

Bases de données

Définition

- une base de données est une **collection d'informations** dont la structure reflète les **relations** qui existent entre ces données
- cette organisation et ce **système relationnel** distingue une base de données d'un simple système de fichiers ou d'une arborescence de répertoires
- une base de données peut être considérée comme la **transcription d'une partie du monde réel** (une entreprise par exemple) en informatique
- cette transcription passe par une phase de **modélisation**

Système de Gestion de Bases de Données

C'est un **logiciel** permettant la gestion des données, notamment :

- la **recherche** efficace des données
- la **suppression** des données
- le **partage** des données par de multiples utilisateurs
- la **sauvegarde** des données
- la **protection** des données

Il y a 2 types de SGBD :

Les SGBD **relationnels** : Oracle Database, MySQL, Microsoft SQLServer, PostgreSQL ...

Les SGBD **NoSQL** : Oracle NoSQL Database, MongoDB, Redis, Cassandra, Neo4J ...

Exploitation d'un système de Gestion de Bases de Données

A l'aide d'une interface graphique, ici avec phpMyadmin

The screenshot shows the phpMyAdmin interface for a MySQL database named 'music'. The 'albums' table structure is displayed in the 'Structure de table' view. The table has five columns: 'alb_id' (int(5), primary key), 'alb_art' (int(5), foreign key), 'alb_nom' (varchar(50), utf8_general_ci), 'alb_annee' (int(4)), and 'alb_prix' (decimal(5,2)). Below the table structure, there are options to print, suggest optimizations, follow the table, and move columns. An 'Index' section shows two indexes: 'PRIMARY' (BTREE, Unique, Compressed) on 'alb_id' and 'fk_albums_1' (BTREE, Not Unique, Not Compressed) on 'alb_art'. At the bottom, there is a form to create an index on 1 column.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires
<input type="checkbox"/>	1	alb_id	int(5)		Non	Aucun(e)	Identifiant de l'album
<input type="checkbox"/>	2	alb_art	int(5)		Non	Aucun(e)	Code artiste
<input type="checkbox"/>	3	alb_nom	varchar(50)	utf8_general_ci	Non	Aucun(e)	Nom de l'album
<input type="checkbox"/>	4	alb_annee	int(4)		Non	Aucun(e)	Année de sortie
<input type="checkbox"/>	5	alb_prix	decimal(5,2)		Non	Aucun(e)	Prix de l'album

Action	Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement
Éditer Supprimer	PRIMARY	BTREE	Oui	Non	alb_id	51	A
Éditer Supprimer	fk_albums_1	BTREE	Non	Non	alb_art	25	A

Exploitation d'un système de Gestion de Bases de Données

En ligne de
commandes

Ici avec
cmd sous
Windows

```
cmd Invite de commandes - mysql -u root -p
C:\xampp\mysql\bin>
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.28-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
+-----+
4 rows in set (0.015 sec)

MariaDB [(none)]> use mysql;
Database changed
MariaDB [mysql]> show tables;
+-----+
| Tables_in_mysql |
+-----+
| column_stats |
| columns_priv |
| db |
| event |
| func |
| general_log |
| global_priv |
| gtid_slave_pos |
| help_category |
| help_keyword |
| help_relation |
| help_topic |
| index_stats |
| innodb_index_stats |
| innodb_table_stats |
| plugin |
| proc |
| procs_priv |
| proxies_priv |
| roles_mapping |
| servers |
| slow_log |
| table_stats |
| tables_priv |
| time_zone |
| time_zone_leap_second |
| time_zone_name |
| time_zone_transition |
| time_zone_transition_type |
| transaction_registry |
| user |
+-----+
31 rows in set (0.001 sec)
```

Fonctionnalités détaillées

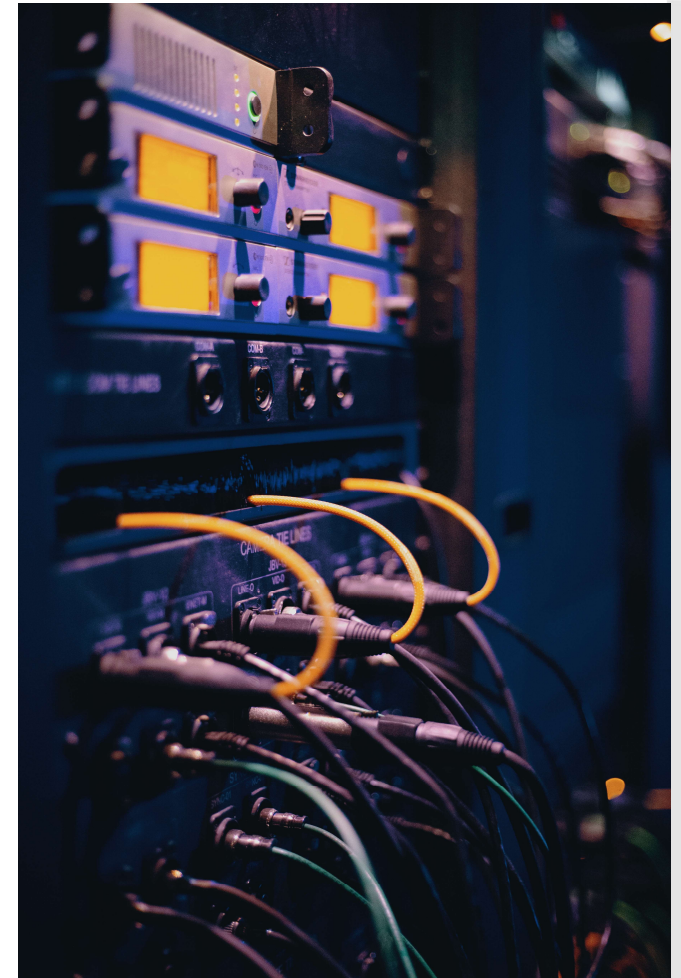
- **indépendance physique (données / programmes) :**
pouvoir modifier l'organisation physique sans modifier les programmes (accès local ou distant à la base de données, changement de stockage ...)
- **indépendance logique :**
pouvoir modifier le schéma conceptuel sans modification des programmes (beaucoup plus difficile)
- **manipulation des données :**
permettre à des utilisateurs qui n'ont pas la connaissance de l'organisation de la base ni les acquis techniques de manipuler des données

Fonctionnalités détaillées

- **efficacité de traitement des données :**
permettre de traiter efficacement les données à l'aide de langages différents utilisant les **fonctionnalités multicritères**
- **administration centralisée de la base de données :**
l'administrateur de la base définit la structure des données, leur **stockage** et leur **contrôle** : utilisateurs, droits d'accès
- **intégrité des données :**
l'administrateur définit des **règles de cohérence** : contraintes d'intégrité, contraintes de mises à jour et de suppression

Fonctionnalités détaillées

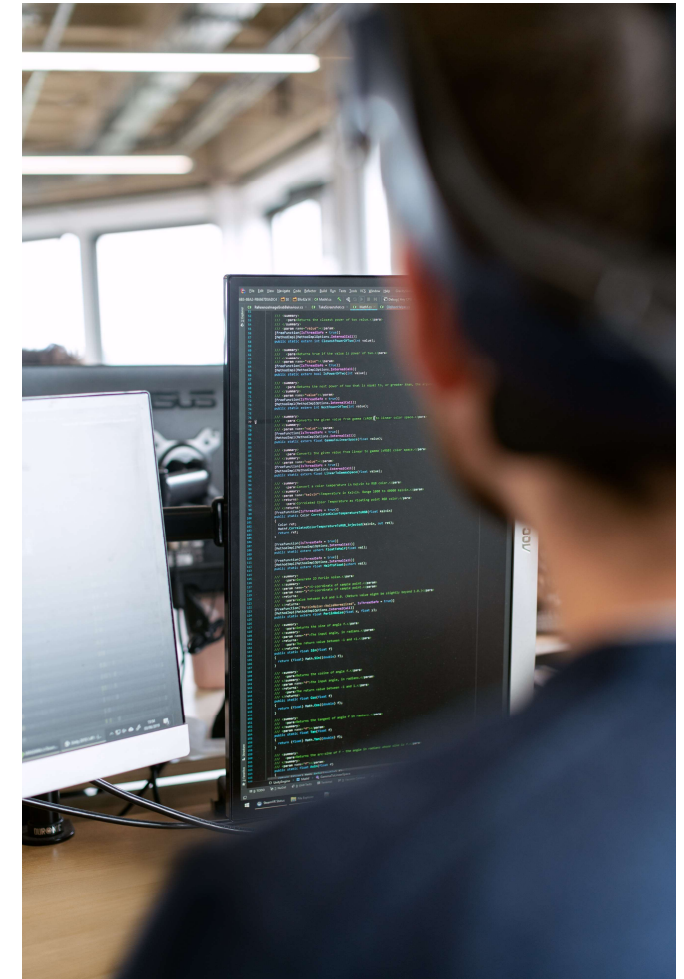
- **partage des données** : plusieurs utilisateurs ont accès aux données via plusieurs applications différentes
- **sécurisation et sauvegarde des données** : l'administrateur définit les **droits d'accès** des utilisateurs aux données et les **modes de sauvegardes** : export, snapshot, réplication etc.



Niveaux d'utilisation

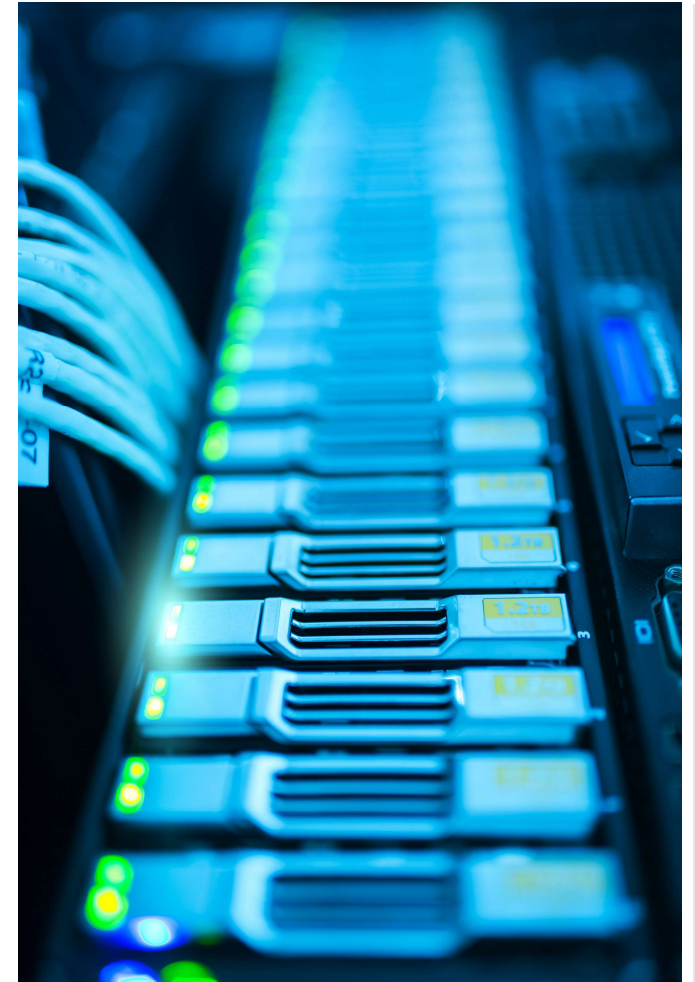
Méthodologie du concepteur

- utilisation de méthodes et langages (Merise, UML) pour créer un **modèle conceptuel** qui permet de définir :
 - les différents **objets** de la base de données
 - les **relations** entre les objets
 - les **contraintes** sur les données



Niveaux d'utilisation

- le niveau interne ou physique
spécification du **stockage**
physique des données (fichiers,
disques, etc.) et des **méthodes**
d'accès
- le niveau externe
possibilité de création de **vues**
externes pour différents
groupes d'utilisateurs sur tout
ou partie de la base de données

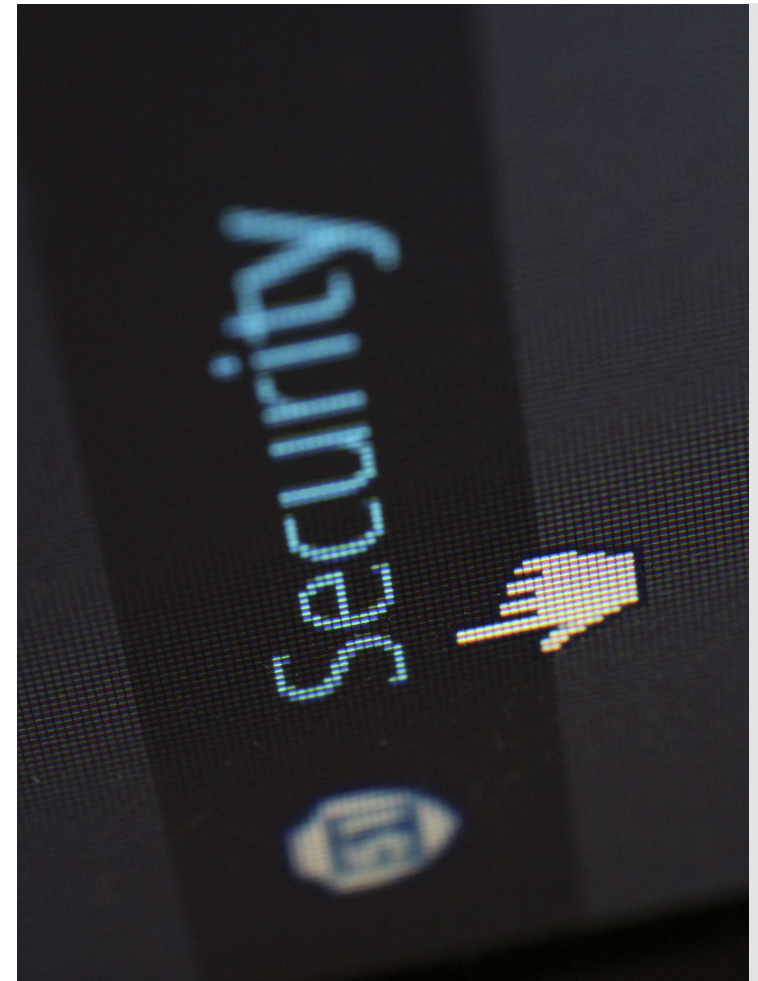


Les acteurs

L'administrateur

L'administrateur a en charge :

- la définition du **schéma conceptuel**
- l'évolution du **schéma conceptuel**
- la création / l'évolution des tables
- les modalités de **protection** des données
- diverses tâches de maintenance et de **sécurisation** de la base de données : sauvegardes, restauration, optimisation, gestion des utilisateurs et droits d'accès

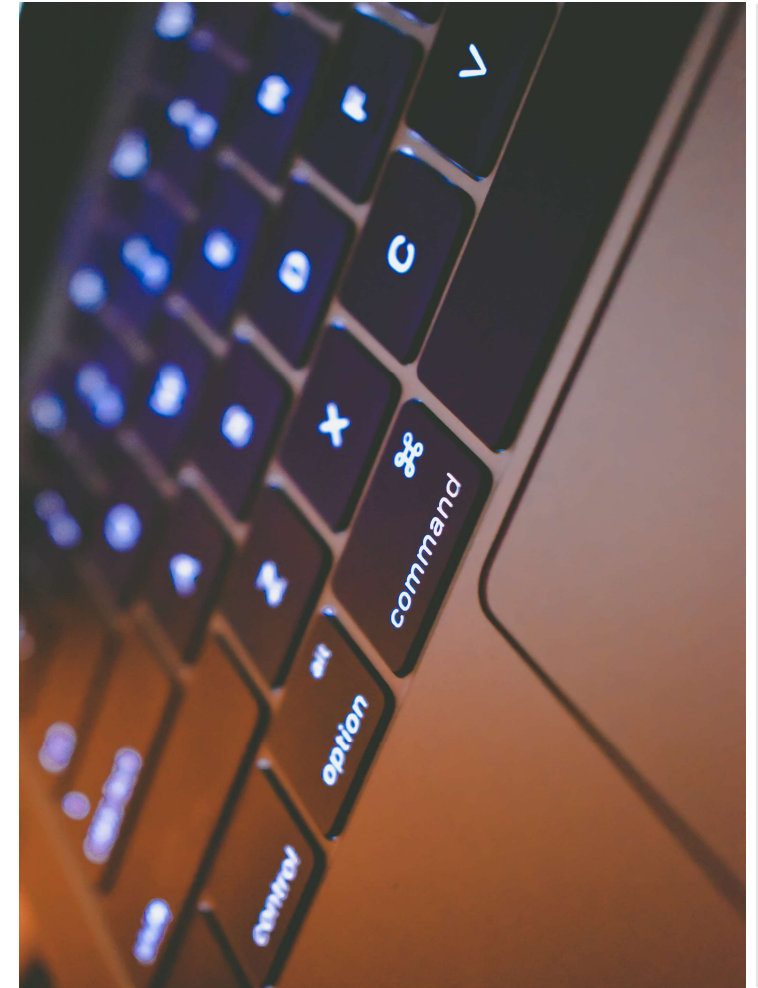


Les
acteurs

Le
développeur

Le développeur

- il crée et maintient des bibliothèques de programmes de manipulation de la base de données : interrogation, mise à jour, etc.
- il utilise des langages de programmation (de manipulation de données) liés ou non au moteur même de la base de données



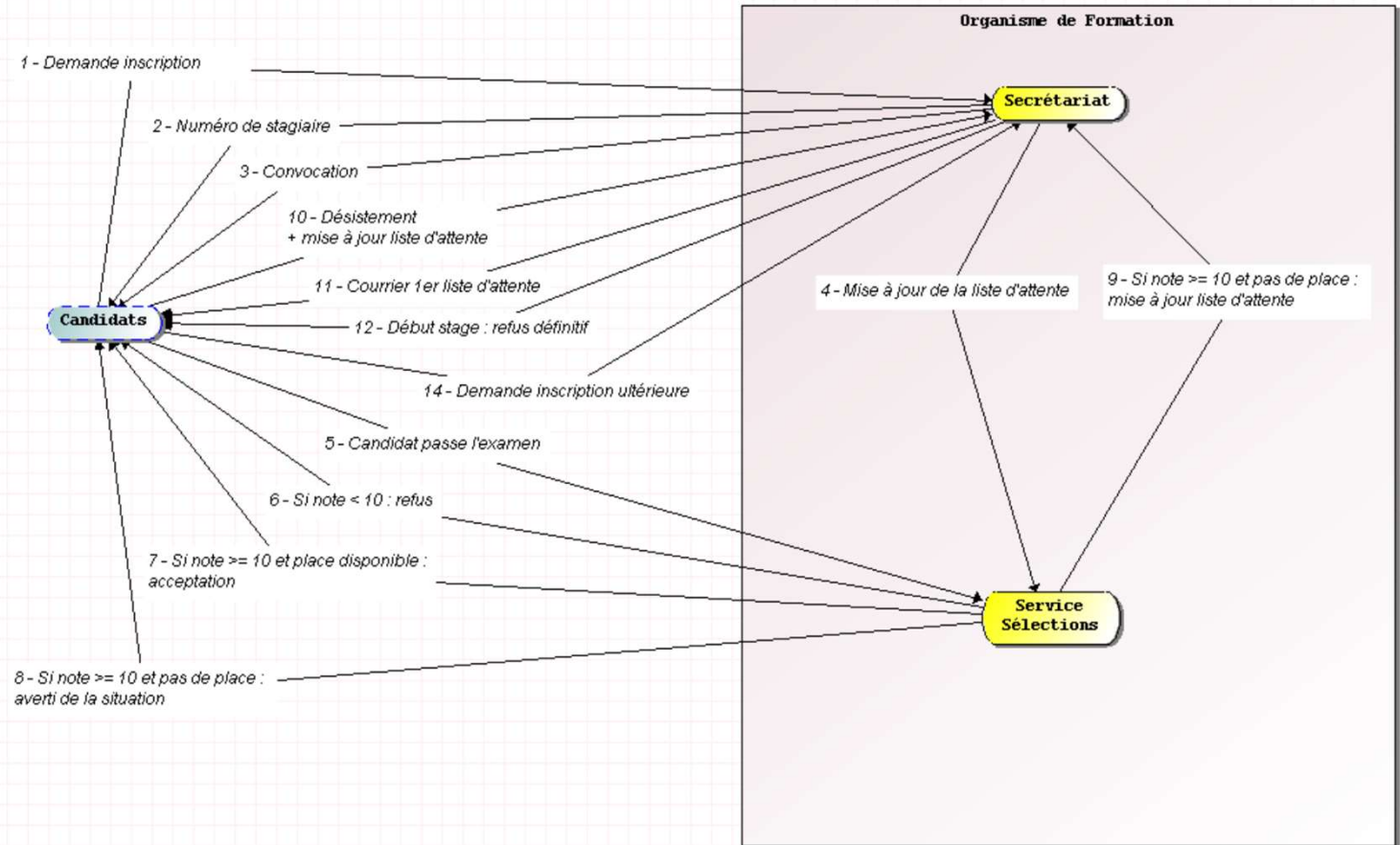
Modélisation

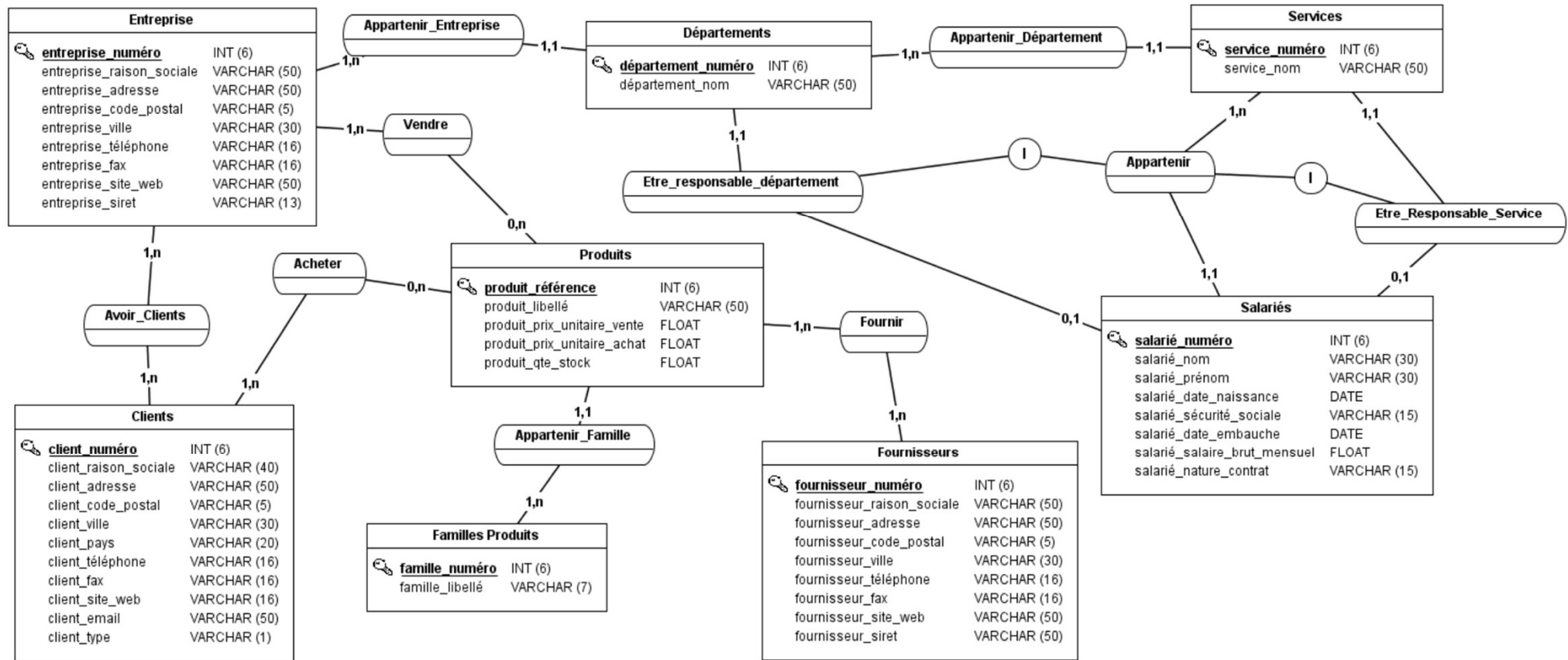
Le diagramme des flux de données

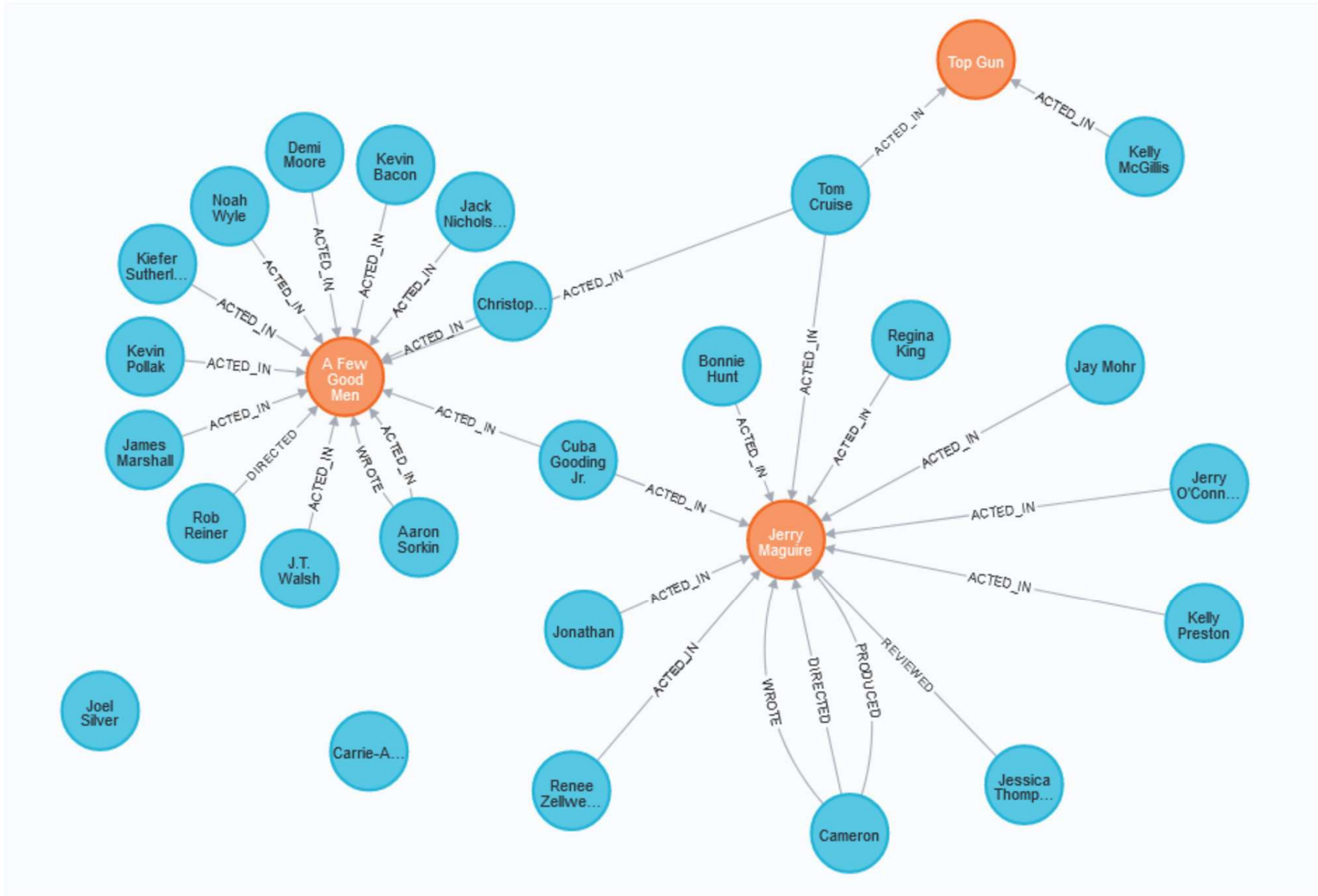
- un diagramme de flux met en évidence la **circulation des données** au sein d'un système d'information
- il précise
 - les **acteurs** internes et externes au domaine d'étude
 - les **flux d'informations orientés**
 - la **chronologie des flux**

Modélisation

Le diagramme des flux de données







Les
nouveaux
besoins

Du
centralisé
aux
DataCenters

- suite à l'apparition d'application Web et service mobiles (Amazon, Facebook, Google, Twitter ...) avec un énorme volume de données à gérer, les **plateformes traditionnelles** ont montré des **limites**
- on a donc commencé à **distribuer les données et leur traitement** sur beaucoup de serveurs
- c'est l'avènement des **datacenters** dont voici quelques caractéristiques :
 - les serveurs sont regroupés en **racks** de machines reliées
 - un datacenter contient un **grand nombre de racks interconnectés**
 - entre différents datacenters, on a des **connexions internet ultra rapides**

Les
nouveaux
besoins

Du
centralisé
aux
DataCenters



Les
nouveaux
besoins

Du
centralisé
aux
DataCenters

- un datacenter Google contient plusieurs centaines de racks
- chaque rack contient au minimum 50 serveurs
- il peut y avoir plus de 5 000 serveurs par datacenter
- Google à lui seul représente plus d'un million de serveurs



Les insuffisances des SGBDR classiques

Les systèmes de gestion de bases de données **relationnels classiques** sont incapables de traiter de **gros volumes** avec des **temps de réponses** satisfaisants

Ils n'ont **pas été conçus pour cela**.

On constate :

- un **manque d'efficacité** pour des données peu ou non structurées
- un **manque de puissance** dans le cas de **gros volumes**
- **des difficultés d'évolution** de la structure lors de la **montée en charge** (augmentation de volumes de données, augmentation d'un nombre de connexions simultanées à une application ...)

Les insuffisances des SGBDR classiques

- les **meilleurs spécialistes** du domaine travaillent pour les **éditeurs** des bases de données (Oracle, Microsoft ...)
- donc il y a un manque de **compétences** pour l'optimisation :
 - de la structure de la base de données
 - des stockages (types) et des volumes (supports)
 - des requêtes sur les données
- les **prix des licences** de certaines bases de données sont trop élevés pour beaucoup d'entreprises : exemple Oracle

Qu'est-ce que la scalabilité ?

- la **scalabilité** (ou mise à l'échelle) est la capacité d'un produit, matériel ou logiciel, à :
 - s'adapter à un **changement de la demande**
 - supporter la **montée en charge**
 - tout en conservant **fonctionnalités et performances**



Qu'est-ce que la scalabilité ?

La mise à l'échelle **verticale** consiste à **augmenter les capacités matérielles** de la machine

Elle présente **2 inconvénients** :

- les applications sont indisponibles pendant qu'on augmente les capacités des serveurs
- les coûts d'évolution sont importants

La mise à l'échelle **horizontale** consiste à **lancer la même application sur plusieurs serveurs** pour **répartir la charge de travail**

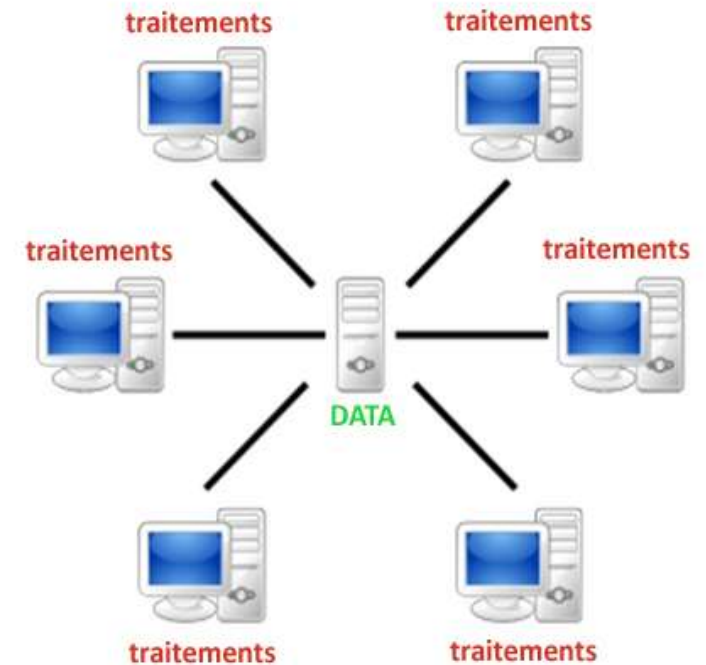
- avec cette solution, il n'y a plus de risques d'indisponibilité de l'application comme dans la mise à l'échelle verticale

2 solutions

Distribution des Traitements

Distribution des traitements

- on distribue les traitements sur un nombre important de serveurs afin d'absorber des charges très importantes
- on envoie les données vers les serveurs où s'exécutent les traitements des données



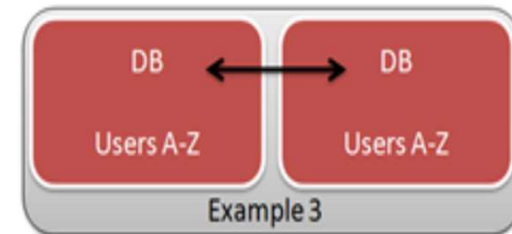
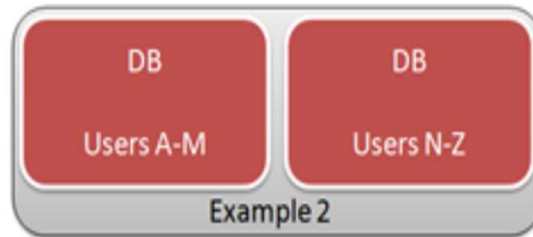
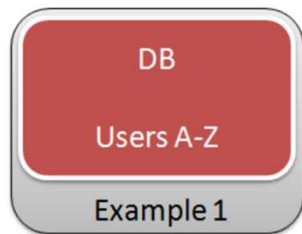
2 solutions

Distribution des Données

Distribution des données

- on distribue les données sur un nombre important de serveurs afin de stocker de très grands volumes de données
- on envoie les programmes vers les serveurs où se trouvent les données
- il est plus efficace de transférer un petit programme sur le réseau plutôt qu'un grand volume de données





- dans l'exemple 1 : si le seul serveur tombe en panne, on perd 100% des données
- dans l'exemple 2 : si un des 2 serveurs tombe en panne, on perd 50% des données
- dans l'exemple 3 : une simple **réplication** de la base de données sur un autre serveur assure une **disponibilité de 100% des données**

Map Reduce

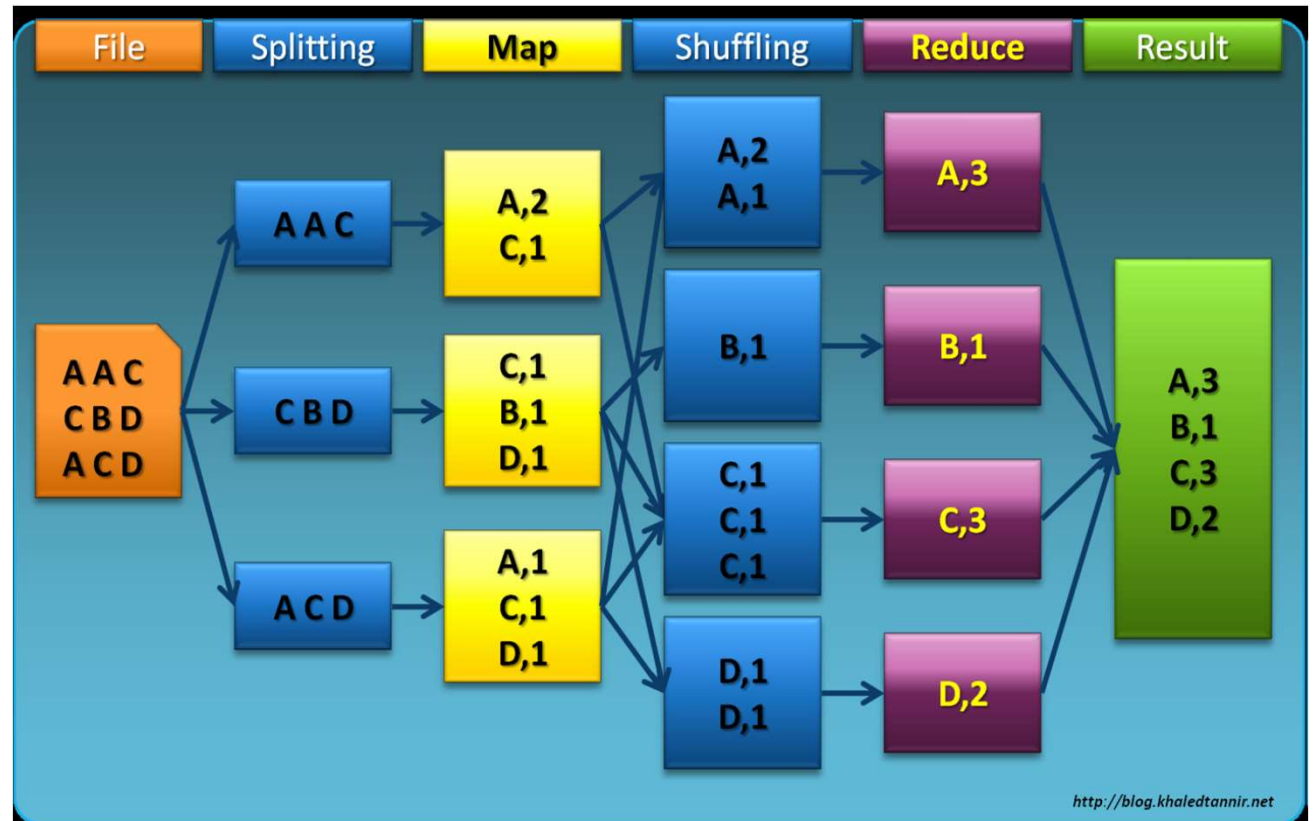
- c'est un modèle de programmation parallèle (calcul distribué) développé par Google pour le **traitement de gros volumes** de données en environnement distribué
- il permet de **répartir la charge** sur un grand nombre de serveurs
- il gère entièrement et de façon transparente les serveurs, la distribution de données, la répartition de la charge, et la tolérance aux pannes
- ajouter des machines augmente la performance
- la librairie MapReduce existe dans **plusieurs langages** (C++, C#, Erlang, Java, Python, Ruby...)
- elle est donc rapidement devenue **populaire**

Map Reduce

Map Reduce est utilisé par les grands acteurs du Web pour :

- construire les index pour les moteurs de recherches (Google Search)
- la détection des spams
- l'analyse d'images astronomiques, la simulation de la météo, l'apprentissage automatique (machine learning) etc...
- trouver le nombre d'occurrences d'une information recherchée dans un très grand volume de données
- classifier de très grands volumes de données provenant par exemple de paniers d'achats de clients (Data Mining)

- soit un fichier document contenant 3 lignes composées chacune de 3 mots parmi les mots {A, B, C, D}.
- il s'agit de compter tous les mots contenus dans ce fichier



UML

3 modèles =
3 approches)

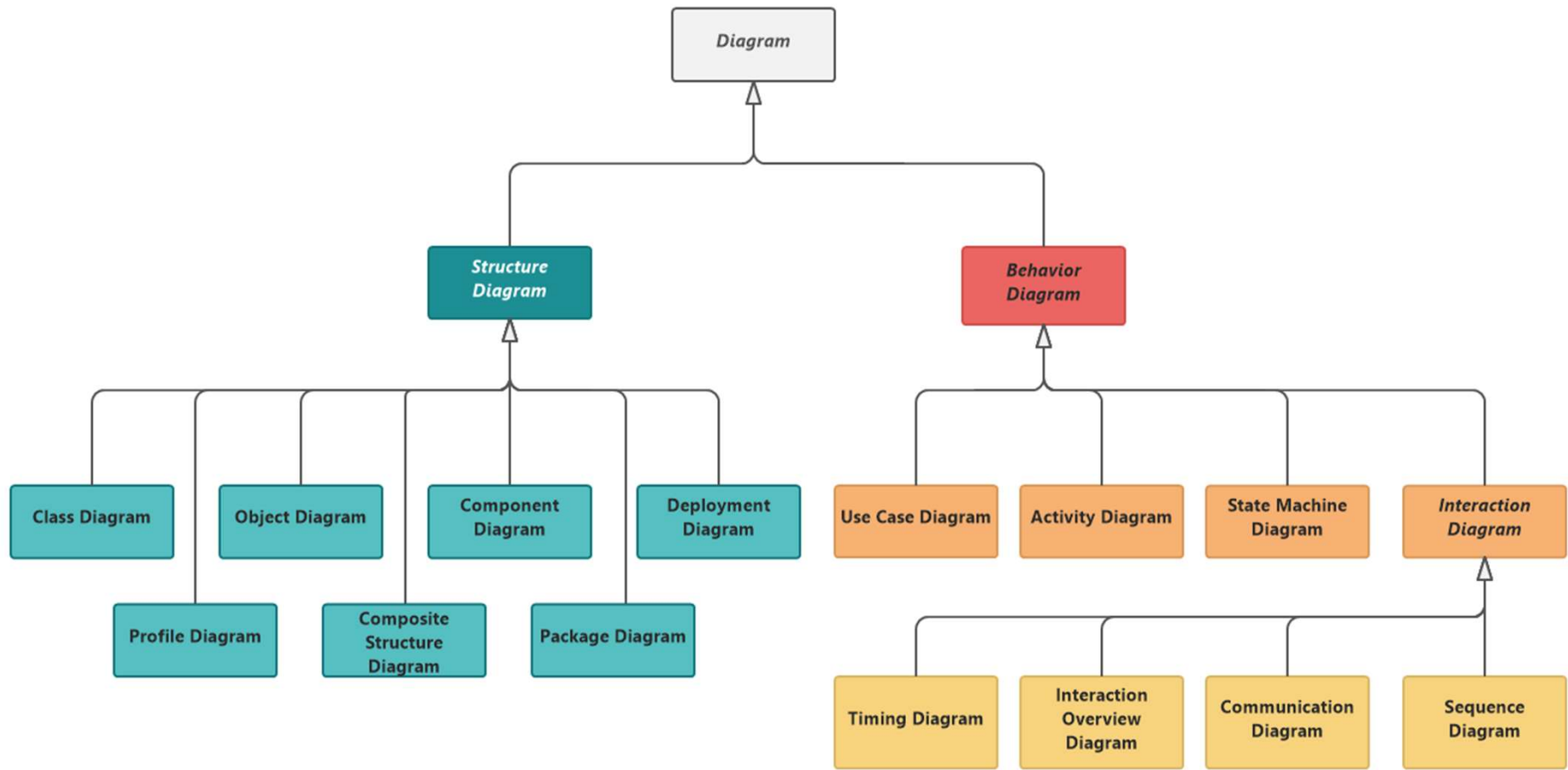
- modèle fonctionnel : description des fonctionnalités du **point de vue de l'utilisateur**
- modèle objet : description des **structures** en termes **d'objets, attributs, associations et opérations**
- modèle dynamique : diagrammes qui décrivent le **comportement interne du système** (interaction, états-transitions et activités)

UML : 2 groupes de diagrammes

- 7 diagrammes **structurels** : classes, objets, composant, structure composite, déploiement, package, profil
- 7 diagrammes **comportementaux** : cas d'utilisation, activité, états-transitions, **séquence, communication, global d'interaction, temps**

N. B. : les 4 derniers diagrammes comportementaux font partie d'un sous-groupe appelé **diagrammes d'interaction**

- il existe donc **14 types de diagrammes UML**
- chaque type de diagramme UML se concentre sur un **aspect spécifique** du modèle UML



Le cas d'utilisation (use case)

- il représente un ensemble de **séquences d'actions** réalisées par le système et observables par un **acteur** particulier
- chaque cas d'utilisation décrit un comportement du système et correspond à une **fonction métier** du système

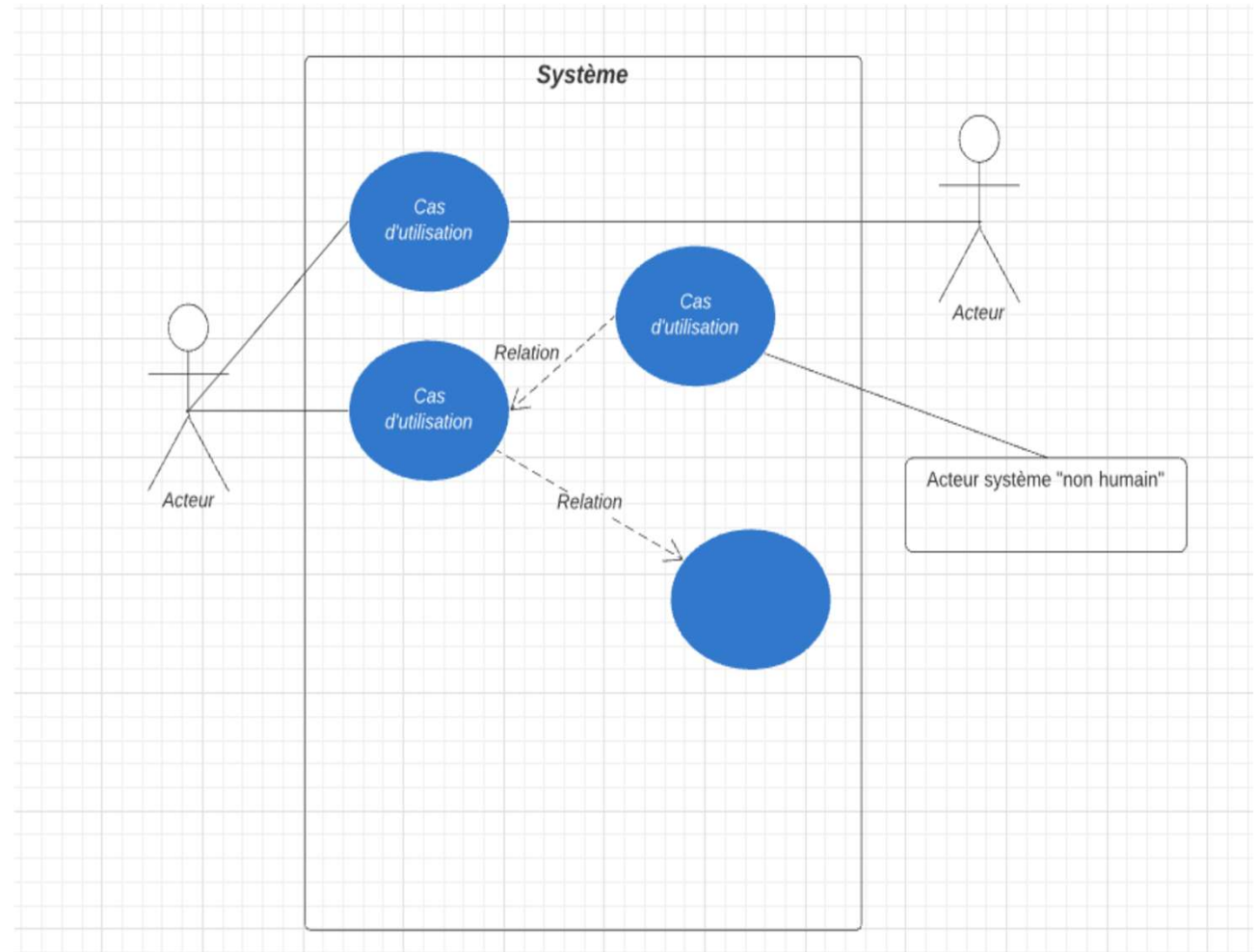


Diagramme de cas d'utilisation



L'acteur

- il représente un rôle joué par une entité externe (utilisateur humain, matériel ou système ...) qui interagit directement avec le système étudié
- sa représentation standard est ce que l'on appelle un **stick man** sous lequel on retrouve le nom de l'acteur
- le **stick man** est par convention réservé aux **acteurs humains**
- on peut également représenter un acteur avec un **rectangle** et le mot-clé **Acteur**, réservé aux matériels ou aux systèmes

- Voici un exemple très simple de **cas d'utilisation** illustrant l'usage d'un distributeur de billets par deux acteurs :
 - un porteur de carte bancaire
 - un client de la banque propriétaire du distributeur de billets

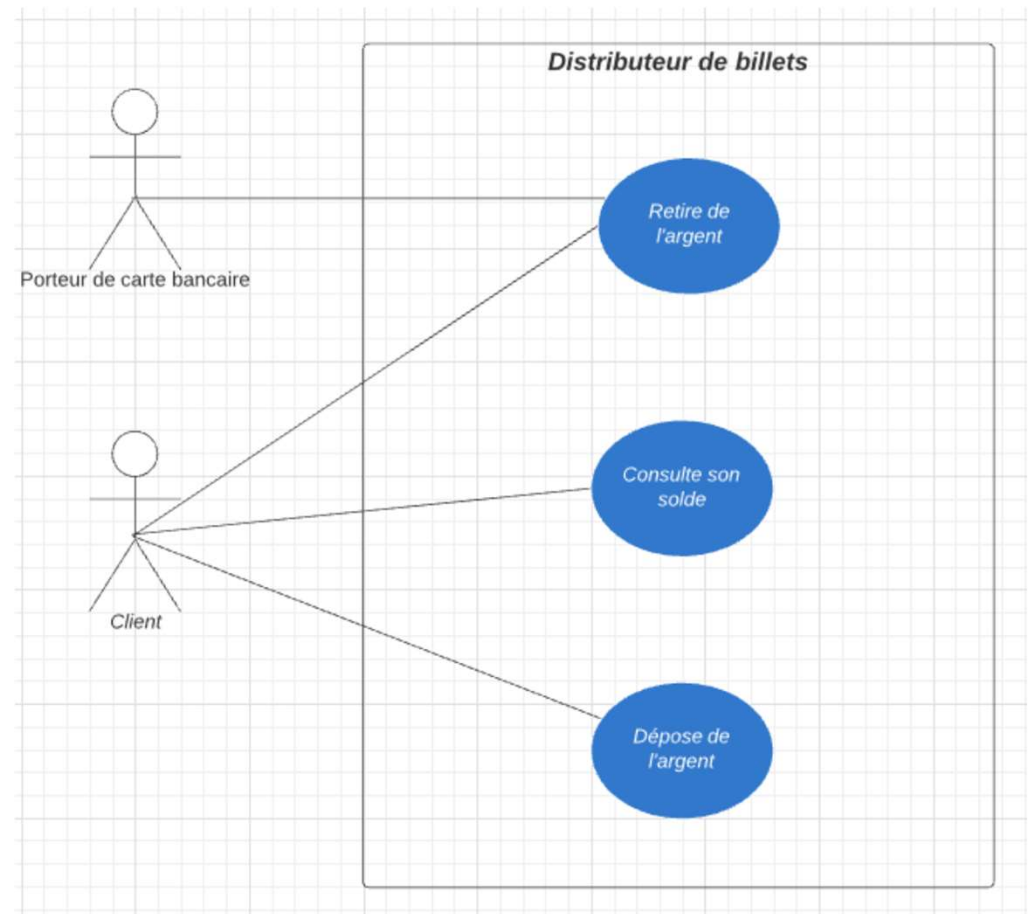


Diagramme de cas d'utilisation

Autre exemple

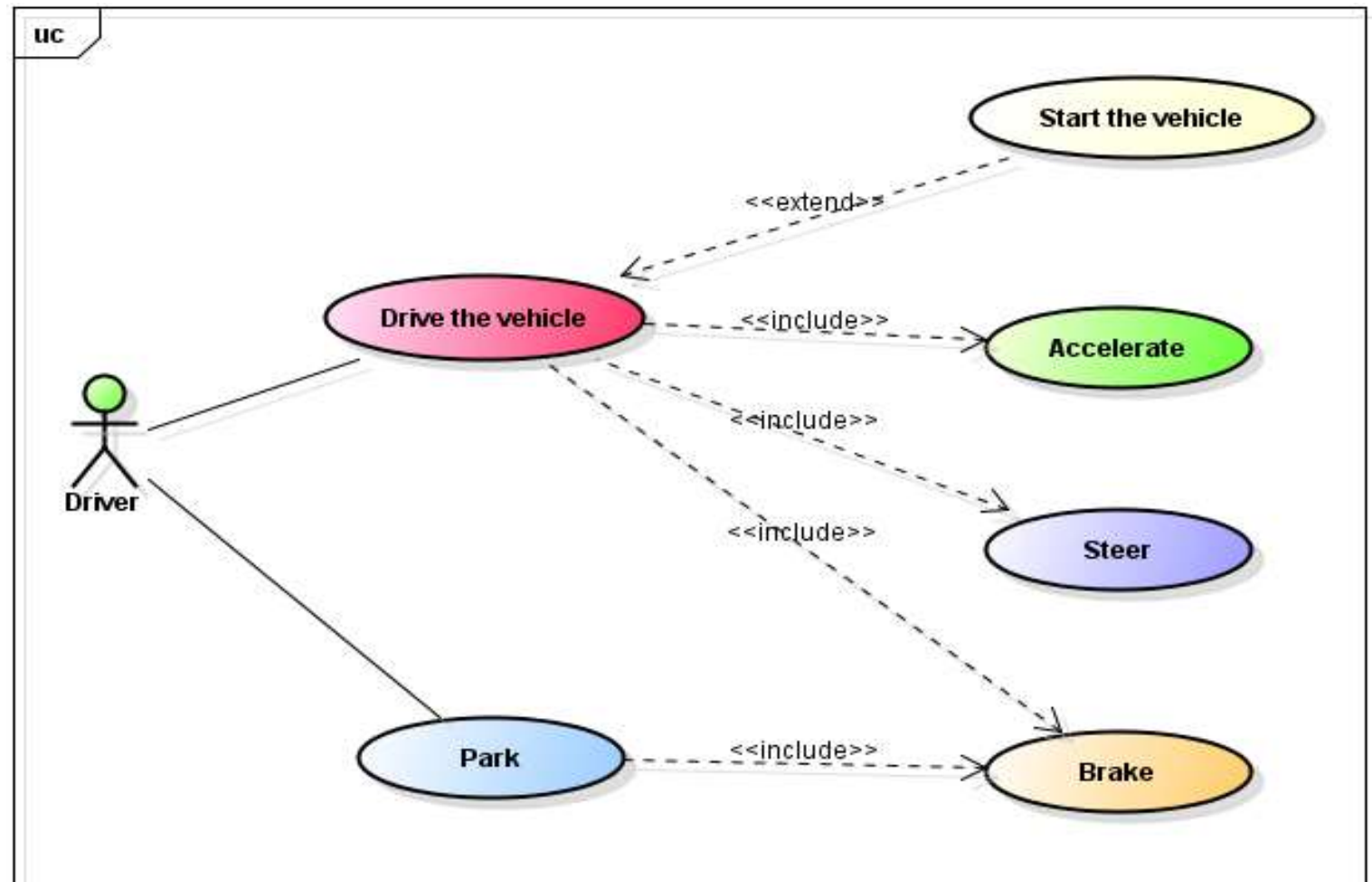


Diagramme
de cas
d'utilisation
→
Diagramme
de
Séquence

Le diagramme de séquence découle du diagramme d'utilisation et ce, grâce à la notion de scénario.

Le scénario

- chaque cas d'utilisation doit être décrit à l'aide d'un **scénario détaillé**, qui précise la **succession des actions** du **début** du cas d'utilisation à la **fin** du cas d'utilisation
- ce scénario n'est pas présent dans ce diagramme mais sera décrit en complément dans un **diagramme de séquences**
- exemple à suivre : le distributeur de billets

Scénario du cas d'utilisation retire de l'argent :

- le porteur de carte introduit sa carte
- le distributeur vérifie la validité de la carte
- le distributeur demande la saisie du code
- le porteur saisit le code de sa carte
- le distributeur vérifie le code de la carte
- le distributeur demande le montant à retirer
- le porteur saisit le montant
- ...

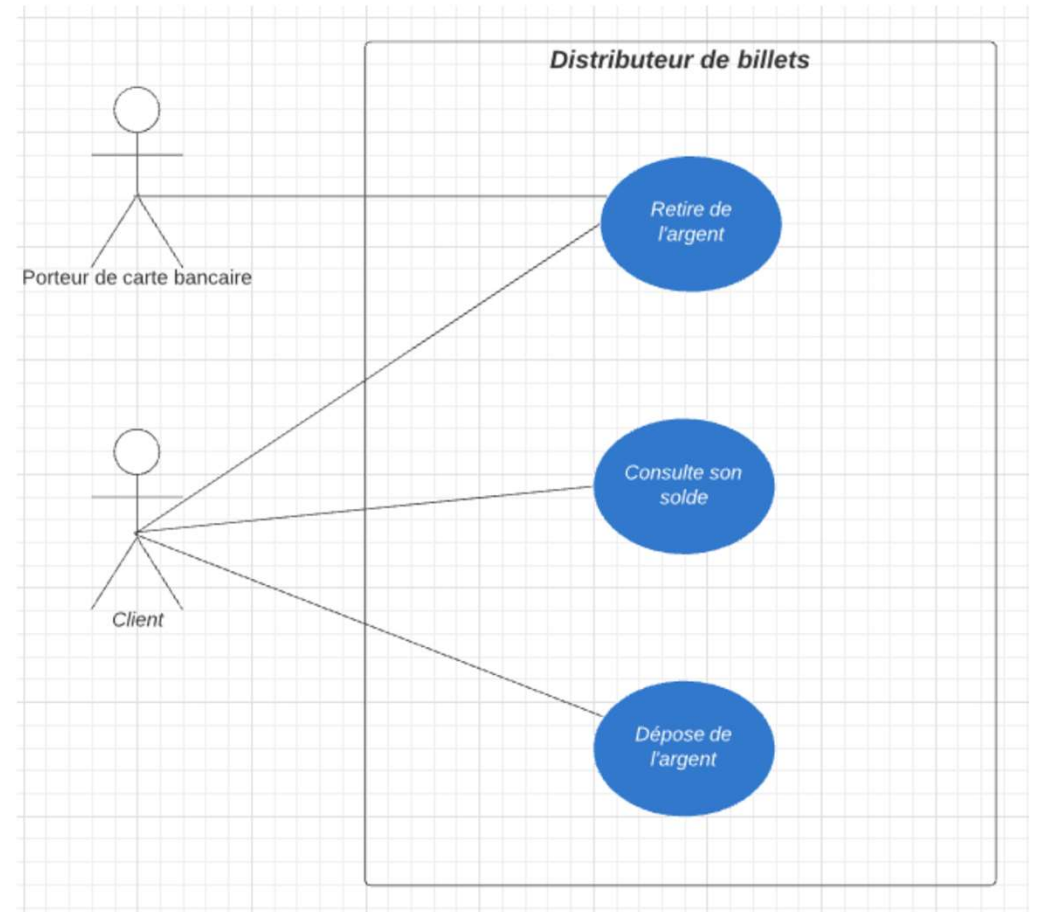


Diagramme de Séquence

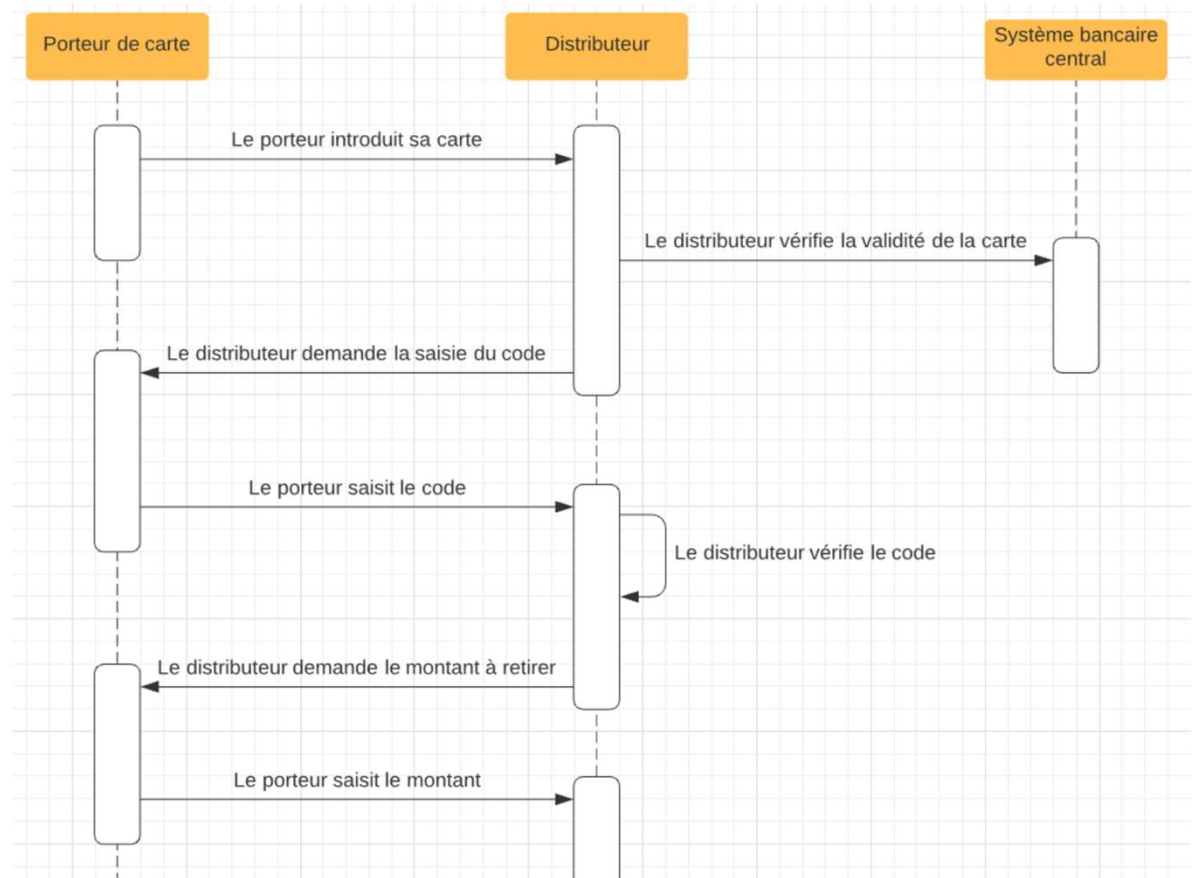
- un diagramme de séquence reprend tous les détails des **cas d'utilisation** décrits dans les **scénarios**
- il permet d'**illustrer** les cas d'utilisation, passant du **scénario textuel** à un **schéma visuel**
- ce **visuel** est beaucoup plus « **communicant** » pour un concepteur / développeur, pour son équipe et pour échanger avec un client
- il est également plus **facilement modifiable** qu'un texte

Diagramme de Séquence

Un diagramme de séquence peut être complété par :

- les **scénarios alternatifs** qui correspondent à d'autres hypothèses que le scénario principal, par exemple si la carte bancaire n'est pas valide ou pas lisible
- les **messages d'erreurs** : on pourrait modéliser les cas d'erreurs de fonctionnement du distributeur (panne ou défaillance de la connexion entre le distributeur et le système bancaire central), ou encore les erreurs de saisie du porteur de cartes et le traitement informatique qui en serait fait (blocage de carte)

- dans l'exemple du distributeur de billets, ce diagramme de séquence reprend le scénario présenté plus haut
- cela donne ce début de schéma →



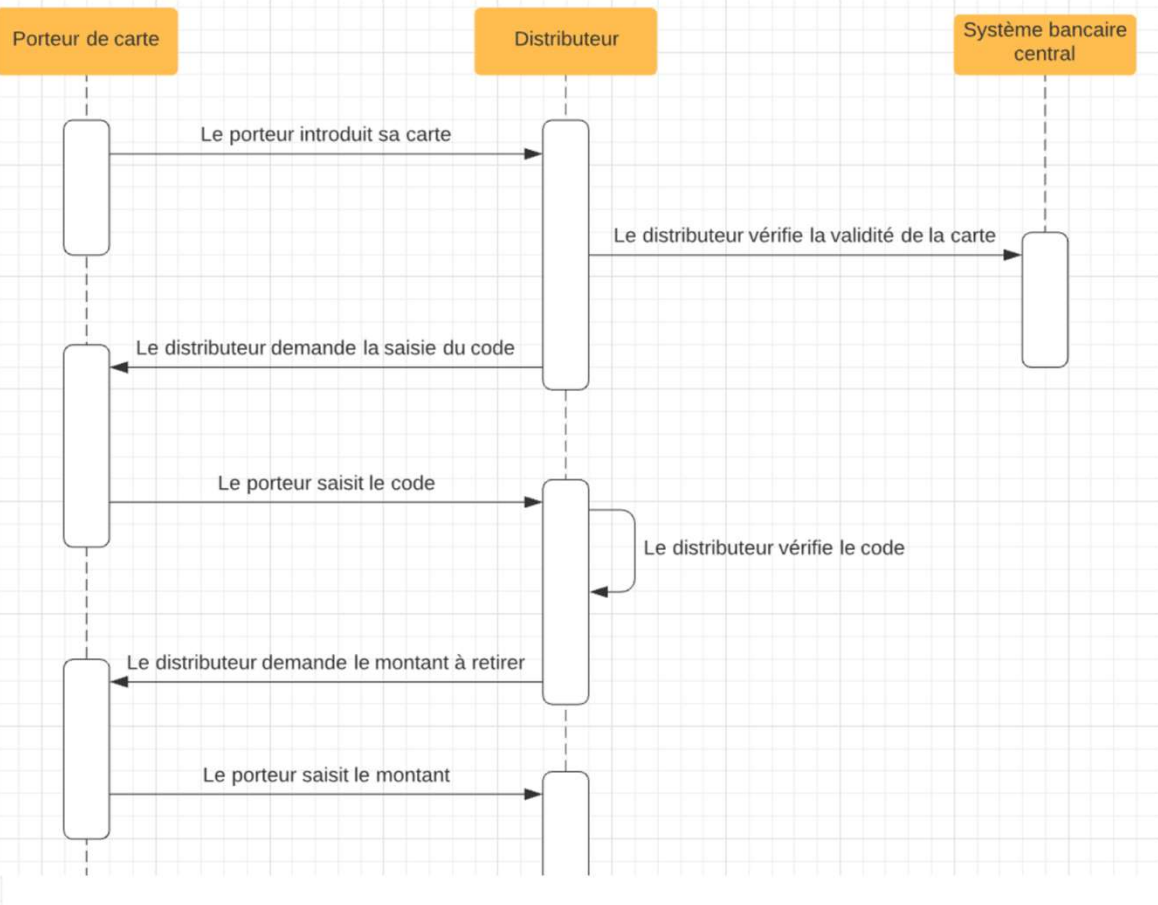
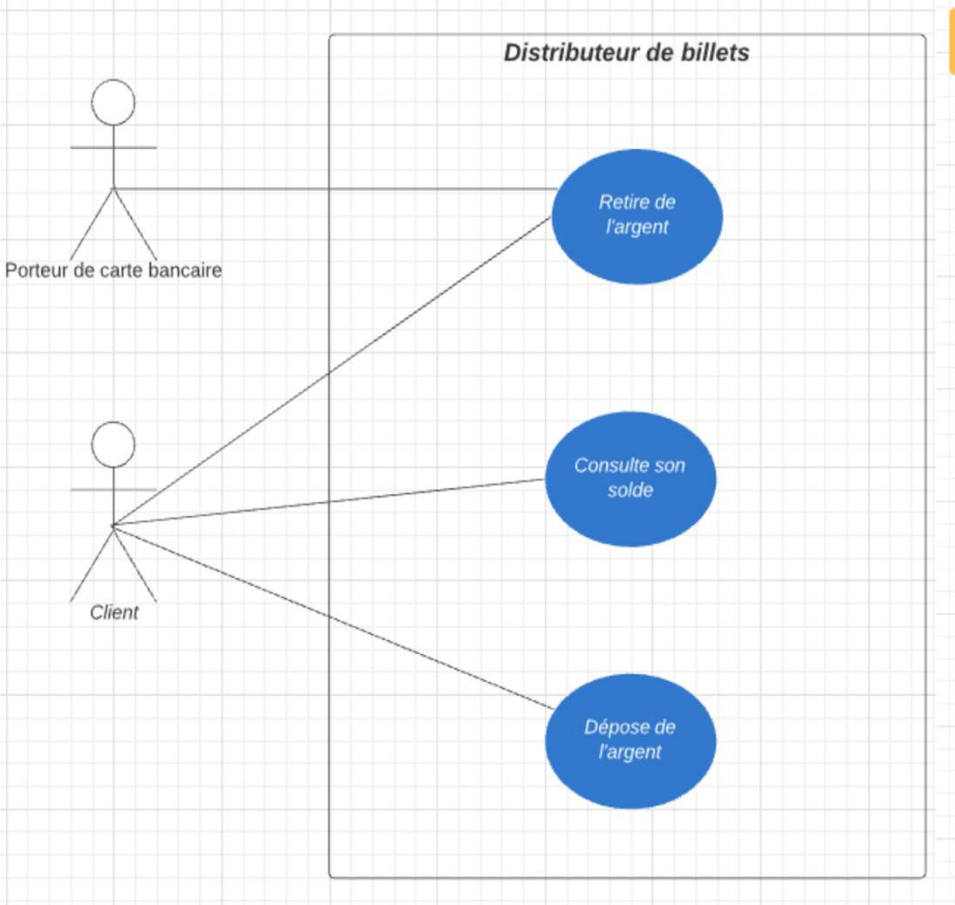
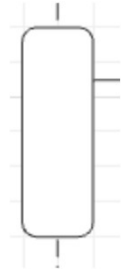


Diagramme de Séquence

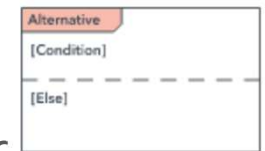
Symboles

- **boîte d'activation** : c'est le temps nécessaire pour qu'un objet accomplisse une tâche.

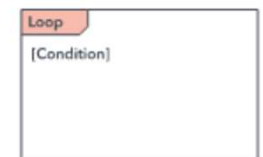
Plus la tâche dure, plus la boîte d'activation est haute.



- **condition** : cette boîte symbolise les alternatives exclusives ou non, entre deux séquences de messages

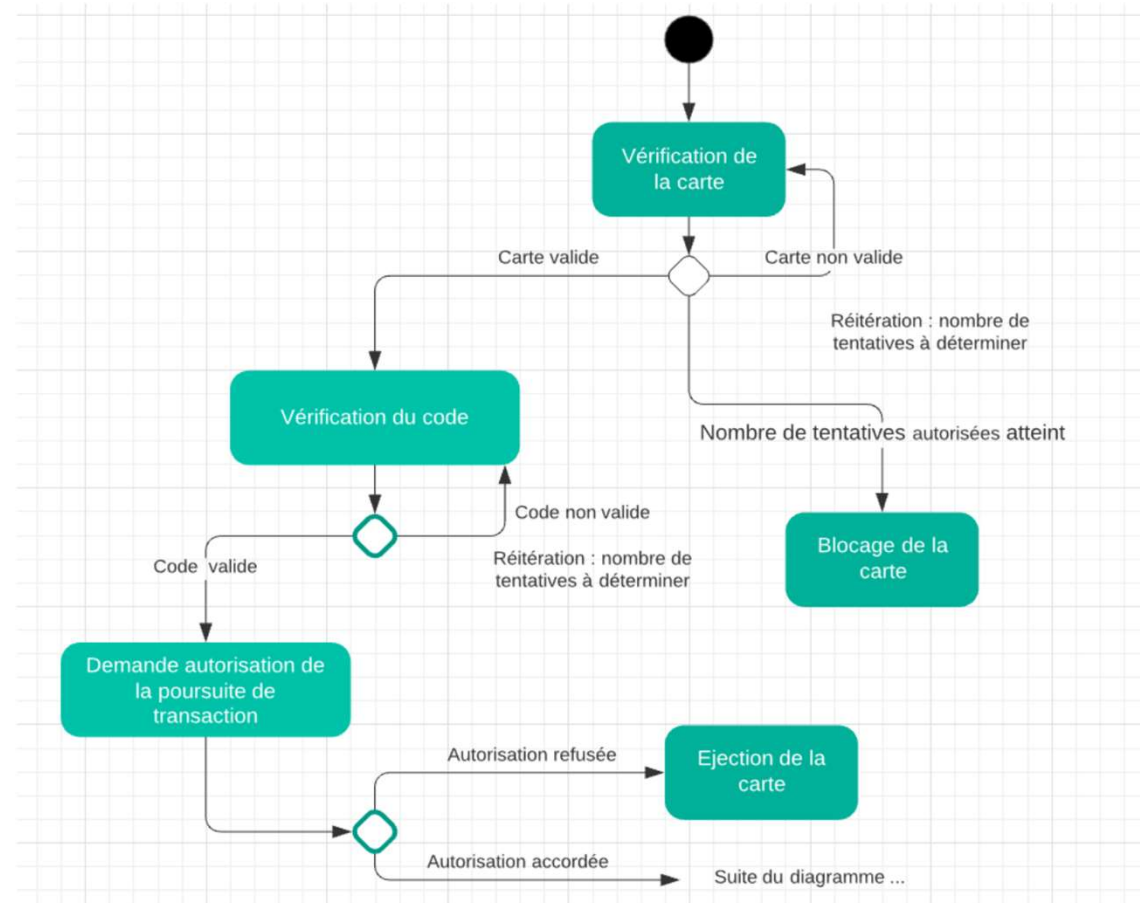


- **boucle** : cette boîte symbolise les tâches qui s'exécuteront de façon récurrente



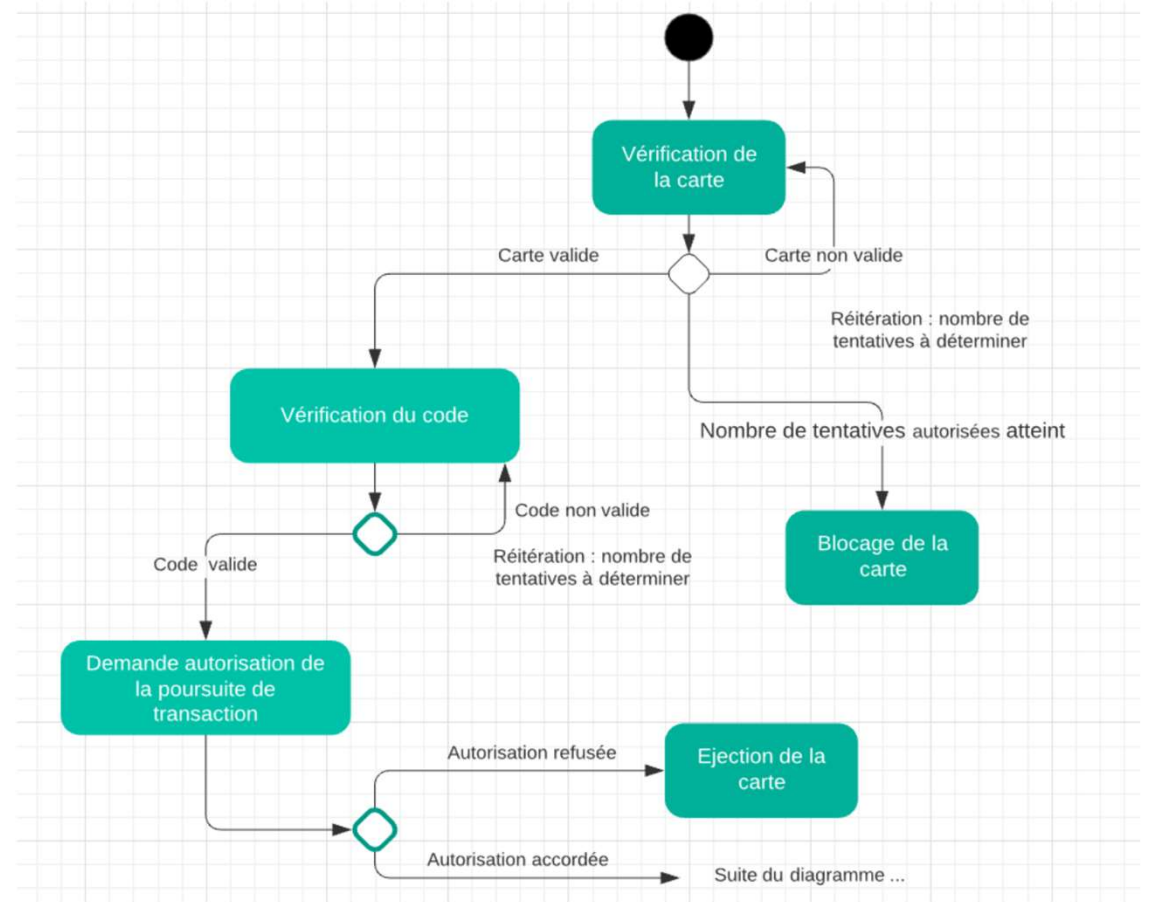
Les définitions sont communes à tous les outils de conception UML, mais la représentation graphique peut différer d'un outil à l'autre.

- Le diagramme d'activité présente les **actions** effectuées dans un cas d'utilisation et affiche le flux de travail d'un point de **départ** à un point d'**arrivée** en détaillant **tous les chemins possibles**
- Il est souvent comparé au terme **organigramme**
- Le parallèle est possible avec un **modèle conceptuel des traitements (MCT)** de la méthode Merise



On y trouve :

- des **arcs** (circuits, flux) symbolisés par les flèches orientées
- des **nœuds** (divers types) :
 - les nœuds **début** et **fin** (points noirs)
 - le **nœud d'action** : symbolisé par un rectangle aux bords arrondis, c'est un nœud exécutable qui représente une action, une transformation ou un calcul. Par exemple, la vérification du code de la carte est un nœud d'action.



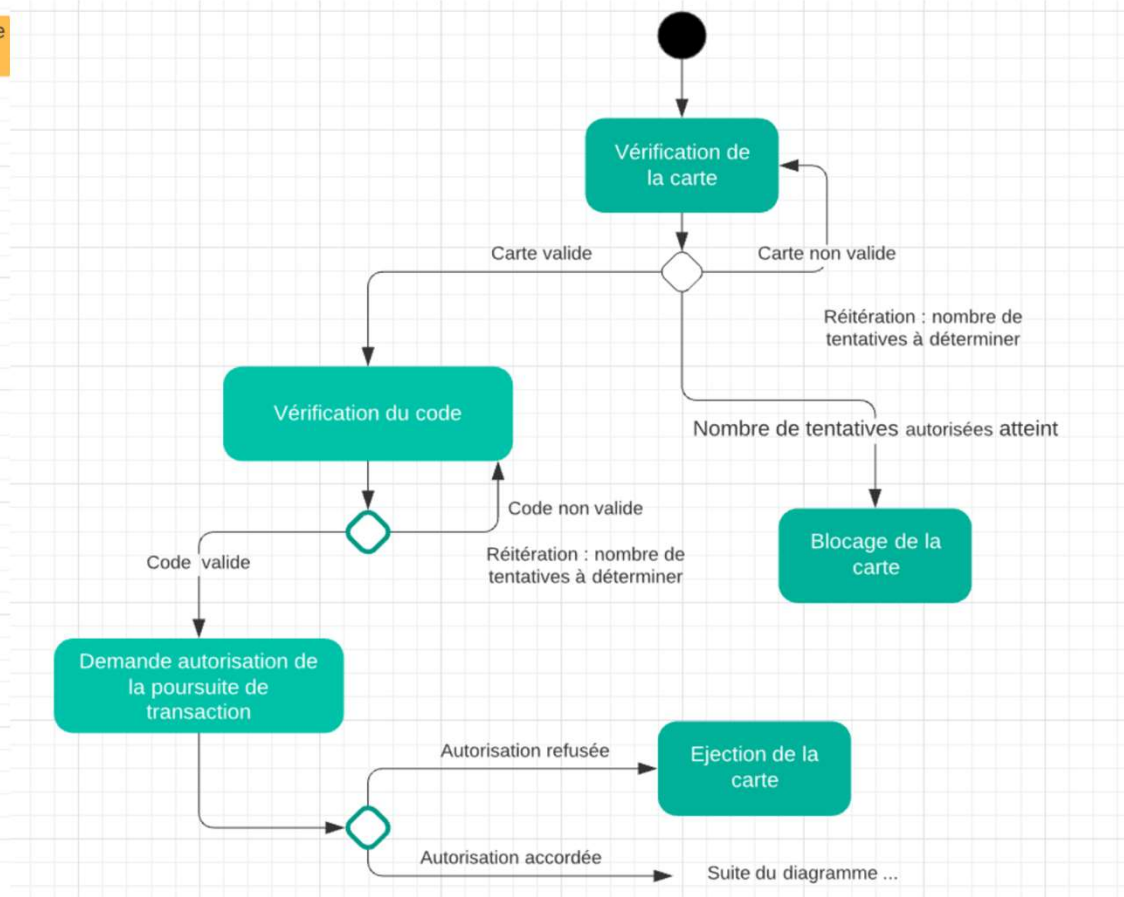
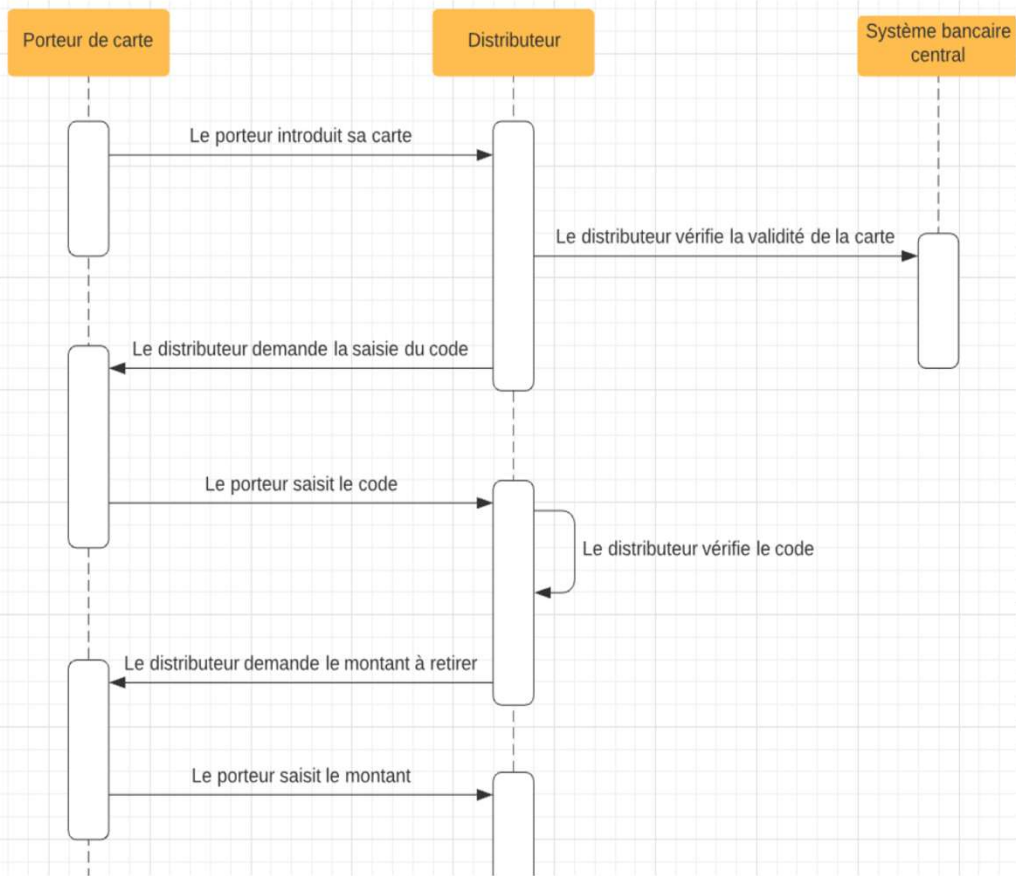


Diagramme de Classes

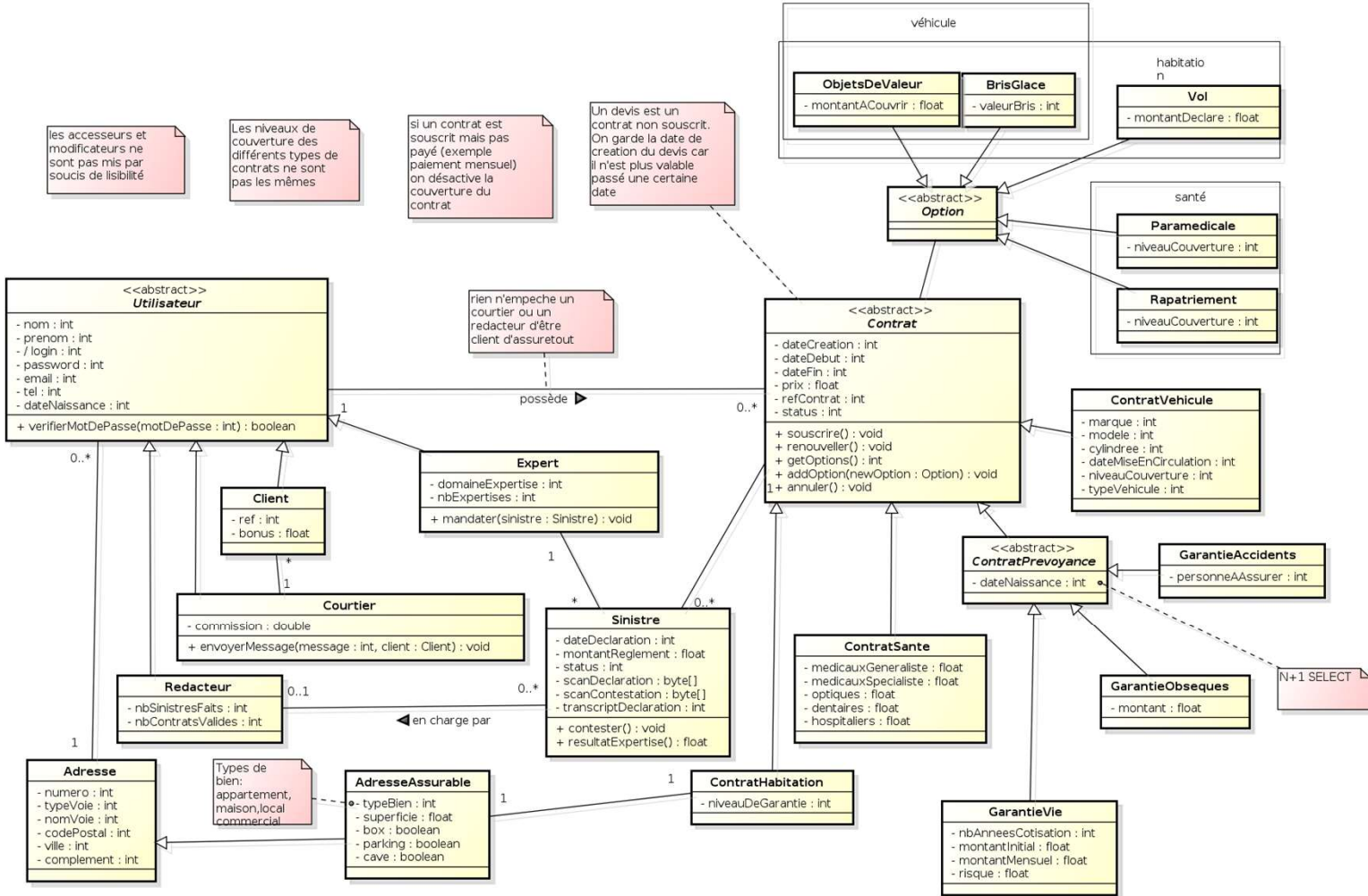
- un **diagramme de classes** représente un ensemble de classes et leurs relations
- c'est une **conception statique** du système d'information
- la **classe** est la description **abstraite** d'un ensemble d'**objets**.
- par analogie avec la poterie ou la sculpture, la **classe** est le **moule** et l'**objet** est la **sculpture** fabriquée à l'aide du moule
- une classe est définie par un **nom** et contient :
 - des **attributs**
 - des **méthodes** (aussi appelées opérations)
 - des **responsabilités**

DIAGRAMME DE CLASSES : associations

Les **associations** expriment une relation entre des instances de classes.

Elles sont décrites par :

- un **nom**
- un **sens** de lecture
- des règles de gestion, aussi appelées **cardinalités**
- des **rôles** (et leur visibilité)



Historique

1950 – 1960

- apparition des premières organisations de fichiers (SGF : Systèmes de Gestion de Fichiers) et les méthodes d'accès à ces fichiers : accès séquentiel, direct, séquentiel indexé.
- les SGF sont généralement incompatibles entre eux et la notion même de structure de fichiers n'est pas uniforme, ce qui rend les applications peu portables
- les langages de l'époque sont des langages procéduraux pour lesquels la gestion d'application multi-fichiers devient rapidement pénible, la programmation à l'aide d'outils non conçus initialement pour traiter des fichiers est donc lourde et contraignante => il faut donc envisager un nouveau modèle de fonctionnement.

Historique

1962 – 1963

- apparition du concept de Base de Données. Le terme « base de données » apparaît pour la première fois en 1964 dans le titre d'une conférence aux Etats-Unis : « *Development and Management of a computer-centered data base.* »
- on s'intéresse pour la première fois au traitement de gros volumes d'informations avec un souci d'efficacité des accès. Les premières règles sont établies :
 - le SGBD est un système de stockage de la base de données qui en assure la maintenance
 - les données sont organisées en base de données, ensemble de données logiquement liées
 - les données sont partageables (accès concurrents) entre plusieurs utilisateurs

Historique

1965 – 1970

- conception des SGBD de 1ère Génération (modèles hiérarchique et réseau)
IMS d'IBM (hiérarchique)
- IDS de General Electric (réseau)

Historique

1970 – 1985

- 2ème génération des SGBD organisés sur le modèle relationnel, avec davantage de spécifications des moyens d'accès aux données et de leur contrôle
- ces premiers systèmes sont commercialisés dans les années 80, notamment :
 - MRDS de Honeywell diffusé par CII-HB
 - SQL/IDS d'IBM
 - INGRES de Relational Technology
 - ORACLE de Relational Software.
- apparition des premières méthodes de conception (MERISE)

Historique

Depuis nous avons du faire face à la prolifération de données plus variées : les textes, sons, photos, vidéos, animations, ..., et la technologie a été adaptée et / ou développée en conséquence :

- bases de données réparties
- bases de données orientées objets
- bases de connaissances et systèmes experts
- bases de données déductives
- génie logiciel et SGBD
- accès intelligent multimodal et naturel (langage naturel écrit, graphique, parole, etc.).

Historique

Nous sommes face à une évolution permanente du contexte technologique, hardware et software

Du côté logiciel, nous sommes passés de l'âge de pierre à une industrie très organisée avec :

- des ateliers de développement
- des langages de plus en plus puissants
- des générateurs de programmes (formes, reports, menus ...)
- des mécanismes assurant la cohérence du Système d'Information (référentiel, dictionnaire de données)
- des outils d'intégration continue

Historique

Nous sommes globalement passés :

- d'une informatique **centralisée** à une informatique **orientée services** aux utilisateurs, dans un **environnement** informatique **hétérogène**: machines / logiciels / humains
- d'une **absence totale de méthodes** à des méthodes d'analyse et de conceptions de systèmes informatiques qui privilégient désormais une approche horizontale basée sur le **principe de séparation des données et des traitements**

Historique

Les utilisateurs n'ont pas tous :

- les mêmes **besoins**
- la même **culture**
- les mêmes **responsabilités**
- la même **vision** des choses



Quelques dates importantes

- 1970 : IBM crée le **premier prototype de base de données** sous le nom de System/R, ainsi que son langage, le SEQUEL
- 1973 : en se basant sur le System/R d'IBM, l'université de Berkeley crée le projet RDBMS, duquel découleront Sybase, Informix, NonStop SQL.
- 1978 : IBM commercialise son System Relational (System R)
- 1978 : **Oracle V1**
- 1981 : création d'Informix, fondation de Borland, apparition de dBase II sur IBM
- 1984 : Oracle V 4 et premier portage sur PC, création de Sybase
- 1986 : Oracle V5 avec architecture client /serveur

Quelques dates importantes

- 1989 : apparition de [Postgres](#)
- 1991 : Microsoft [SQL Server](#)
- 1992 : Oracle V 7 (intégration des contraintes procédures et déclencheurs)
- 1992 : première version de [Microsoft Access](#)
- 1996 : première version publique de [MySQL](#)
- 1997 : Oracle V 8 (objet, Java)
- ... to be continued ...