat' ;
- 'chat,tortue' ;
- 'chat,chien,tortue' ;
- 'chien,tortue' ;
- ...

## 12.4.2. SET

- ❑ Vous remarquerez que lorsqu'on stocke plusieurs valeurs, il faut les séparer par une virgule, sans espace et entourer la totalité des valeurs par des guillemets (non pas chaque valeur séparément).
- ❑ Par conséquent, les valeurs autorisées d'une colonne **SET** ne peuvent pas contenir de virgule elles-mêmes.
- ❑ On ne peut pas stocker la même valeur plusieurs fois dans un **SET**.  
**"chien,chien"** par exemple, n'est donc pas valable.



## 12.5. TYPES TEMPORELS

- ❑ Pour les données temporelles, MySQL dispose de cinq types qui permettent, lorsqu'ils sont bien utilisés, de faire énormément de choses.
- ❑ Avant d'entrer dans le vif du sujet, une petite remarque importante : lorsque vous stockez une date dans MySQL, certaines vérifications sont faites sur la validité de la date entrée.
- ❑ Cependant, ce sont des vérifications de base : le jour doit être compris entre 1 et 31 et le mois entre 1 et 12.

## 12.5. TYPES TEMPORELS

- ❑ Il vous est tout à fait possible d'entrer une date telle que le 31 février 2011.
- ❑ Soyez donc prudents avec les dates que vous entrez et récupérez.
- ❑ Les cinq types temporels de MySQL sont ***DATE, DATETIME, TIME, TIMESTAMP*** et ***YEAR***.

## 12.5.1. DATE, TIME ET DATETIME

❑ Comme son nom l'indique, **DATE** sert à stocker une date.

❑ **TIME** sert quant à lui à stocker une heure, et **DATETIME** stocke...une date ET une heure !

## 12.5.1.1. DATE

- ❑ Pour entrer une date, l'ordre des données est la seule contrainte.
- ❑ Il faut donner d'abord l'année (deux ou quatre chiffres), ensuite le mois (deux chiffres) et pour finir, le jour (deux chiffres), sous forme de nombre ou de chaîne de caractères.
- ❑ S'il s'agit d'une chaîne de caractères, n'importe quelle ponctuation peut être utilisée pour délimiter les parties (ou aucune).

## 12.5.1.1. DATE

❑ Voici quelques exemples d'expressions correctes (**A** représente les années, **M** les mois et **J** les jours) :

- **'AAAA-MM-JJ'** (c'est sous ce format-ci qu'une DATE est stockée dans MySQL)
- **'AAMMJJ'**
- **'AAAA/MM/JJ'**
- **'AA+MM+JJ'**

## 12.5.1.1. DATE

- **'AAAA%MM%JJ'**
- **AAAAMMJJ** (nombre)
- **AAMMJJ** (nombre)

## 12.5.1.1. DATE

- ❑ L'année peut donc être donnée avec deux ou quatre chiffres.
- ❑ Dans ce cas, le siècle n'est pas précisé, et c'est MySQL qui va décider de ce qu'il utilisera, selon ces critères :
  - si l'année donnée est entre 00 et 69, on utilisera le 21<sup>e</sup> siècle, on ira donc de 2000 à 2069 ;
  - par contre, si l'année est comprise entre 70 et 99, on utilisera le 20<sup>e</sup> siècle, donc entre 1970 et 1999.

## 12.5.1.1. DATE

❑ MySQL supporte des **DATE** allant de '**1001-01-01**' à '**9999-12-31**'.



## 12.5.1.2. DATETIME

- ❑ Très proche de **DATE**, ce type permet de stocker une heure, en plus d'une date.
- ❑ Pour entrer un **DATETIME**, c'est le même principe que pour **DATE** : pour la date, année-mois-jour, et pour l'heure, il faut donner d'abord l'heure, ensuite les minutes, puis les secondes.
- ❑ Si on utilise une chaîne de caractères, il faut séparer la date et l'heure par une espace.

## 12.5.1.2. DATETIME

□ Quelques exemples corrects (**H** représente les heures, **M** les minutes et **S** les secondes) :

- **'AAAA-MM-JJ HH:MM:SS'** (c'est sous ce format-ci qu'un DATETIME est stocké dans MySQL)
- **'AA\*MM\*JJ HH+MM+SS'**
- **AAAAMMJJHHMMSS** (nombre)

## 12.5.1.2. DATETIME

□ MySQL supporte des DATETIME allant de **'1001-01-01 00:00:00'** à **'9999-12-31 23:59:59'**.

### 12.5.1.3. TIME

- ❑ Le type **TIME** est un peu plus compliqué, puisqu'il permet non seulement de stocker une heure précise, mais aussi un intervalle de temps.
- ❑ On n'est donc pas limité à 24 heures, et il est même possible de stocker un nombre de jours ou un intervalle négatif.
- ❑ Comme dans **DATETIME**, il faut d'abord donner l'heure, puis les minutes, puis les secondes, chaque partie pouvant être séparée des autres par le caractère : (*deux points*).

### 12.5.1.3. TIME

❑ Dans le cas où l'on précise également un nombre de jours, alors les jours sont en premier et séparés du reste par une espace.

❑ Exemples :

- **'HH:MM:SS'**

- **'HHH:MM:SS'**

- **'MM:SS'**

## 12.5.1.3. TIME

- **'J HH:MM:SS'**
- **'HHMMSS'**
- **HHMMSS** (nombre)

□ MySQL supporte des TIME allant de **'-838:59:59'** à **'838:59:59'**.

## 12.5.2. YEAR

- ❑ Si vous n'avez besoin de ne retenir que l'année, **YEAR** est un type intéressant car il ne prend qu'un seul octet en mémoire.
- ❑ Cependant, un octet ne pouvant contenir que 256 valeurs différentes, **YEAR** est fortement limité : on ne peut y stocker que des années entre **1901** et **2155**.
- ❑ Ceci dit, ça devrait suffire à la majorité d'entre vous pour au moins les cent prochaines années.
- ❑ On peut entrer une donnée de type **YEAR** sous forme de chaîne de caractères ou d'entiers, avec 2 ou 4 chiffres.

## 12.5.2. YEAR

- ❑ Si l'on ne précise que deux chiffres, le siècle est ajouté par MySQL selon les mêmes critères que pour **DATE** et **DATETIME**, à une exception près : si l'on entre 00 (un entier donc), il sera interprété comme la valeur par défaut de YEAR 0000.
- ❑ Par contre, si l'on entre '00' (une chaîne de caractères), elle sera bien interprétée comme l'année 2000.



### 12.5.3. TIMESTAMP

- ❑ Par définition, le timestamp d'une date est le nombre de secondes écoulées depuis le 1<sup>er</sup> janvier 1970, 0h0min0s (TUC) et la date en question.
- ❑ Les timestamps étant stockés sur 4 octets, il existe une limite supérieure : le 19 janvier 2038 à 3h14min7s.
- ❑ Par conséquent, vérifiez bien que vous êtes dans l'intervalle de validité avant d'utiliser un timestamp.

## 12.5.3. TIMESTAMP

- ❑ Le type ***TIMESTAMP*** de MySQL est cependant un peu particulier.
- ❑ Prenons par exemple le ***4 octobre 2011, à 21h05min51s.***
- ❑ Entre cette date et le ***1er janvier 1970, 0h0min0s***, il s'est écoulé exactement ***1317755151 secondes.***
- ❑ Le nombre ***1317755151*** est donc, par définition, le timestamp de cette date du ***4 octobre 2011, 21h05min51s.***

### 12.5.3. TIMESTAMP

- ❑ Pourtant, pour stocker cette date dans un **TIMESTAMP SQL**, ce n'est pas **1317755151** qu'on utilisera, mais **20111004210551**. C'est-à-dire l'équivalent, au format numérique, du **DATETIME '2011-10-04 21:05:51'**.
- ❑ Le **TIMESTAMP SQL** n'a donc de timestamp que le nom.
- ❑ Il ne sert pas à stocker un nombre de secondes, mais bien une date sous format numérique **AAAAMMJJHHMMSS** (alors qu'un **DATETIME** est donc stocké sous forme de chaîne de caractères).

### 12.5.3. **TIMESTAMP**

- ❑ Il n'est donc pas possible de stocker un **"vrai" timestamp** dans une colonne de type ***TIMESTAMP***.
- ❑ C'est évidemment contre-intuitif, et source d'erreur.
- ❑ Notez que malgré cela, le ***TIMESTAMP SQL*** a les même limites qu'un vrai timestamp : il n'acceptera que des date entre le ***1e janvier 1970 à 00h00min00s*** et le ***19 janvier 2038 à 3h14min7s***.

## 12.5.4. LA DATE PAR DÉFAUT

- ❑ Lorsque MySQL rencontre une date/heure incorrecte, ou qui n'est pas dans l'intervalle de validité du champ, la valeur par défaut est stockée à la place.
- ❑ Il s'agit de la valeur **"zéro"** du type.
- ❑ On peut se référer à cette valeur par défaut en utilisant '0' (caractère), 0 (nombre) ou la représentation du "zéro" correspondant au type de la colonne (voir tableau suivant).

Type	Date par défaut ("zéro")
DATE	' 0000-00-00 '
DATETIME	' 0000-00-00 00:00:00 '
TIME	' 00:00:00 '
YEAR	0000
TIMESTAMP	00000000000000000000

## 12.5.4. LA DATE PAR DÉFAUT

❑ Une exception toutefois, si vous insérez un **TIME** qui dépasse l'intervalle de validité, MySQL ne le remplacera pas par le "**zéro**", mais par la plus proche valeur appartenant à l'intervalle de validité (-838:59:59 ou 838:59:59).

## 12.6. EN RÉSUMÉ

- ❑ MySQL définit plusieurs types de données : des **numériques entiers**, des **numériques décimaux**, des **textes alphanumériques**, des **chaînes binaires alphanumériques** et des **données temporelles**.
- ❑ Il est important de toujours utiliser le type de données adapté à la situation.
- ❑ **SET** et **ENUM** sont des types de données qui n'existent que chez MySQL. Il vaut donc mieux éviter de les utiliser.



## Chap. 13 : CRÉATION D'UNE BASE DE DONNÉES

□ Ça y est, le temps est venu d'écrire nos premières lignes de commande SQL.

## **13.1. AVANT-PROPOS : CONSEILS ET CONVENTIONS**

## 13.1.1. CONSEILS

### 1. Noms de tables et de colonnes

❑ N'utilisez jamais, au grand jamais, ***d'espaces ou d'accents*** dans vos ***noms de bases, tables ou colonnes***.

❑ Au lieu d'avoir une colonne ***"date de naissance"***, préférez ***"date\_de\_naissance"*** ou ***"date\_naissance"***. Et au lieu d'avoir une colonne ***"prénom"***, utilisez ***"prenom"***.

## 13.1.1. CONSEILS

### 1. Noms de tables et de colonnes

- ❑ Cela reste lisible, et ça vous évitera pas mal de problèmes.
- ❑ Évitez également d'utiliser des **mots réservés** comme **nom de colonnes/tables/bases**.
- ❑ Par "**mot réservé**", j'entends un mot-clé SQL, donc un mot qui sert à définir quelque chose dans le langage SQL.

## 13.1.1. CONSEILS

### 1. Noms de tables et de colonnes

- ❑ Vous trouverez une liste exhaustive des mots réservés dans la documentation officielle. Parmi les plus fréquents : ***date, text, type***.
- ❑ Ajoutez donc une précision à vos noms dans ces cas-là (***date\_naissance, text\_article*** ou ***type\_personnage*** par exemple).

## 13.1.1. CONSEILS

### 1. Noms de tables et de colonnes

- ❑ Notez que MySQL permet l'utilisation de mots-clés comme noms de tables ou de colonnes, à condition que ce nom soit entouré de ` (accent grave/backquote).
- ❑ Cependant, ceci est propre à MySQL et ne devrait pas être utilisé.

## 13.1.1. CONSEILS

### 2. Soyez cohérents

- ❑ Vous vous y retrouverez bien mieux si vous restez cohérents dans votre base.
- ❑ Par exemple, mettez tous vos noms de tables au singulier, ou au contraire au pluriel. Choisissez, mais tenez-vous-y.
- ❑ Même chose pour les noms de colonnes.

## 13.1.1. CONSEILS

### 2. Soyez cohérents

- ❑ Et lorsqu'un nom de table ou de colonne nécessite plusieurs mots, séparez les toujours avec '\_' (ex: ***date\_naissance***) ou bien toujours avec une majuscule (ex: ***dateNaissance***).
- ❑ Ce ne sont que quelques exemples de situations dans lesquelles vous devez décider d'une marche à suivre, et la garder tout au long de votre projet.



## 13.1.2. CONVENTIONS

### 1. Mots-clés

- ❑ Une convention largement répandue veut que les commandes et mots-clés SQL soient écrits complètement en majuscules.
- ❑ Il est plus facile de relire une commande de 5 lignes lorsqu'on peut différencier au premier coup d'œil les commandes SQL des noms de tables et de colonnes.

## 13.1.2. CONVENTIONS

### 2. Noms de bases, de tables et de colonnes

- ❑ On vient de voir que les mots-clés SQL seront écrits en majuscule pour les différencier du reste, donc évidemment, les noms de bases, tables et colonnes seront écrits en minuscule.

## 13.1.2. CONVENTIONS

### 2. Noms de bases, de tables et de colonnes

- ❑ Notez que MySQL n'est pas nécessairement sensible à la casse en ce qui concerne les noms de tables et de colonnes.
- ❑ En fait, il est très probable que si vous travaillez sous Windows, MySQL ne soit pas sensible à la casse pour les noms de tables et de colonnes.
- ❑ Sous Mac et Linux par contre, c'est le contraire qui est le plus probable.

## 13.2. CRÉATION ET SUPPRESSION D'UNE BASE DE DONNÉES

## 13.2.1. CRÉATION

- ❑ Nous allons donc créer notre première base de données, que nous appellerons ***elevage***.
- ❑ La commande SQL pour créer une base de données est la suivante :

**Code : SQL**

```
CREATE DATABASE nom_base;
```

## 13.2.1. CRÉATION

- ❑ Il faut également définir l'encodage à utiliser (l'UTF-8 par exemple).
- ❑ Voici donc la commande complète à taper pour créer notre base de données :

**Code : SQL**

```
CREATE DATABASE elevage CHARACTER SET 'utf8';
```

## 13.2.1. CRÉATION

□ Lorsque nous créerons nos tables dans cette base de données, automatiquement elles seront encodées également en UTF-8.

## 13.2.2. SUPPRESSION

- ❑ Si vous avez envie d'essayer cette commande (***DROP DATABASE nom\_base***), faites-le maintenant, tant qu'il n'y a rien dans votre base de données.
- ❑ Soyez très prudents, car vous effacez tous les fichiers créés par MySQL qui servent à stocker les informations de votre base.

Code : SQL

```
DROP DATABASE elevage;
```



## 13.2.2. SUPPRESSION

- ❑ Si vous essayez cette commande alors que la base de données ***elevage*** n'existe pas, MySQL vous affichera une erreur :

### Code : Console

```
mysql> DROP DATABASE elevage;  
ERROR 1008 (HY000) : Can't drop database 'elevage'; database doesn't exist  
mysql>
```

## 13.2.2. SUPPRESSION

- ❑ Pour éviter ce message d'erreur, si vous n'êtes pas sûrs que la base de données existe, vous pouvez utiliser l'option ***IF EXISTS***, de la manière suivante :

Code : SQL

```
DROP DATABASE IF EXISTS elevage;
```

## 13.2.2. SUPPRESSION

- ❑ Si la base de données existe, vous devriez alors avoir un message du type :

**Code : Console**

```
Query OK, 0 rows affected (0.00 sec)
```

## 13.2.2. SUPPRESSION

❑ Si elle n'existe pas, vous aurez :

**Code : Console**

```
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

## 13.2.2. SUPPRESSION

❑ Pour afficher les warnings de MySQL, il faut utiliser la commande ci-dessous :

**Code : SQL**

```
SHOW WARNINGS;
```

## 13.2.2. SUPPRESSION

☐ Cette commande affiche un tableau :

Level	Code	Message
Note	1008	Can't drop database 'elevage'; database doesn't exist

## 13.3. UTILISATION D'UNE BASE DE DONNÉES

- ❑ Nous avons maintenant créé une base de données.
- ❑ Mais pour pouvoir agir sur cette base, vous devez encore avertir MySQL que c'est bien sûr cette base-là que vous voulez travailler.
- ❑ Une fois de plus, la commande (***USE nom\_base***) est très simple :

**Code : SQL**

```
USE elevage
```

## 13.3. UTILISATION D'UNE BASE DE DONNÉES

- ❑ À partir de maintenant, toutes les actions effectuées le seront sur la base de données **elevage** (création et modification de tables par exemple).
- ❑ Notez que vous pouvez spécifier la base de données sur laquelle vous allez travailler lors de la connexion à MySQL.
- ❑ Il suffit d'ajouter le nom de la base à la fin de la commande de connexion :

**Code : Console**

```
mysql -u sdz -p elevage
```



## 13.4. EN RÉSUMÉ

- ❑ Pour créer une base de données, on utilise la commande « **CREATE DATABASE nom\_base** ».
- ❑ Pour supprimer une base de données : « **DROP DATABASE nom\_base** ».
- ❑ À chaque connexion à MySQL, il faut préciser avec quelle base on va travailler, avec « **USE nom\_base** ».

## Chap. 14 : CRÉATION DE TABLES

- ❑ Dans ce chapitre, nous allons créer, étape par étape, une table ***Animal***, qui servira à stocker les animaux présents dans notre élevage.
- ❑ Pour commencer, il faudra définir de quelles colonnes (et leur type) la table sera composée.
- ❑ Ne négligez pas cette étape, c'est la plus importante.
- ❑ Une base de données mal conçue est un cauchemar à utiliser.

## Chap. 14 : CRÉATION DE TABLES

- ❑ Ensuite, nous découvrirons cette fonctionnalité exclusive de MySQL que sont les moteurs de table.
- ❑ Enfin, la table ***Animal*** sera créée, et la requête de création des tables décortiquée.
- ❑ Et dans la suite, nous verrons également comment supprimer une table.

## Chap. 14 : CRÉATION DE TABLES

- ❑ Ensuite, nous découvrirons cette fonctionnalité exclusive de MySQL que sont les moteurs de table.
- ❑ Enfin, la table ***Animal*** sera créée, et la requête de création des tables décortiquée.
- ❑ Et dans la suite, nous verrons également comment supprimer une table.

## 14.1. DÉFINITION DES COLONNES

## 14.1.1. TYPE DE COLONNE DE LA TABLE

☐ Cf section précédente (Diagramme de Classes et/ou Modèle Relationnel).

## 14.1.2. NULL OR NOT NULL ?

❑ Il faut maintenant déterminer si l'on autorise les colonnes à ne pas stocker de valeur (ce qui est donc représenté par **NULL**) ou pas (**NOT NULL**).

### 14.1.3. CLÉ PRIMAIRE

- ❑ La clé primaire d'une table est une contrainte d'unicité, composée d'une ou plusieurs colonnes.
- ❑ La clé primaire d'une ligne permet d'identifier de manière unique cette ligne dans la table.
- ❑ Si l'on parle de la ligne dont la clé primaire vaut  $x$ , il ne doit y avoir aucun doute quant à la ligne dont on parle.



### 14.1.3. CLÉ PRIMAIRE

- ❑ Lorsqu'une table possède une clé primaire (et il est extrêmement conseillé de définir une clé primaire pour chaque table créée), celle-ci doit être définie.
- ❑ Nous définirons donc *id* comme la clé primaire de la table ***Animal***, en utilisant les mots-clés ***PRIMARY KEY(id)***.

### 14.1.3. CLÉ PRIMAIRE

- ❑ Lorsque vous insérerez une nouvelle ligne dans la table, MySQL vérifiera que vous insérez bien un *id*, et que cet *id* n'existe pas encore dans la table.
- ❑ Si vous ne respectez pas ces deux contraintes, MySQL n'insérera pas la ligne et vous renverra une erreur.

## 14.1.3. CLÉ PRIMAIRE

- ❑ Par exemple, dans le cas où vous essayez d'insérer un id qui existe déjà, vous obtiendrez l'erreur suivante :

**Code : Console**

```
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

- ❑ Nous reviendront en détails sur les clés primaires plus tard.

## 14.1.4. AUTO-INCRÉMENTATION

- ❑ Il faut donc, pour chaque animal, décider d'une valeur pour *id*.
- ❑ Le plus simple, et le plus logique, est de donner le numéro **1** au premier individu enregistré, puis le numéro **2** au second, etc.
- ❑ Mais si vous ne vous souvenez pas quel numéro vous avez utilisé en dernier, pour insérer un nouvel animal il faudra récupérer cette information dans la base, ensuite seulement vous pourrez ajouter une ligne en lui donnant comme *id*, **le dernier id utilisé + 1**.

## 14.1.4. AUTO-INCRÉMENTATION

- ❑ C'est bien sûr faisable, mais c'est fastidieux...
- ❑ Heureusement, il est possible de demander à MySQL de faire tout ça pour nous !
- ❑ Comment ? En utilisant l'**auto-incrémentation** des colonnes.
- ❑ Incrémenter veut dire « ajouter une valeur fixée ».

## 14.1.4. AUTO-INCRÉMENTATION

- ❑ Donc, si l'on déclare qu'une colonne doit s'auto-incrémenter (grâce au mot-clé ***AUTO\_INCREMENT***), plus besoin de chercher quelle valeur on va mettre dedans lors de la prochaine insertion.
- ❑ MySQL va chercher ça tout seul comme un grand en prenant la dernière valeur insérée et en l'incrémentant de 1.

## 14.1.5. LES MOTEURS DE TABLES

- ❑ Les moteurs de tables sont une spécificité de MySQL.
- ❑ Ce sont des moteurs de stockage.
- ❑ Cela permet de gérer différemment les tables selon l'utilité qu'on en a.
- ❑ Nous n'allons pas détailler tous les moteurs de tables existant.
- ❑ Si vous voulez plus d'informations, consultez la documentation officielle.

## 14.1.5. LES MOTEURS DE TABLES

□ Les deux moteurs les plus connus sont ***MyISAM*** et ***InnoDB***.



## 14.1.5.1. MyISAM

- ❑ C'est le moteur par défaut.
- ❑ Les commandes d'insertion et sélection de données sont particulièrement rapides sur les tables utilisant ce moteur.
- ❑ Cependant, il ne gère pas certaines fonctionnalités importantes comme les clés étrangères, qui permettent de vérifier l'intégrité d'une référence d'une table à une autre table ou les transactions, qui permettent de réaliser des séries de modifications "en bloc" ou au contraire d'annuler ces modifications.

## 14.1.5.2. InnoDB

- ❑ Plus lent et plus gourmand en ressources que **MyISAM**, ce moteur gère les clés étrangères et les transactions.
- ❑ Étant donné que nous nous servons des clés étrangères dès la deuxième partie, c'est celui-là que nous allons l'utiliser.
- ❑ De plus, en cas de crash du serveur, il possède un système de récupération automatique des données.

## 14.1.6. PRÉCISER UN MOTEUR LORS DE LA CRÉATION DE LA TABLE

- ❑ Pour qu'une table utilise le moteur de notre choix, il suffit d'ajouter ceci à la fin de la commande de création :

**Code : SQL**

```
ENGINE = moteur;
```

## 14.1.6. PRÉCISER UN MOTEUR LORS DE LA CRÉATION DE LA TABLE

❑ En remplaçant bien sûr « *moteur* » par le nom du moteur que nous voulons utiliser, ici **InnoDB** :

**Code : SQL**

```
ENGINE = INNODB;
```

## 14.2. SYNTAXE DE CREATE TABLE

- ❑ Par souci de clarté, nous allons diviser l'explication de la syntaxe de **CREATE TABLE** en deux.
- ❑ La première partie vous donne la syntaxe globale de la commande, et la deuxième partie s'attarde sur la description des colonnes créées dans la table.

## 14.2. SYNTAXE DE CREATE TABLE

**Code : SQL**

```
CREATE TABLE [IF NOT EXISTS] Nom_table (  
    colonne1 description_colonne1,  
    [colonne2 description_colonne2,  
    colonne3 description_colonne3,  
    ..., ]  
    [PRIMARY KEY (colonne_clé_primaire)]  
)  
[ENGINE=moteur];
```

## 14.2. SYNTAXE DE CREATE TABLE

- ❑ Le « ***IF NOT EXISTS*** » est facultatif (d'où l'utilisation de crochets [ ]), et a le même rôle que dans la commande ***CREATE DATABASE*** : si une table de ce nom existe déjà dans la base de données, la requête renverra un warning plutôt qu'une erreur si ***IF NOT EXISTS*** est spécifié.
- ❑ Ce n'est pas non plus une erreur de ne pas préciser la clé primaire directement à la création de la table.

## 14.2. SYNTAXE DE CREATE TABLE

□ Il est tout à fait possible de l'ajouter par la suite. Nous verrons comment un peu plus tard.