

Systèmes d'Information Bases de Données Relationnelles L3 MIASHS - IDS

Guillaume Metzler

guillaume.metzler@univ-lyon2.fr



INSTITUT
DE LA
COMMUNICATION



Université de Lyon, Lyon 2, ERIC EA3083, Lyon, France

Automne 2020

Organisation du cours

Un cours décomposé en trois parties

- 21h de cours magistraux
- environ 9h de travaux dirigés
- environ 9h de travaux pratiques

Certaines heures de CM se transformeront en heures TD pour vous laisser du temps pour pratiquer les différentes notions

Evaluation

- Un contrôle continu d'une heure
- Une évaluation en TP ou projet

Sommaire

Contenu du cours :

- Introduction rapide sur les bases de données
- Langage SQL : présentation des fonctions
- Modèle relationnel de Codd : troisième forme normale, dépendance fonctionnelles, normalisations
- Formalisme Entité - Association : élaboration d'un schéma conceptuel, schéma de la base de données

On finira par une étude de cas pour la *modélisation conceptuelle des données* et sa traduction pour l'obtention d'un modèle physique, i.e. la construction de la base de données.

Avant propos

Le cours présenté se base sur

- les cours présentés par les intervenants des années précédentes
- un cours dispensé à l'UJM de Saint-Etienne par *Francois Jacquenet*
- un livre de Jean-Luc Hainaut, *Bases de Données et modèles de calcul* (Jean-Luc-Hainaut, 2005)

Introduction

Système d'information

Une définition ?

Un **système d'information** est l'ensemble des ressources (matériels, logiciels, données, procédures, ...) structurés pour acquérir, traiter, mémoriser et rendre disponible l'information sous de multiples formes (textes, sons, images, ...) dans et entre plusieurs organisations.

Aujourd'hui, le système d'information permet d'automatiser et de dématérialiser quasiment toutes les opérations incluses dans les activités ou procédures de notre vie quotidienne : personnelle ou professionnelle.

Si l'on doit résumer en quatre étapes, son rôle est de :

COLLECTER → STOCKER → TRAITER → DIFFUSER

Bases de données : Besoins

Des besoins dans le domaine de la gestion de l'information

- Décrire l'information
- Manipuler l'information
- Interroger la collection d'informations
- Exactitude et Cohérence
- Garanties tant en terme de fiabilité que de contrôle
- Confidentialité
- Efficacité

Besoin de pouvoir décrire

Décrire les données de l'application. Par exemple, pour la SNCF cela correspond aux *trains*, *trajets* ou encore *réservations*, *conducteurs*. Cette description doit être effectuée sans faire référence à une solution informatique particulière.

—> **Modélisation conceptuelle**

Elaborer une description équivalente pour le stockage des données dans la SGBD choisie

—> **Modélisation logique**

On utilisera ensuite un langage de description des données

Besoin de pouvoir manipuler

Créer la base de données initiales avec les données représentant le réseau SNCF

→ **Langage permettant l'insertion de données**

Créer au fur et à mesure les données sur les réservations. Modifier si besoin et éventuellement supprimer toute donnée déjà rentrée

→ **Langage de Manipulation de Données (insertion, modification, suppression) :SQL**

Besoin de pouvoir interroger

Répondre à toute demande d'information portant sur les données contenues dans la base.

Par exemple :

- Jean Lestrade a-t-il une réservation pour aujourd'hui ?
Si oui, donner les informations connues sur cette réservation.
- Quels sont les horaires des trains de Lyon à Strasbourg entre 9h et 10h le dimanche ?
- Donner les destinations au départ de Lyon sans arrêt intermédiaire.

→ **Langage de requête ou d'interrogation : SQL**

Besoin d'exactitude et de cohérence

Il faut pouvoir exprimer toutes les règles qui contraintent les valeurs pouvant être enregistrées de façon à éviter toute erreur qui peut être détectée.

Par exemple :

- Il ne faut jamais donner la même place dans le même train à deux clients
- Les arrêts d'un train sont numérotés de façon continue (il ne peut y avoir pour un train donné un arrêt n3 s'il n'y a pas un arrêt n2 et un arrêt n1)
- La date de réservation pour un train doit correspondre à un jour de circulation de ce train
- L'heure de départ d'une gare doit être postérieure à l'heure d'arrivée dans cette gare
- L'heure d'arrivée à un arrêt doit être postérieure à l'heure de départ de l'arrêt précédent

→ **Langage d'expression de contraintes d'intégrité**

Besoin de garanties

Il ne faut pas que les informations (par exemple, les réservations) soient perdues à cause d'un dysfonctionnement quelconque : erreur de programmation, panne système, panne d'ordinateur, coupure de courant, plantage des serveurs, ...

→ **Garantie de confidentialité**

Il ne faut pas qu'une action faite pour un utilisateur (par exemple l'enregistrement d'une réservation) soit perdue du fait d'une autre faite simultanément pour un autre utilisateur (réservation de la même place).

→ **Garantie de contrôle de concurrence**

Besoin de confidentialité

Toute l'information doit pouvoir être protégée contre l'accès par des utilisateurs non autorisés que ce soit

- en lecture
- en écriture

Interdire par exemple aux clients de modifier les numéros de trains ou les horaires ou leur réservation.

→ **Garantie de confidentialité**

Besoin d'efficacité

Le temps de réponse du système doit être conforme aux besoins :

- en interactif : pas plus de deux secondes
- en programmation : assez rapide pour assumer la charge de travail attendue (nombre de transactions par jour).

Usage de mécanismes d'optimisations et, éventuellement, répartition/duplication des données sur plusieurs sites

Base de Données

Les **Bases de Données** sont des collections de données qui sont structurées et cohérentes.

Les informations sont organisées de manières à être facilement triées, classées et modifiées par le biais d'un logiciel appelé **Système de Gestion de Base de Données** (SGBD).

Le travail effectuer, de conception et de structure de l'information, doit répondre de façon pérenne à un problème, la structure se doit être évolutive et **ne pas répondre uniquement à un problème ponctuel**.

Plus de précision

Le **Système de Gestion de Bases de Données** est un logiciel permettant de

- définir une représentation des informations
- stocker, interroger et manipuler de grandes quantités de données (plus que la mémoire vive)
- garantir la longévité des données
- garantir l'accessibilité de manière concurrente (plusieurs utilisateurs simultanés)
- assurer une très grande fiabilité

Composants logiciels d'une base de données

SGBD

- gère le niveau logique et physique de la base selon l'architecture ANSI-SPARC

Les outils frontaux L4G

- générateurs : de formes, de rapports, des applications
- intégrés au SGBD ou externes
Powerbuilder, Borland ...
- Interfaces WEB : HTML, XML, ...
- Interface OLAP & Data Mining
Intellinet Data Miner (IBM)

Utilitaires : chargement, statistiques, aide à la conception, ...

Qui utilise des Bases de Données

Les utilisateurs interactifs

- ils cherchent des informations sans connaître la Base de Données
- utilisent des interfaces (formulaire, web, ...)
- peuvent à la rigueur utiliser des langages tels que QBE

Les programmeurs d'applications

- construisent les interfaces pour les utilisateurs interactifs
- spécialistes de SQL

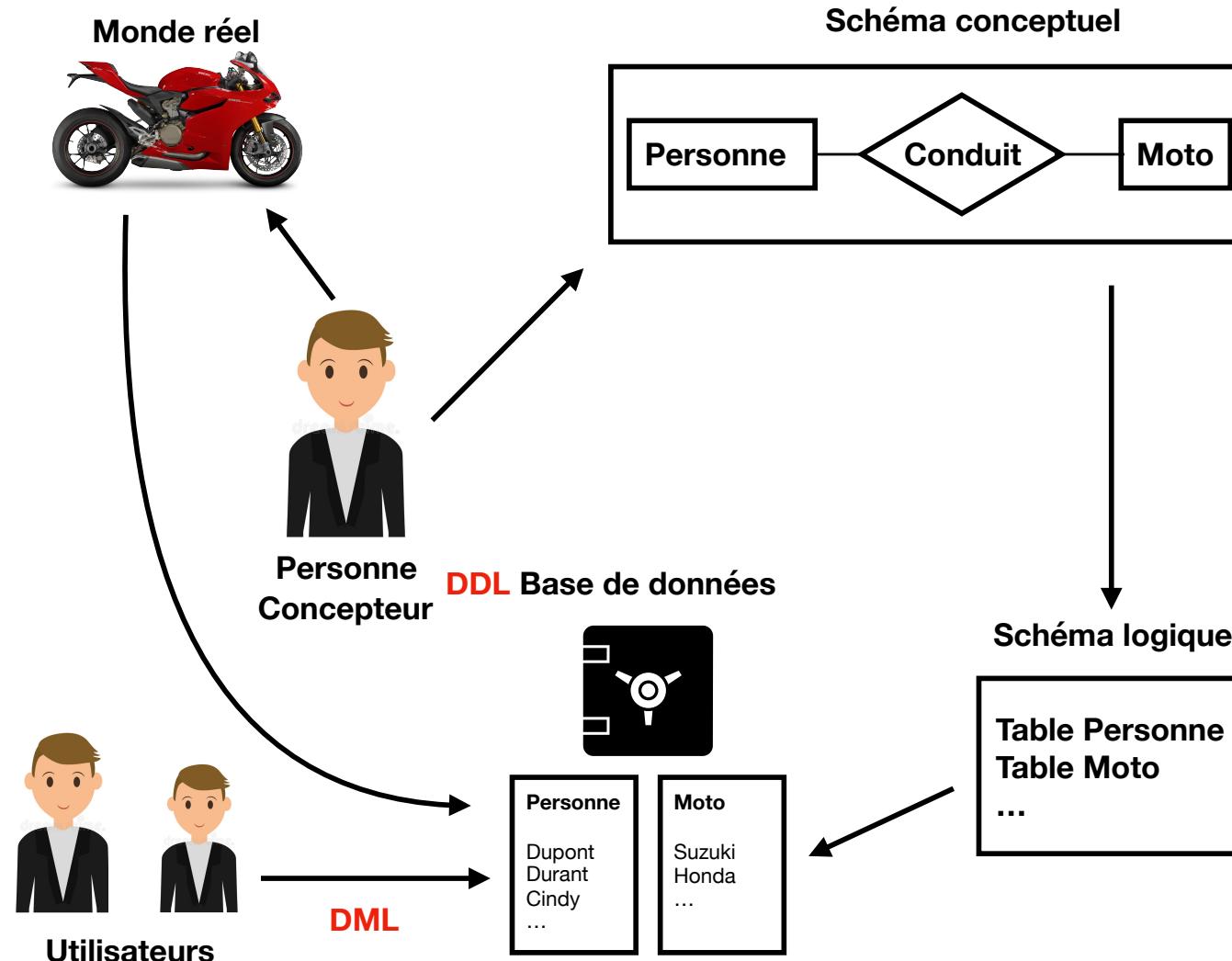
Les DBA (DataBase Administrators)

- modélisent les bases de données
- créés et maintiennent les bases de données
- ont la priorité sur tous les autres usagers
- peuvent être très bien payé ... car ils sont à la base du BI !

Formalisme Entité - Association

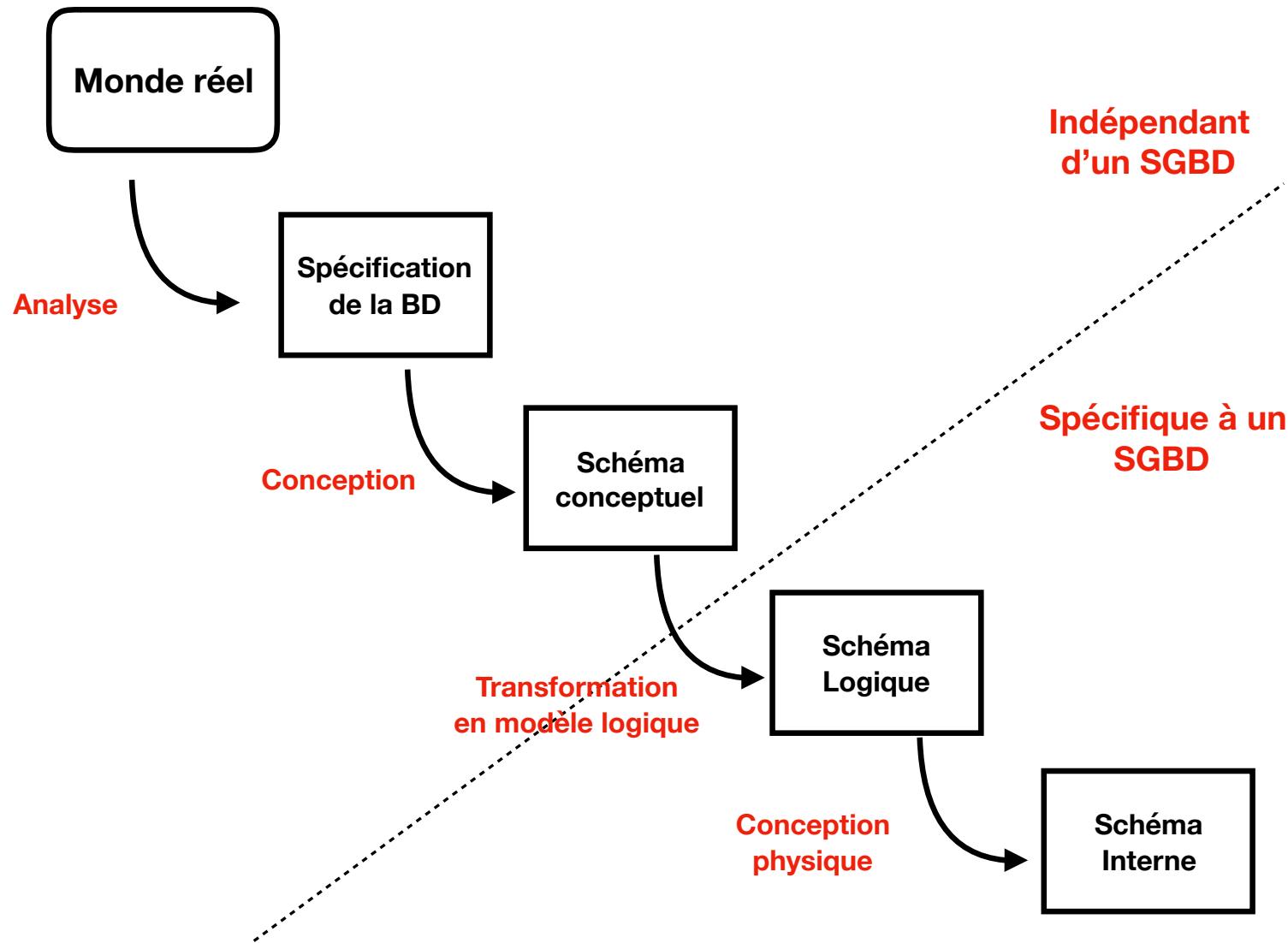
Cycle de vie d'une base de données

De multiples interactions



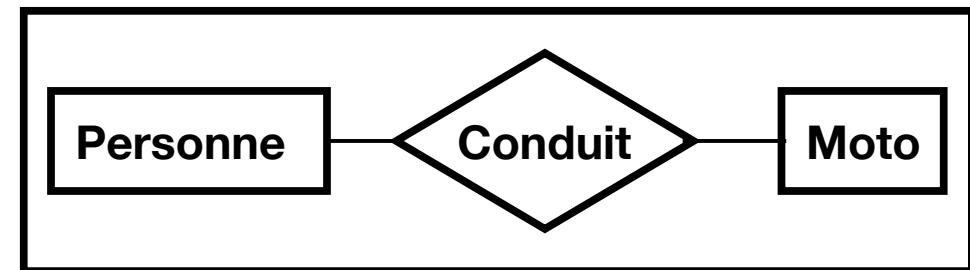
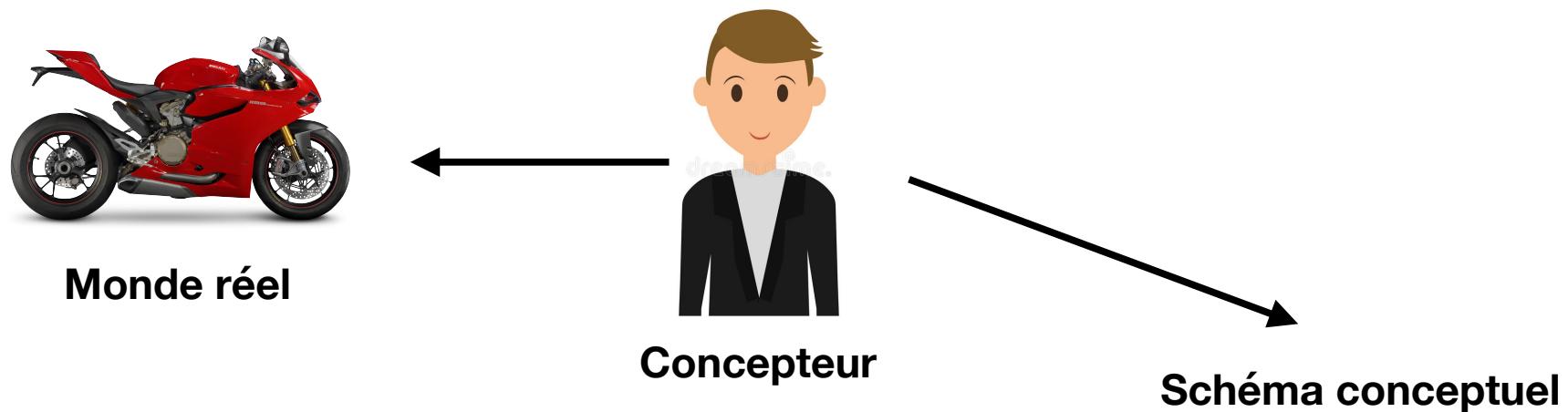
Cycle de vie d'une base de données

Une représentation plus schématique et détaillée



Focus

Ce qui va nous intéresser pour le moment c'est comment passer d'une *problématique réelle* au *schéma conceptuel* et comment écrire un tel schéma.



Rôle du concepteur

Le rôle du concepteur est fondamental ! C'est à lui que revient la charge de transcrire les observations du monde réel de façon abstraite en prenant en compte tous les liens et leurs caractéristiques. Il doit donc

- réfléchir à la structure de base (tables, attributs, relations) de façon à prendre en compte tous les aspects du problème
- avoir une représentation cohérente et qui corresponde à la réalité telle qu'elle est perçue par les utilisateurs.

C'est une étape d'autant plus importante qu'elle servira de fondations à ce qui suivra : cohérence - facilité des manipulations - logique.

C'est également une étape qui se veut indépendante de la technologie utilisée (on ne fera pas mention du langage SQL pour le moment !)

Rôle du concepteur

Pour faire simple, le concepteur doit :

- se mettre à la place de l'utilisateur
- rester fidèle à la réalité observée
- développer une représentation simple et compréhensible relative à l'application

Cela permet aussi de définir *un bon schéma conceptuel* : simplicité - fidélité - longévité - portabilité - compréhensibilité - ...

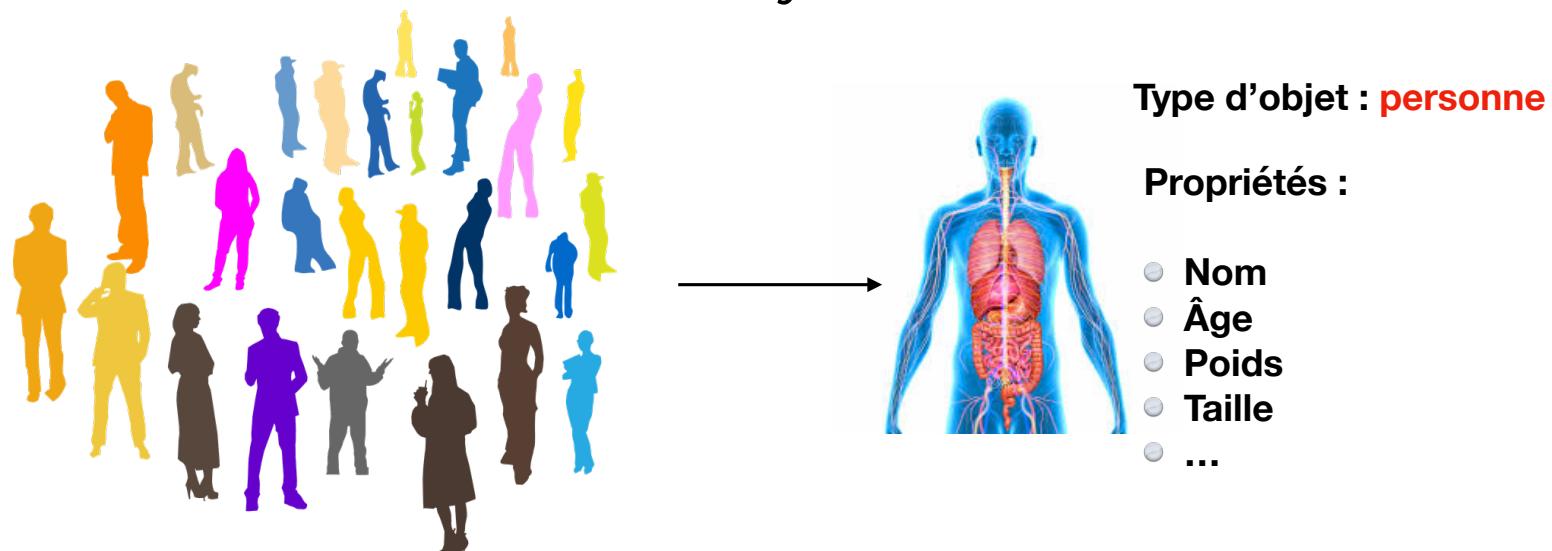
Elaborer un schéma conceptuel

Il faut tout d'abord analyser le monde réel : identifier les différents phénomènes à représenter dans la base de données

Les caractériser : définir leurs caractéristiques : contenu - structure - règles

Réalité perçue → Représentation

Faire abstraction des particularités : passer des *objets* aux *types ou classes d'objets*



De quoi s'agit-il ?

Lorsque l'on cherche à modéliser un système d'informations, on peut faire appel à deux *formalismes* :

- le formalisme entité/association
- le formalisme UML (*Universal Modelling Language*) : c'est un ensemble de notations qui permet de représenter divers aspects d'une application informatique (1990)

Mais on va se focaliser sur le modèle entité/association : traduction d'une problématique réelle à l'aide d'un graphique → passage schéma conceptuel. Pour ensuite aboutir au schéma logique.

A propos du modèle EA

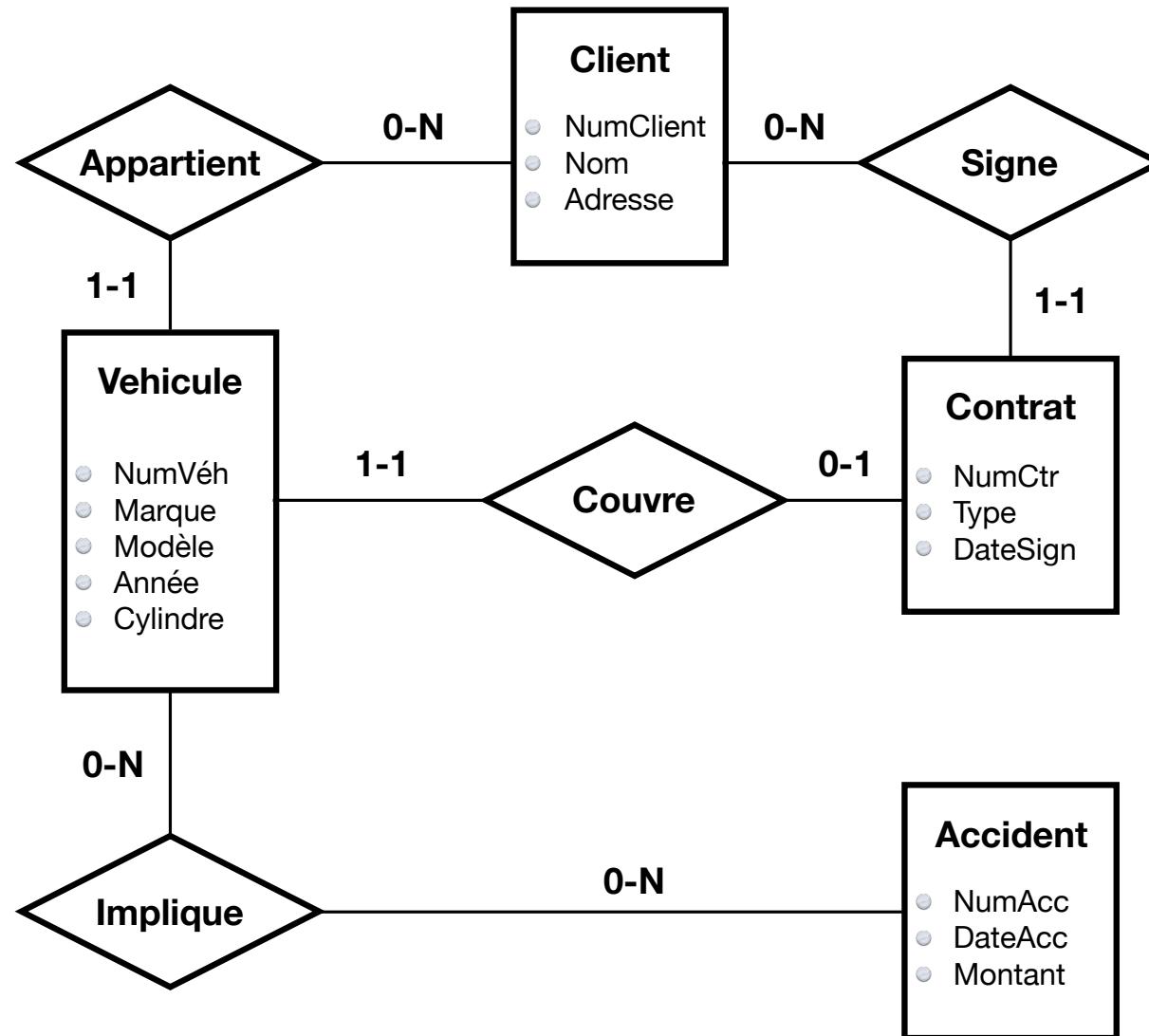
C'est certainement le modèle conceptuel le plus utilisé actuellement.

Il a été conçu en 1976 par Peter Pin-Shan Chen dans un article intitulé :

the Entity-Relationship Model: Toward a unified View of Data, ACM Transactions on Database Systems, volume 1, pages 9-36, 1976.

Il existe bien sûr d'autres modèles conceptuels : UML ou MERISE

Exemple



Questions

Que sont alors qu'une entité et une association ? Quel lien avec les bases de données

En fait c'est très simple mais un peu abstrait avouons-le !

Un schéma comme une collection de types :

- entités ↔ objets
- associations ↔ liens
- attributs ↔ caractéristiques

La base de données, quant à elle, contiendra les valeurs des attributs (ou propriétés) des différentes entités.

Définitions

Entité

C'est un objet qui peut être à la fois concret ou abstrait, ayant une existence propre. Il peut très bien s'agir de personnes, d'aliments, voitures ou encore de molécules chimiques. Ces entités admettent un identifiant, qui vont permettre de les distinguer, mais aussi une liste de *propriétés*, *attributs*. L'ensemble des entités est regroupé sous le nom de *type d'entités* lorsque plusieurs entités présentent des caractéristiques semblables.

Exemple :

Dans un contexte d'assurance automobile, on peut retrouver plusieurs classes - type d'entités : VOITURE - PERSONNE - CONTRAT - ACCIDENT

Représente une population et peuvent donc être de différentes natures

Définitions

Propriété, attributs

Il s'agit d'une valeur ou d'une modalité que peu prendre l'un des descripteurs. Elle permet de décrire l'entité. L'attribut peut lui même prendre plusieurs valeurs et permettra de distinguer les différents éléments de la population représentés par l'entité.

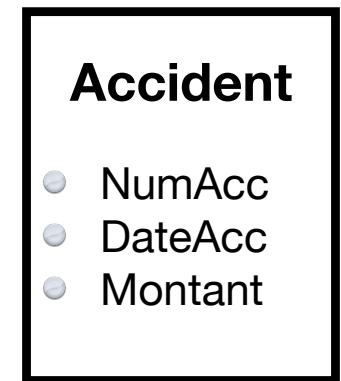
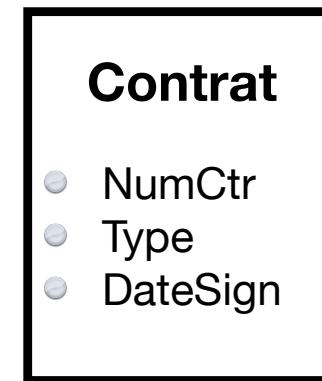
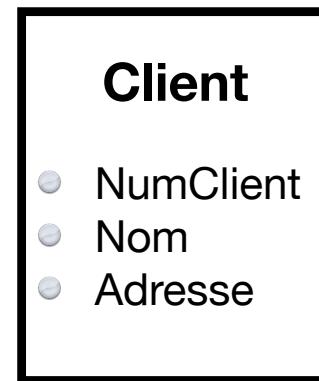
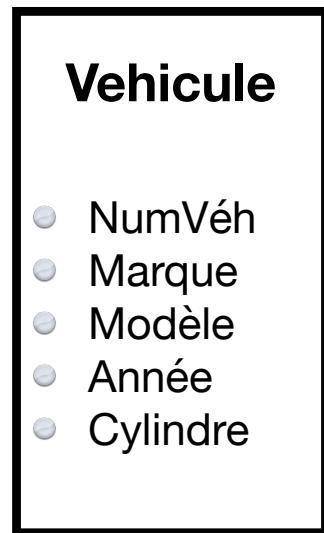
Exemple

CLIENT : *numéro de client* - *Nom* - *Adresse* VOITURE : *Numéro Véhicule* - *Marque* - *Modèle* - ...

Comment caractériseriez vous l'entité CONTRAT ou encore ACCIDENT ?
i.e. quels sont ses attributs ?

Exemple d'entités avec leurs attributs

Si on reprend notre exemple des contrats automobiles



→ on a 4 entités décrites par respectivement 5, 3, 3 et 3 attributs.

Nature des attributs

Un attribut peut être par nature **simple** ou **complexe**

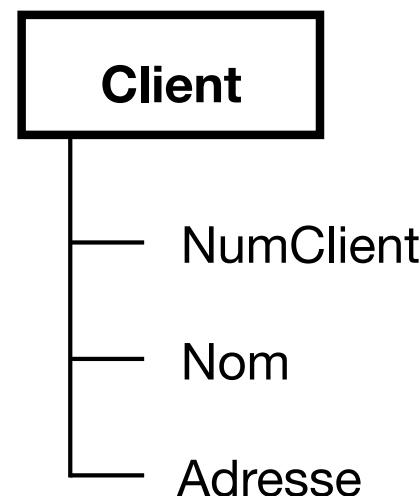
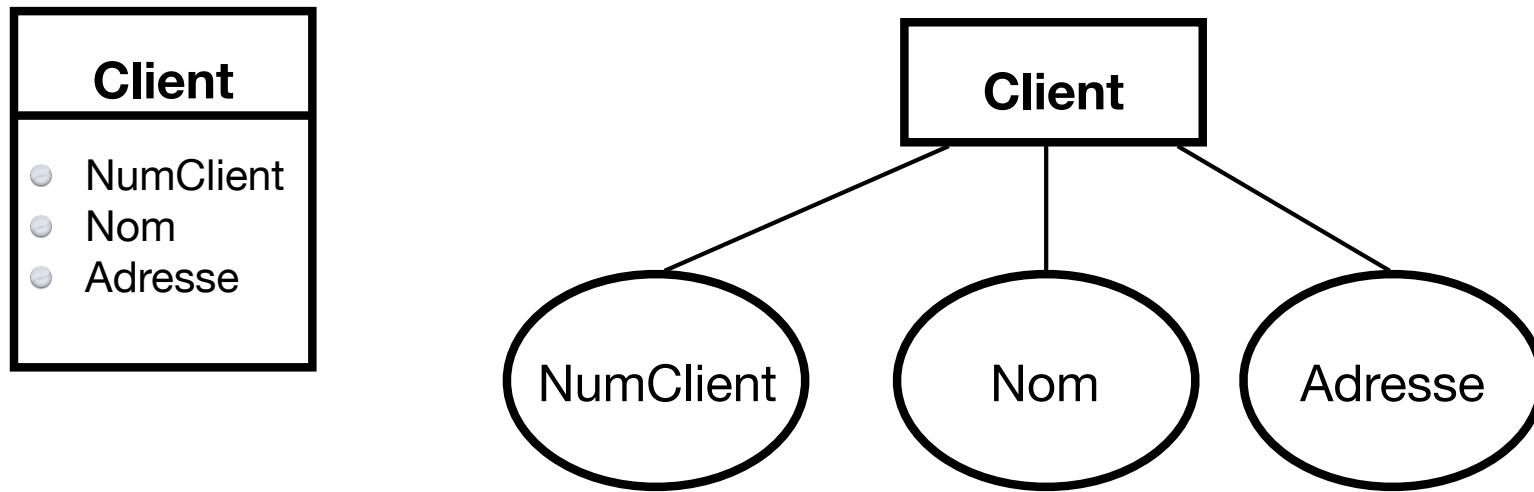
- **Simple** dans le sens où l'attribut est non décomposable : jour de l'année, prénom, année de naissance, lieu, numéro de immatriculation d'un véhicule, numéro de contrat.
Les valeurs prises sont *simples* et sont aussi bien des nombres que des chaînes de caractères
- **Complex**e dans le sens où il est décomposable : une date qui se décompose en jour - mois - année ou adresse postale complète.
Il se caractérise donc comme une composition d'attributs simples voire même complexe.

Nature des attributs

Un attribut peut également être **obligatoire** ou **facultatif**

- **Obligatoire** : au moins une valeur est attendue pour cet attribut, *i.e.* le cardinal de l'ensemble des valeurs possibles est au moins égal à 1.
Ex : Nom, Prénom
- **Facultatif** : il n'est pas nécessaire d'affecter une valeur, le cardinal de l'ensemble des valeurs possibles peut être nul.
Ex : montant de l'accident, salaire, numéro de téléphone, ...

Autres notations possibles



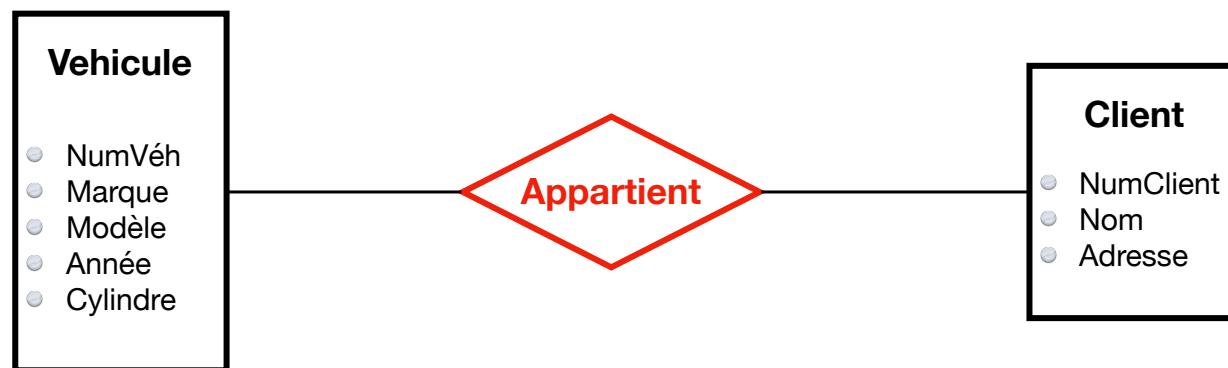
Définitions

Associations

Représentation d'un lien non orienté entre plusieurs entités.

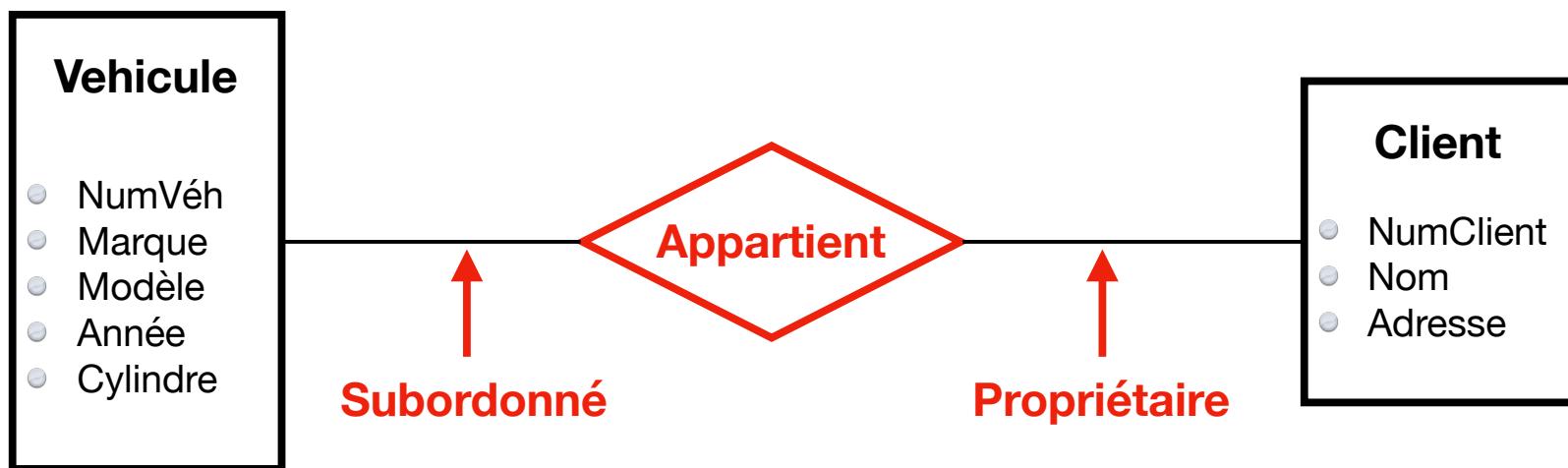
On désignera par type d'associations : la représentation d'un ensemble d'associations ayant la même sémantique et décrites par les mêmes caractéristiques

Exemple



Associations - liens

Rerenons notre association (binaire)

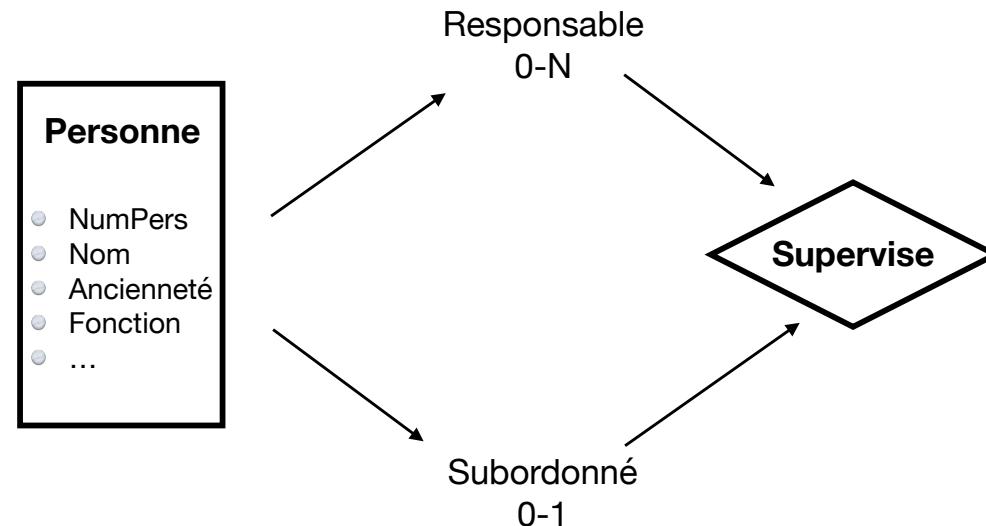


- Dans une association, chaque entité va jouer son un rôle précis
- Dans le cas d'une association binaire, on identifiera deux rôles

Associations - liens

Exemple d'une association binaire cyclique

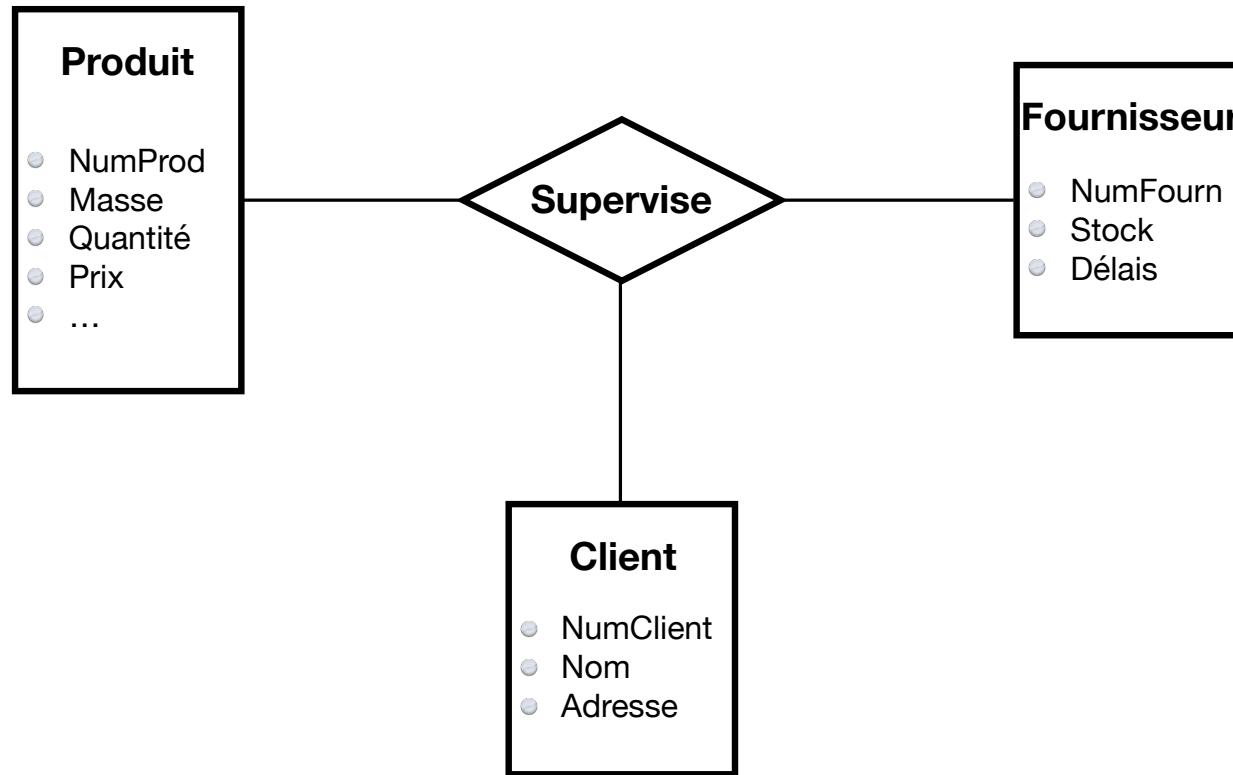
Il s'agit du cas où une entité est en relation avec elle-même, cela peut être le cas si l'on raisonne sur l'entité PERSONNE avec la relation *supervise* dans le cadre d'une entreprise.



Important : spécifier le rôle de chaque entité dans ce cas là

Associations - liens

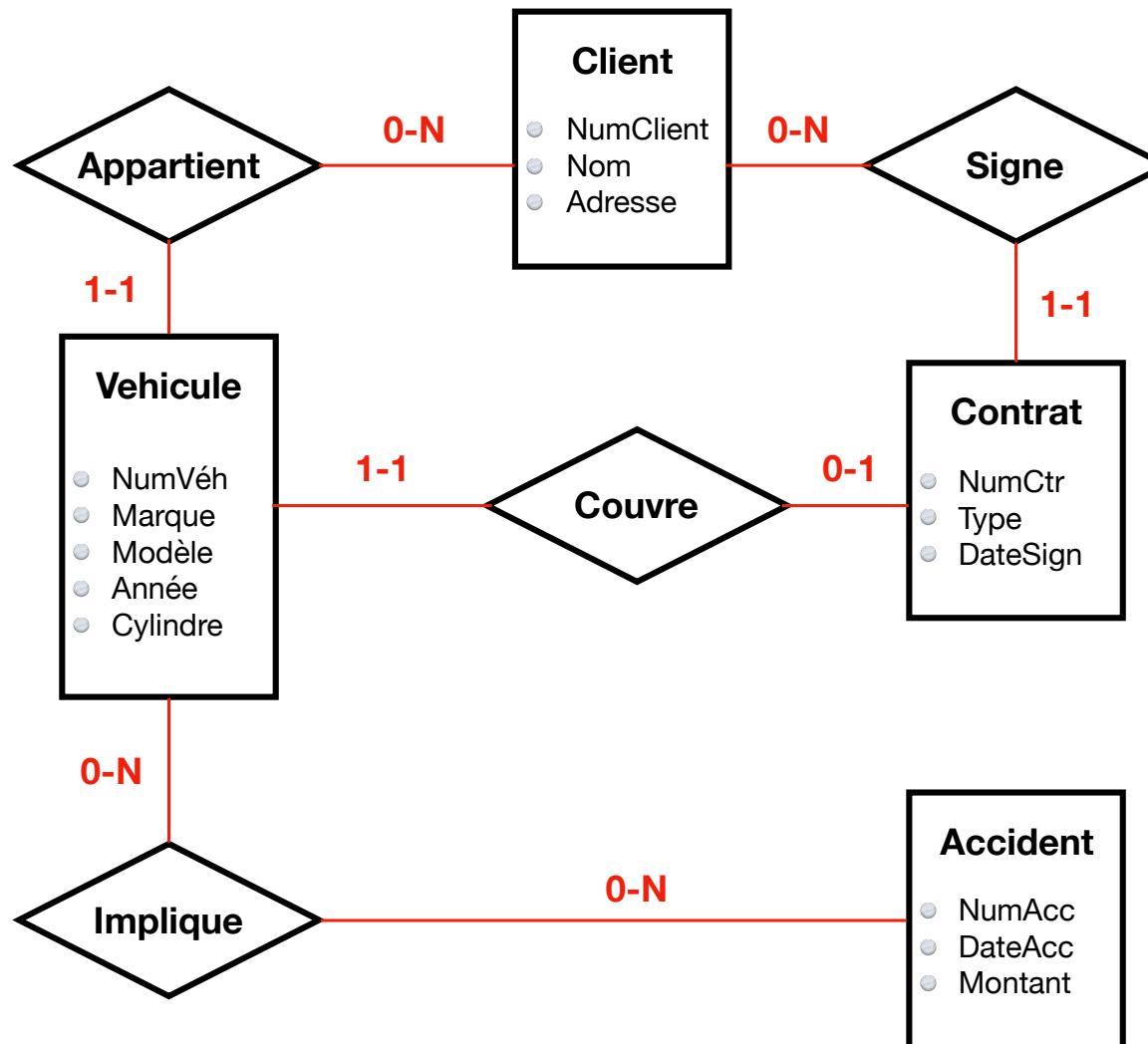
Exemple d'une association ternaire : commerce



On peut imaginer avoir des associations ternaires cycliques ou mettant un jeu encore plus d'entités, mais cela est rarement le cas en pratique

Un dernier point

Il nous reste un dernier point à définir : la cardinalité



Point vocabulaire

Cardinalité

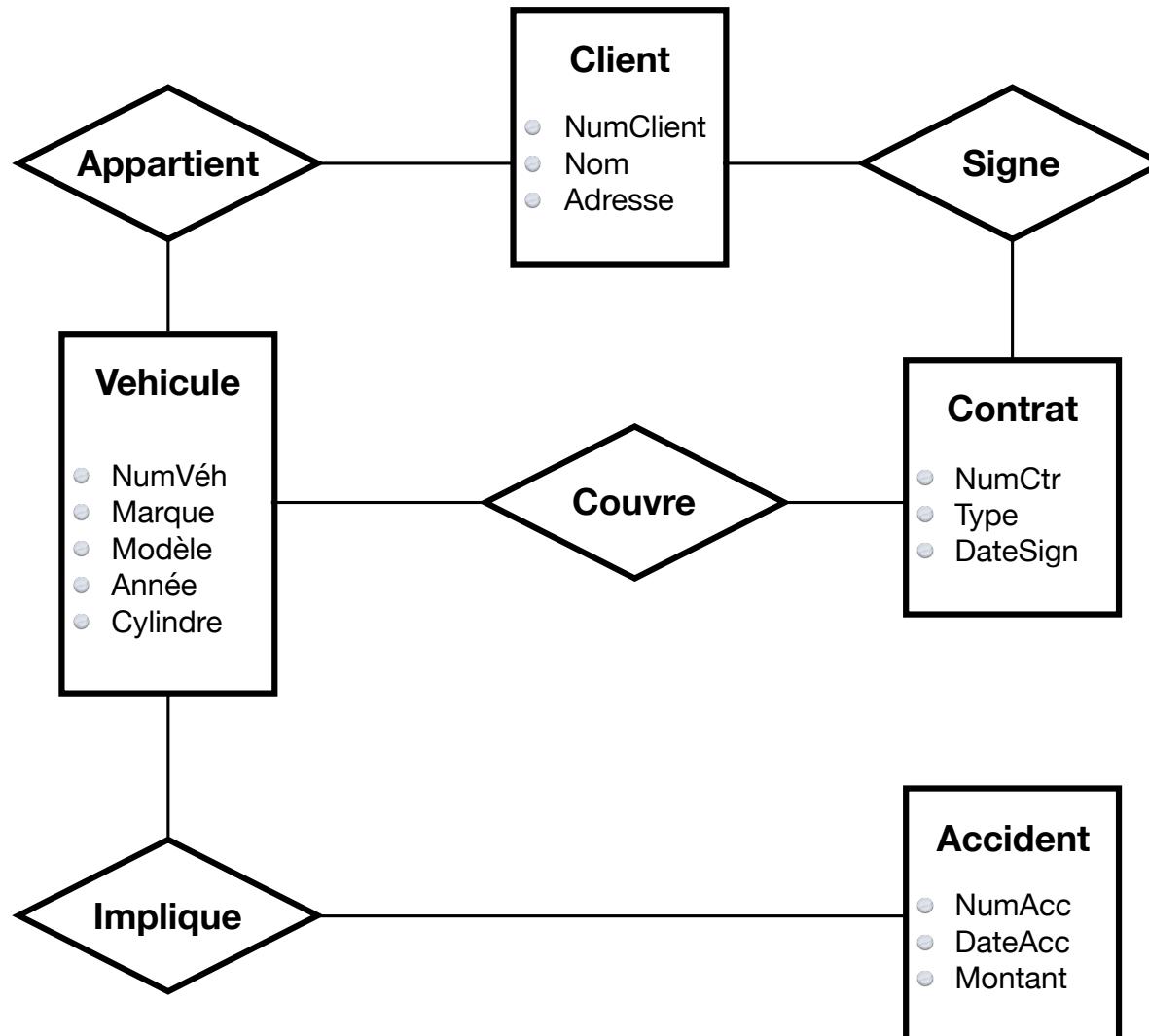
Il s'agit d'un indicateur qui montre combien, pour chacune des entités en relation, à combien d'associations chaque entité peut ou doit participer. Il existe trois types de *cardinalités* (également appelés types *d'associations*) : *associations 1-1* - *associations 1-N* - *associations N-N*.

De façon plus précise, on indique les participations minimales et maximales des entités à l'association

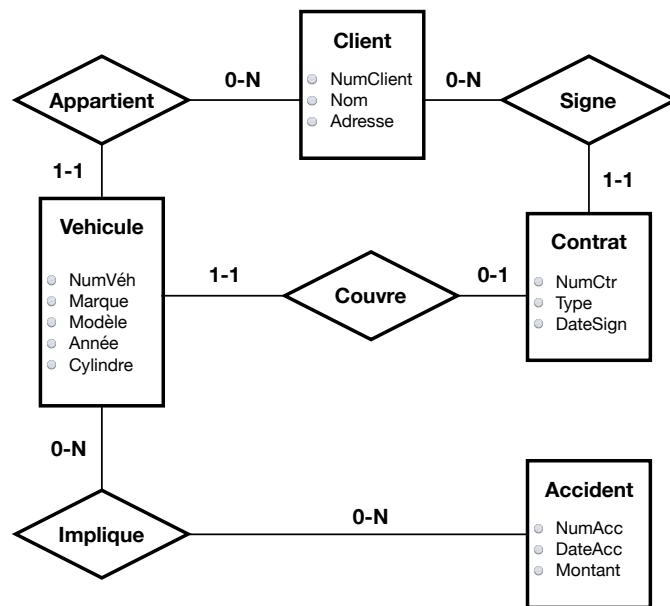
- 1-1 : une seule et unique relation
- 0-1 : au plus une relation
- 0-N : plusieurs relations possibles
- 1-N : au moins une relation
- M-N : entre M et N relations

Exemple

Indiquer la cardinalité des associations suivantes



Exemple



- un client **peut ne pas posséder de véhicule ou en posséder plusieurs**
- un véhicule appartient à **un seul et unique client**
- un accident peut **ne pas ou impliquer plusieurs véhicules**
- un véhicule est couvert par **une seul et unique assurance**
- un contrat est signé par **un seul et unique client**
- un contrat couvre **au maximum un véhicule**
- ...

Autres notations

Associations

0-1

.....

1-1

—

0-N

.....

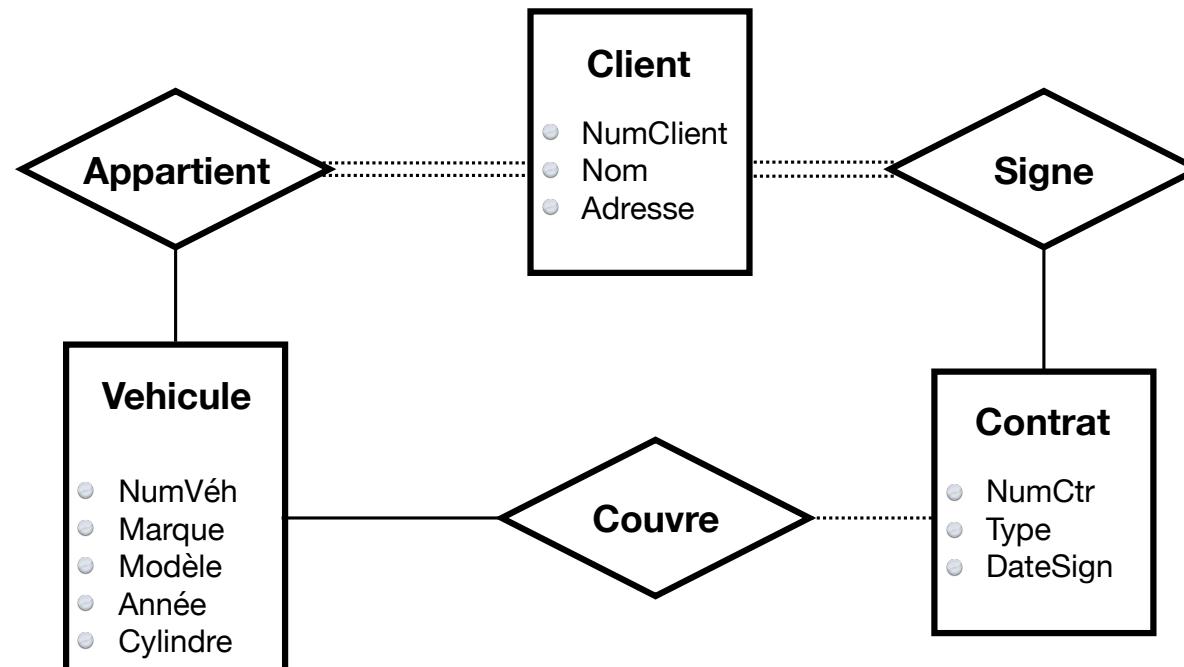
1-N

.....

N-M

==

Schémas



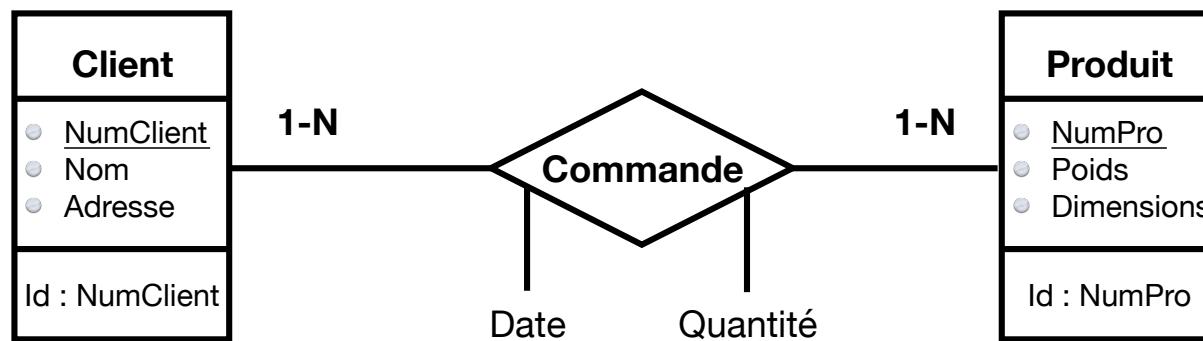
Associations porteuses de propriétés

Remarque

Dans des associations de type M-N, il est possible de caractériser l'association par des attributs pour en clarifier la nature.

Exemple

On peut spécifier la relation *Commande* entre *Produit* et *Client* en indiquant la date de la commande ainsi que la quantité commandée.



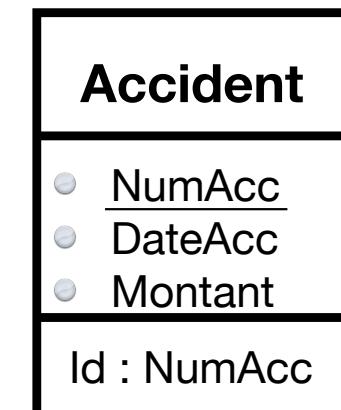
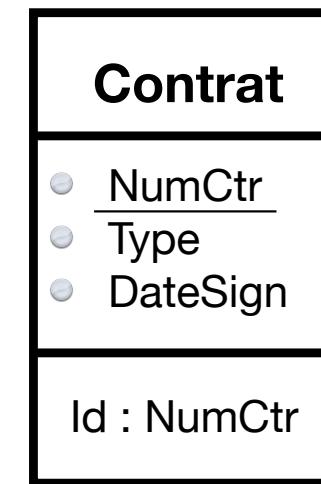
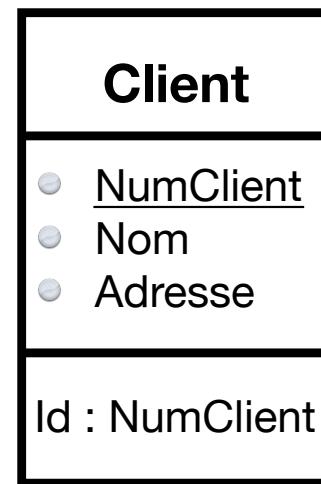
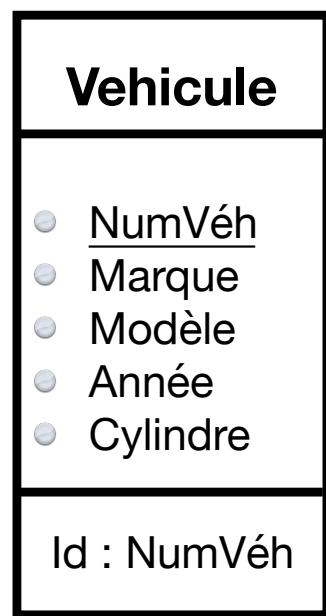
Identifiants

En général, mais cela n'est pas toujours le cas, un type d'entités est doté d'un attribut qui identifie les entités de ce type. Cet identifiant permet donc de **distinguer** les différentes entités d'un même type.

Le plus souvent, cet identifiant constitue la **clé primaire** de notre table, on l'appelle aussi **identifiant primaire**.

Notations : il est d'usage de faire précéder par *id* : l'attribut constituant l'identifiant primaire et de le souligner dans la liste des attributs.

Identifiants



Identifiants hybrides

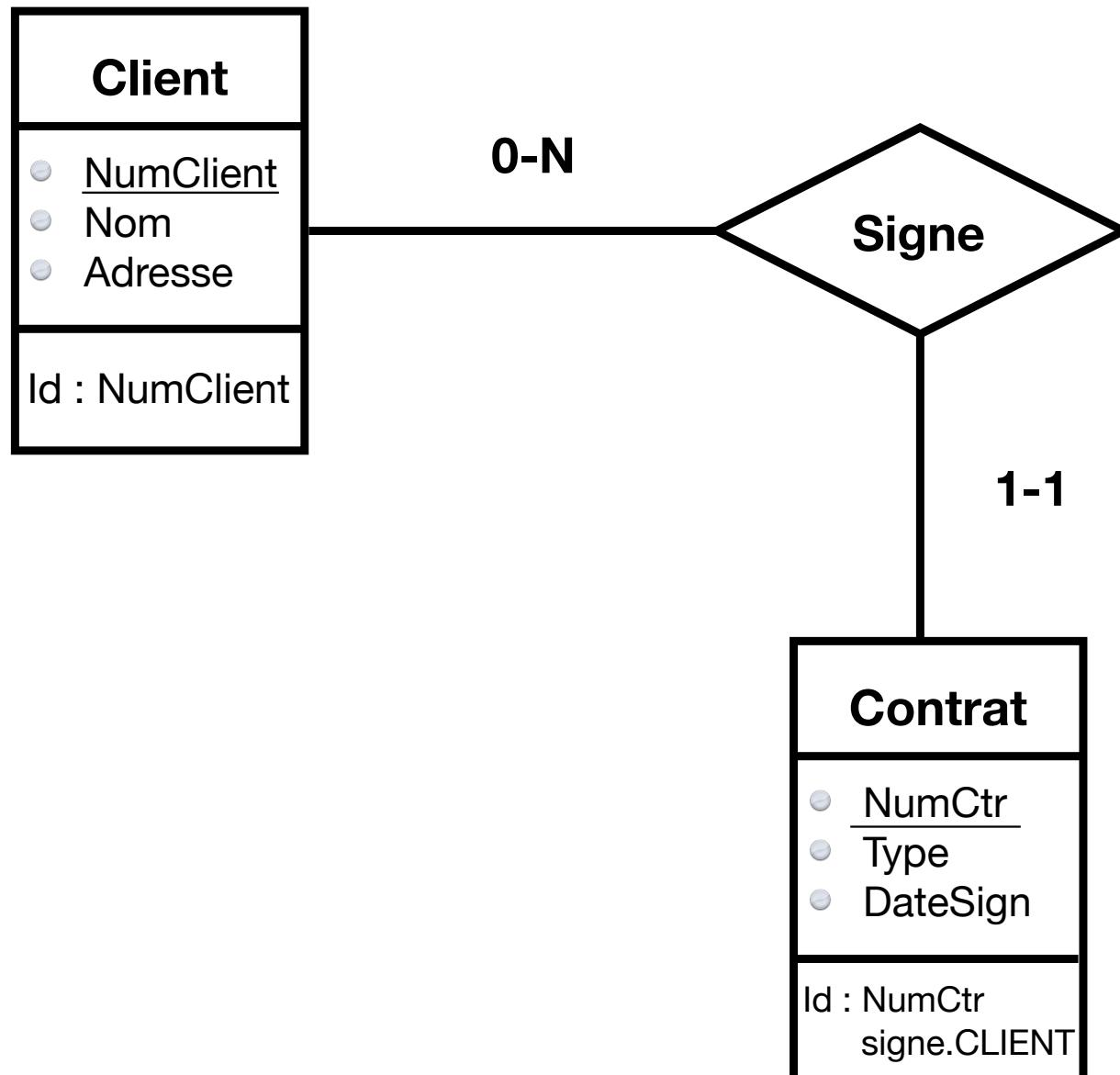
Reprendons le cas du type d'entités CONTRAT qui peut s'avérer plus complexe.

Un client peut potentiellement signer **plusieurs contrats différents** (imaginons qu'ils soient numérotés).

Si l'on souhaite donc **associer un contrat à un client particulier**, outre le numéro du contrat il est également nécessaire de **prendre en compte un autre attribut qui permettrait d'identifier clairement le client en question**.

Ces identifiants font références à une table précise (ici CLIENT) d'où la nature **hybride** —> identifiants issus de deux tables.

Identifiants hybrides



Quelques remarques - liens : identifiants/clé

Clé primaire

Il s'agit d'un attribut ou d'un ensemble d'attributs dont les valeurs permettent de distinguer les n-uplets les uns des autres (on retrouve cette notion d'identifiant).

Ex : *NumClient* est une clé primaire de l'entité CLIENT.

Clé étrangère

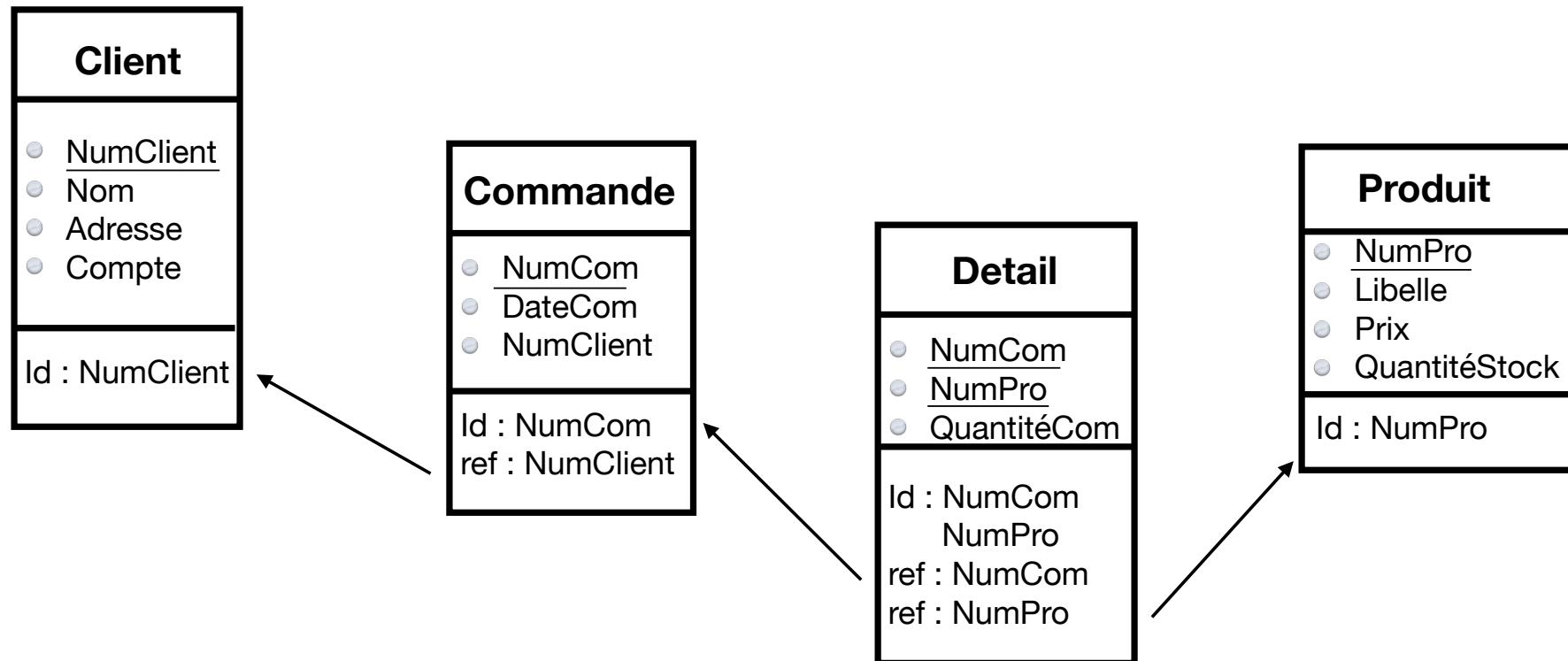
Il s'agit d'un attribut qui est une clef primaire d'une autre table ou autre type d'entités.

Elle joue un rôle important dans la notion de *contraintes référentielles* par exemple ou sont très pratiques pour créer une dépendance explicite entre deux tables.

La clé étrangère peut faire référence à n'importe quel identifiant d'une autre table (type d'entités), on choisira, la plupart du temps, son identifiant primaire.

Exemple

Un exemple graphique pour la notion de *références* (étrangères)



Remarques

La notion de références à une table nécessite d'imposer des règles logiques de structure et de contenu des tables qui sont liées entre elles.

Les références ne peuvent être effectuées n'importe comment et il est nécessaire de respecter une cohérence entre les tables.

Modifier une table doit entraîner des modifications dans les tables dépendantes et aussi prévenir de toutes modifications qui entraîneraient une incohérence (logique ou non) dans la base de données.

De telles contraintes peuvent également intervenir au sein d'une table en elle-même

→ **Travail sur la notion de contraintes d'intégrités**

Contraintes d'intégrités

Définition - Remarque

- Il s'agit d'un ensemble de règles définissant les états (CI statiques) et les transitions d'états (CI dynamique) possibles de la base de données
- Ces règles doivent être décrites explicitement (dans le langage approprié) si elles ne peuvent pas être décrites avec les concepts du modèle de données
- Une base de données sera qualifiée de cohérente si toutes les CI définies sont respectées par les deux de la base de données.

Regardons cela sur quelques exemples pour démystifier tout cela.

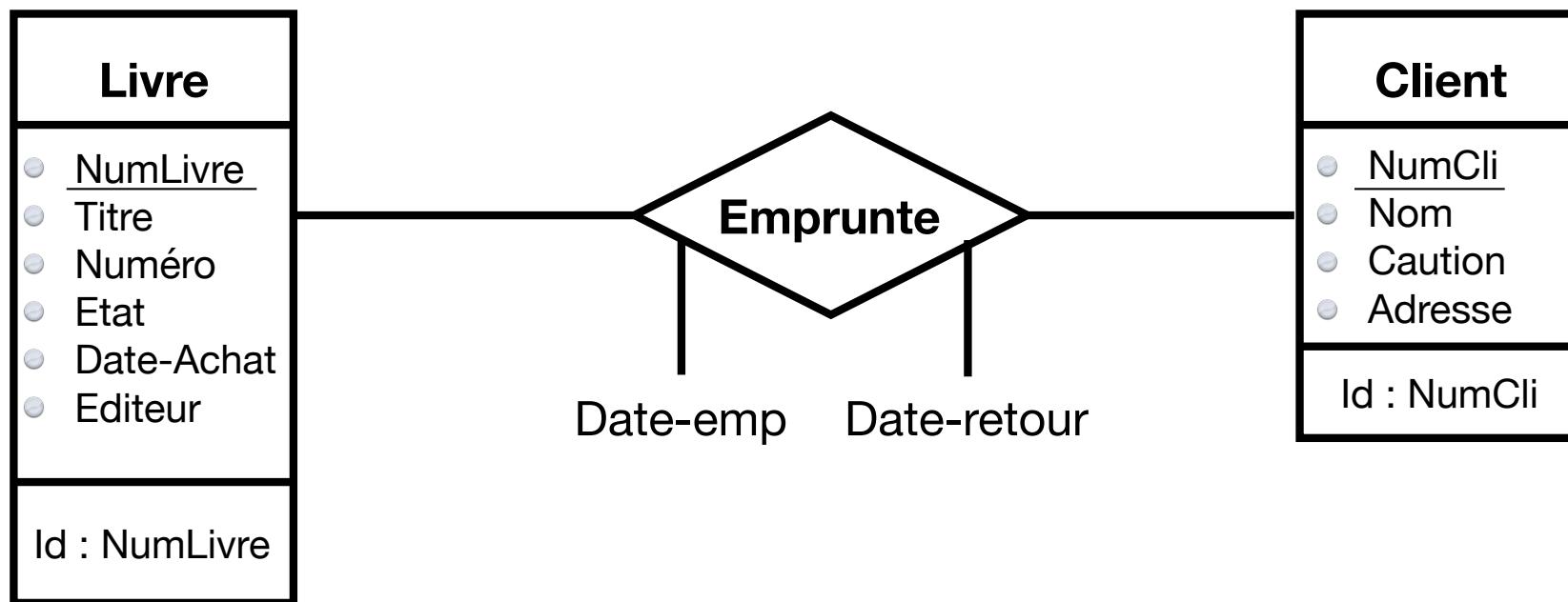
Quelques exemples

Il existe différentes contraintes portant directement sur les attributs :

- les contraintes d'unicité en liant avec la notion d'identifiant -> contraintes lors de la création ou de la modification d'un ligne.
- les contraintes de domaines : un attribut ne peut prendre ses valeurs que dans un champ de valeurs restreint
Ex : les notes à un examen sont comprises entre 0 et 20
- les contraintes référentielles : elles précisent que certaines colonne d'une table, appelées clé étrangère (cf. ce qui a été vu plus tôt dans le cours) doivent à tout instant, pour chaque ligne, contenir les valeurs qu'on retrouve comme identifiant primaire d'une ligne dans une autre table.

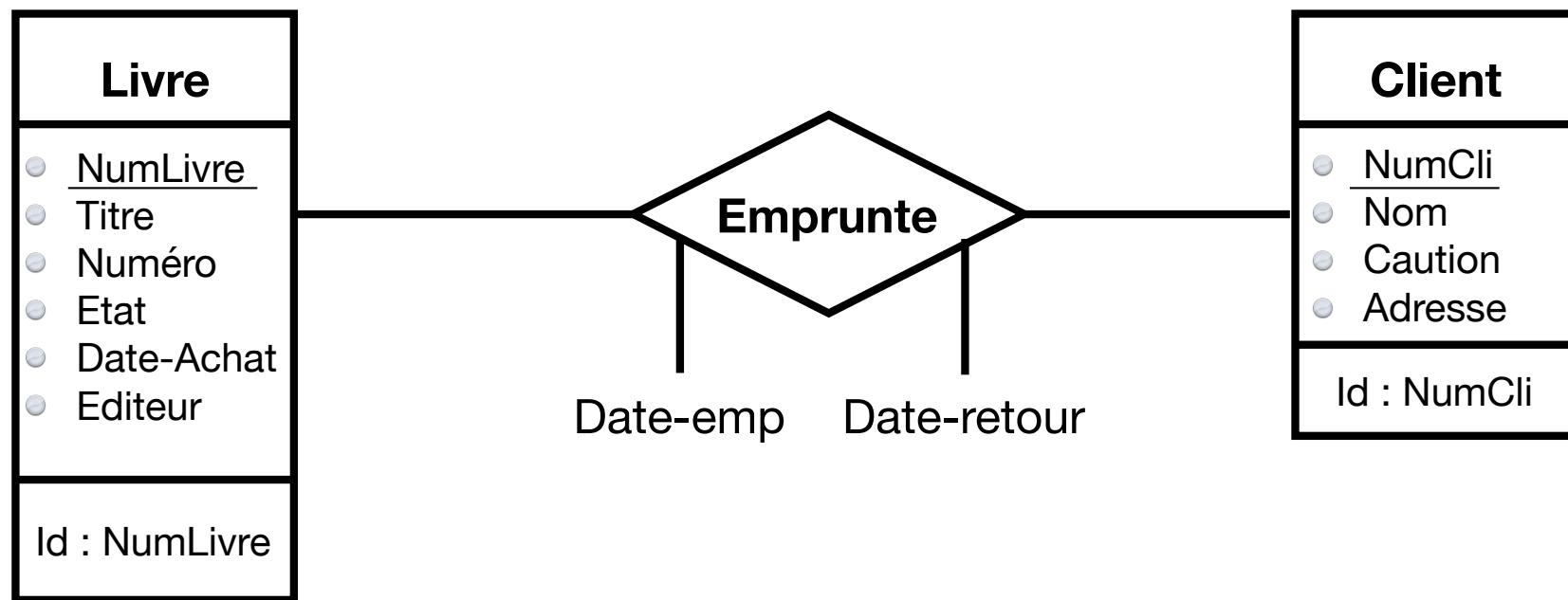
Quelques exemples

Considérons le modèle entité association suivant, citez moi au moins deux contraintes d'intégrités entre les attributs.



Quelques exemples

Considérons le modèle entité association suivant, citez moi au moins deux contraintes d'intégrités entre les attributs.



- Pour chaque occurrence d'*Emprunte*, si la date de retour existe, alors elle doit supérieure à la date d'emprunt
- Pour chaque occurrence de *Livre*, la date-achat doit être inférieure à la date d'emprunt de toutes les occurrences d'*Emprunte* qui lui sont liées

Contraintes statiques

Définition

Une contrainte d'intégrité statique précise une propriété que les données doivent vérifier à tout instant. Les contraintes d'unicité, référentielles et de colonnes obligatoires en sont les exemples les plus fréquents. Ces derniers sont d'ailleurs gérés automatiquement par la plupart des systèmes de gestion de bases de données.

Exemple :

```
create trigger MAX-5-DET
before insert or update NCOM on DETAIL
for each row
begin
    if (select COUNT(*) from DETAIL where NCOM=new.NCOM) = 5
        then abort(); end if;
end;
```

Contraintes dynamiques

Définition

Une contrainte dynamique indique quels changements d'états sont valides. On ne peut détecter une violation lors d'une modification qu'en connaissant les deux états avant et après l'évènement. Elle spécifie donc les changements d'états valides.

Exemple :

- On ne peut augmenter le prix de plus de 5%
- une personne peut passer du statut "célibataire" au statut "marié(e)" ou "pacsé(e)" mais pas aux statuts "divorcé(e)" ou "veuf(ve)"
- On ne peut ajouter les détails d'un produit qui se retrouvent en rupture de stock
- Un nouveau client d'un magasin ne peut se trouver que dans un nombre restreint d'état

Contraintes dynamiques : exemples

Dans l'exemple suivant, le déclencheur protège les produits contre toute modification du prix qui dépasserait 5%

```
create trigger MAJOR
before update of PRIX on CLIENT
for each row
begin
    if new.PRIX > (old.PRIX * 1.05)
    then abort(); end if;
end;
```

Exercice : Ecrire un déclencheur qui autorise la suppression d'un client que s'il n'a plus envoyé de commandes depuis le 1er Janvier 2020

Les étapes pour établir un modèle entité association

Démarche à suivre :

- Identifier les types d'entités
- Identifier les attributs
- Associer les attributs aux différents types d'entités
- Identifier les associations entre les types d'entités
- Identifier les attributs de chaque association
- Evaluer la cardinalité des associations

Il est important de bien lire l'énoncé, de bien analyser le contexte de votre étude afin d'identifier clairement les éléments importants qui pourront vous servir à la construction de votre modèle.

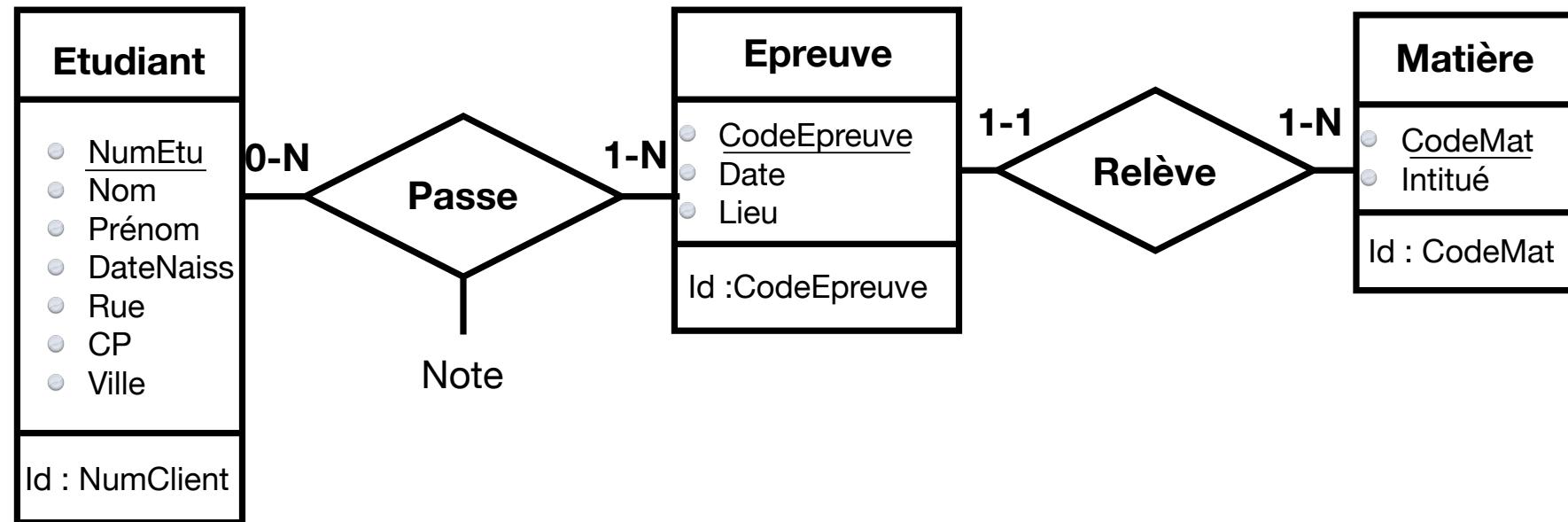
Exemple

Considérons une base de données relative à la notion d'Examen.

Construire le modèle entité association à partir de l'énoncé suivant

- les étudiants d'une université sont caractérisés par un numéro unique, leur nom, prénom, date de naissance, rue, code postal et ville
- ils passent des épreuves et obtiennent une note pour chacune
- les épreuves sont caractérisées par un code ainsi que la date et le lieu auxquels elles se déroulent
- chaque épreuve relève d'une matière unique. En revanche, une matière peut donner lieu à plusieurs épreuves
- les matières sont caractérisées par un code et un intitulé

Schéma Entité-Association

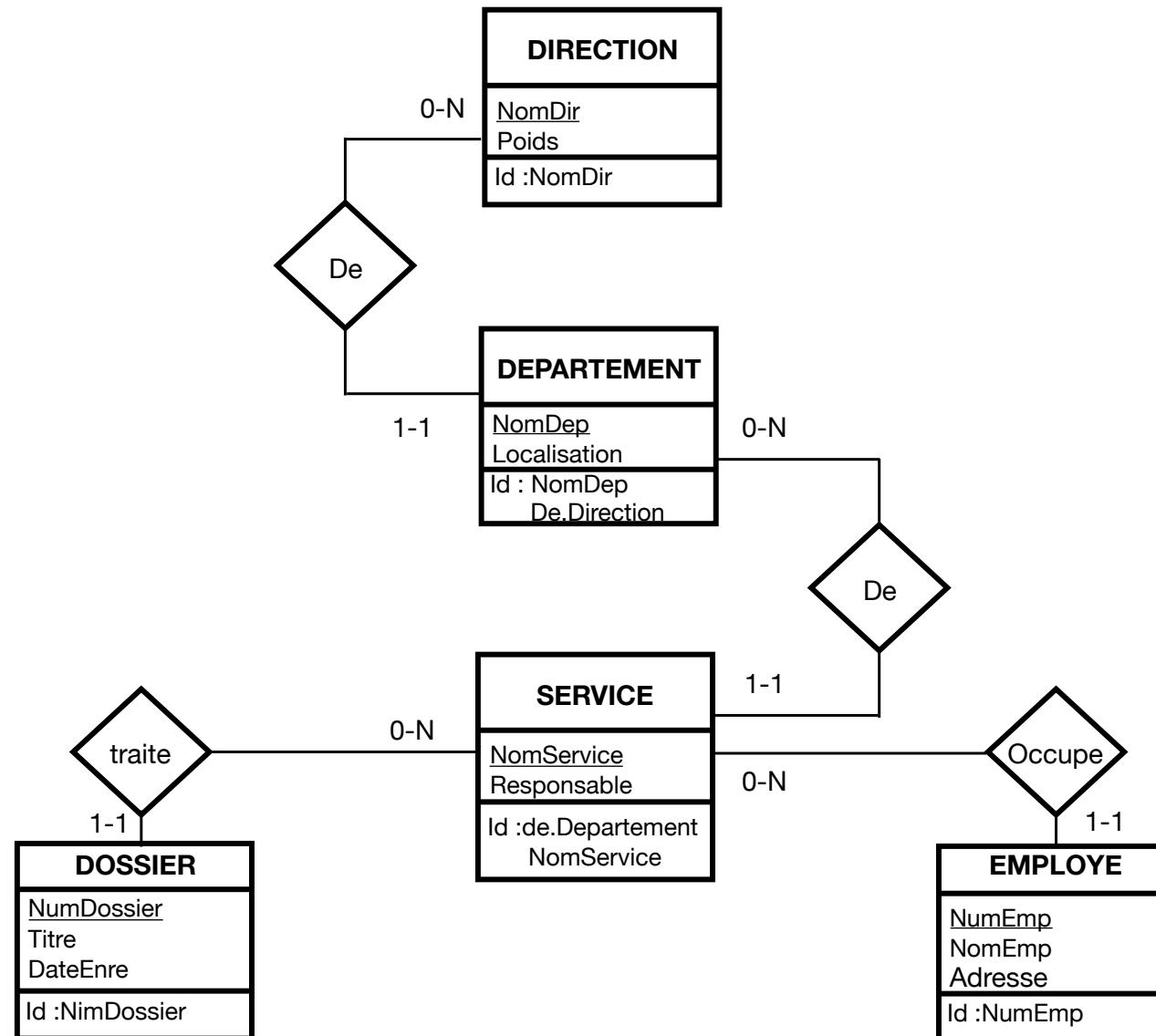


Exemple d'une structure administrative

Exemple : on considère un sous-ensemble d'une structure administrative. D'une direction (caractérisée par un nom identifiant et le nom de son président-directeur général) dépendent plusieurs départements (dotés chacun d'un nom identifiant dans sa direction et de sa localisation). Un département est découpé en services, dotés chacun d'un nom (identifiant de son département) et d'un responsable. Un service a la charge d'un certain nombre de dossiers identifiés par un numéro et dotés d'un titre et d'une date d'enregistrement/ Dans chaque service travaillent des employés identifiés par un numéro et caractérisés par leur nom et par leur adresse.

Ecrire le modèle entité-association lié à cet énoncé.

Schéma Entité-Association : Administration

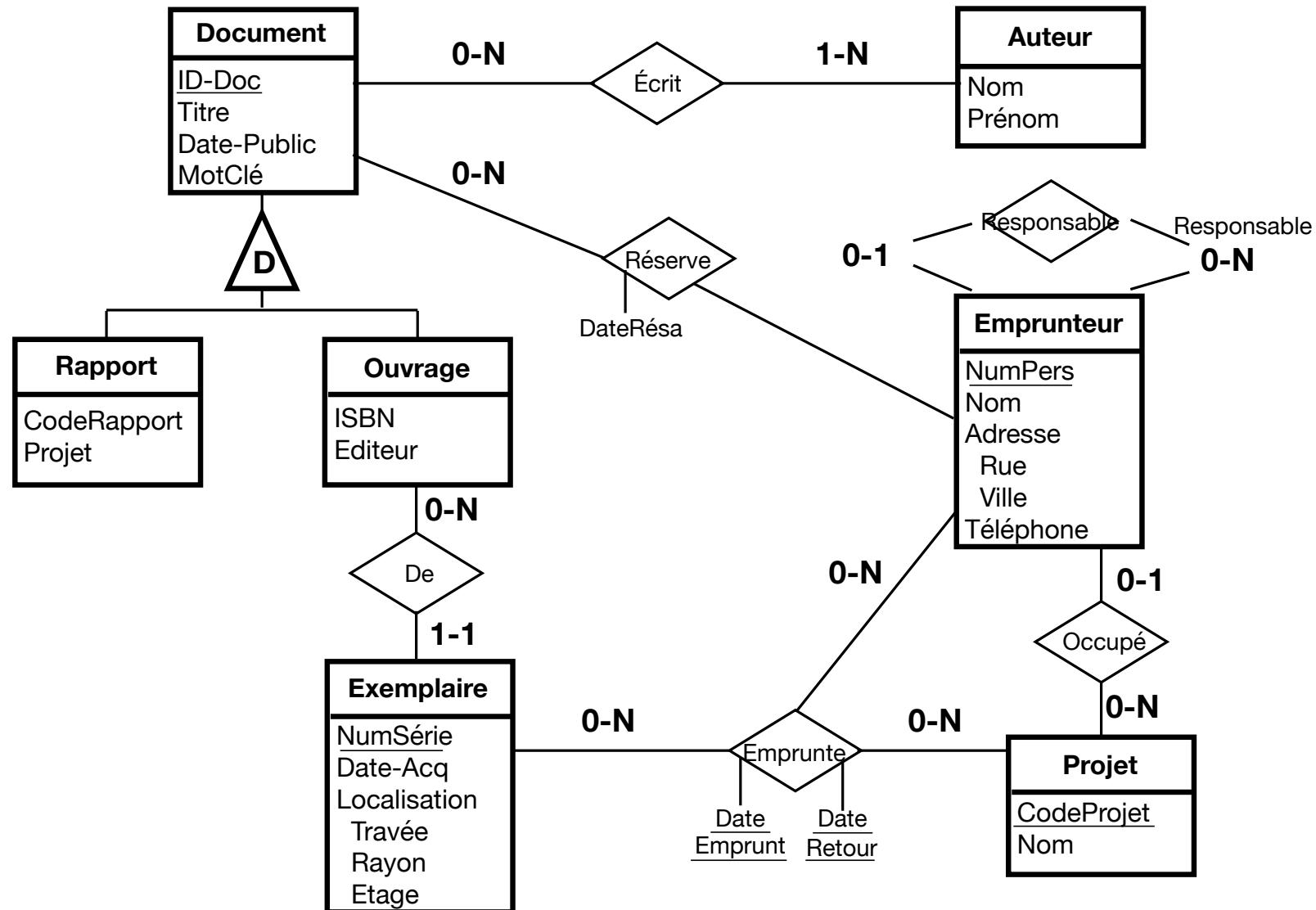


Extensions du modèle entité/associations

Jusqu'à présent, le formalisme présenté reste simple et permet de modéliser la plupart des problèmes qui peuvent se poser en pratique pour un non professionnel.

Nous allons maintenant regarder quelques extensions du modèle entité-association en présentant quelques extensions. Pour cela, on va se baser sur le schéma plus complexe suivant

Extensions du modèle entité/associations



Extensions du modèle entité/associations

- **Relations surtype/sous-type** ou **relation IS-A** : c'est lorsque le fait qu'une entité puisse appartenir à plus d'un type. Par exemple avec les rapports et les ouvrages qui forment deux classes spéciales de documents. *Tout rapport IS A document.*
Les entités *rappor*t et *documents* héritent des caractéristiques du type *document* (attributs + identifiants qui sont ceux de documents). Cette transmission, cet héritage est automatique.
- **Identifiants** : On remarque que certains types d'entités n'ont pas d'identifiants comme c'est le cas pour le type d'entité auteur.

Extensions du modèle entité/associations

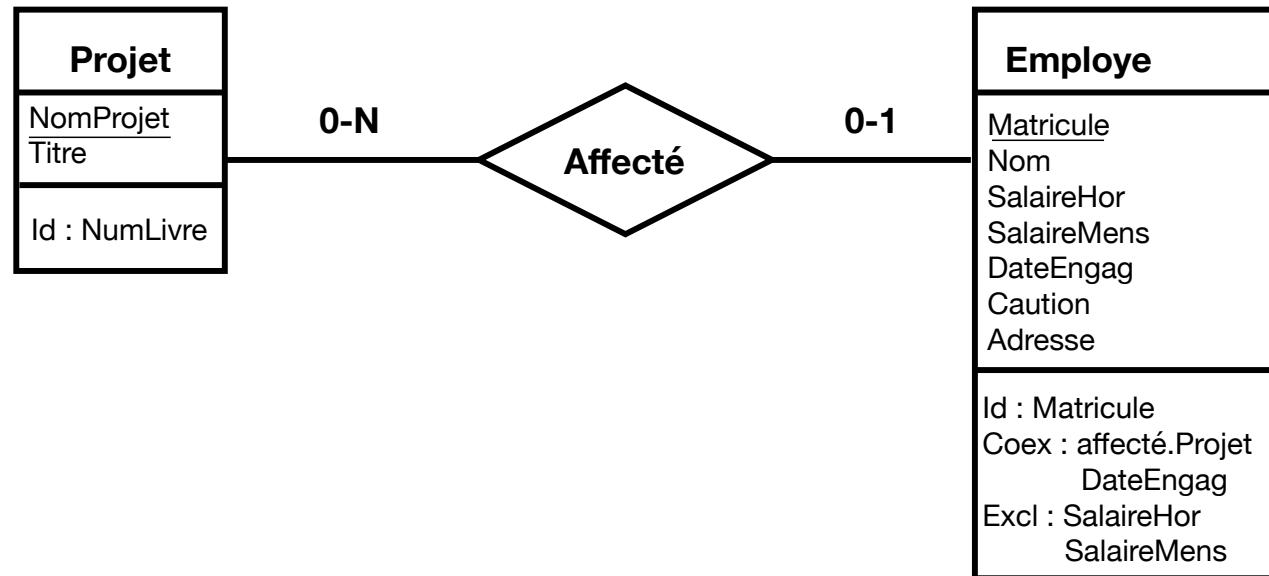
- **Associations N-aires** : ce sont des associations déjà rencontrées plus tôt et qui mettent en relations plusieurs types d'entités.
Ex : relation ternaire entre EMPRUNTEUR - PROJET - EXEMPLAIRE
- **Attributs de types d'associations** : nous avons également rencontré cela plus tôt. Il est courant d'associer des attributs à des associations afin d'en spécifier les caractéristiques.
Ex : préciser la date d'emprunt d'un exemplaire (entre EXEMPLAIRE et EMPRUNTEUR)

Extensions du modèle entité/associations

- **Attributs composés** : cela arrive lorsque notre attribut est *complexe*, ce qui est souvent le cas avec une adresse par exemple. Il est alors d'usage de le décomposer en attributs *simples/élémentaires*.
Ex : voir les types d'identités EXEMPLAIRE et EMPRUNTEUR.
- **Attributs multivalués** : En principe chaque entité possède une valeur de chacun de ses attributs. Si un attribut est multivalué (ex : liste des vaccins effectués), alors une entité peut posséder plusieurs valeurs de cet attribut.
Ex : Voir les attributs Mot Clé de DOCUMENT et Téléphone de EMPRUNTEUR.
Le nombre de valeurs par entité est caractérisé par une contre de cardinalité de type *min-max*.

Extensions du modèle entité/associations

- **Contraintes d'intégrités** : voir exemple ci-dessous.



Si un employé est affecté à un projet, alors il possède une date d'engagement et inversement (Coex)

Un employé a un salaire horaire ou un salaire mensuel, mais pas les deux (Excl)

Conclusions et ... suite !

Il existe bien d'autres formalisme conceptuel, nous avons vu ensemble le *modèle entité/associations*. Nous aurions également pu aborder le modèle *Mérise* qui lui est antérieur ou encore le modèle *UML : Universal Modelling Language* mais ce n'est pas le but de ce cours.

Regardons maintenant comment passer du **modèle conceptuel** au **modèle logique ou relationnel** !