# Introduction

Objectifs du cours

* Connaître les principales caractéristiques des bases de données
* Être capable d'effectuer une modélisation de données
* Comprendre la documentation spécifique de modélisation
* Maîtriser l'essentiel du langage SQL dit « standard »
* Savoir écrire une requête dans les règles de l’Art (standard, performances…)

## Une BDD dans un système d’information

### Définitions

* Le Système d’information : ensemble organisé de ressources (humaines, matérielles) et de procédures, permettant l'exécution des processus internes à une organisation via les manipulations de ses données.
* Le modèle relationnel : organisation consistant à stocker l'information dans des tables très précisément définies par leur schéma (leurs différentes colonnes, clés primaires, clés étrangères). Cela permet de ne pas stocker l'information plusieurs fois, et de pouvoir facilement consolider les données avec des requêtes SQL et des jointures.
* Base de données : Ensemble de données mémorisées sur des supports accessibles par un ordinateur pour satisfaire simultanément plusieurs utilisateurs de façon sélective et en temps très court. Elles constituent le cœur du système d’information.

La Base de données relationnelle correspond à la structure d’organisation des données correspondant au modèle relationnel.

* Donnée : un fait brut, impossible à interpréter tel quel. Exemple : un stock à 0
* Information : une donnée prenant du sens dans un contexte d'interprétation. Exemple : manque de carottes en stock
* Connaissance : une information analysée dans un contexte d'action. Exemple : un stock à reconstituer par l'achat de lots de carottes

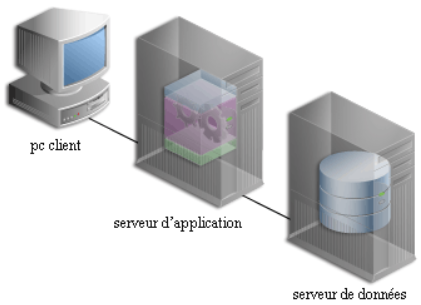
### Une base de données dans son environnement

Les bases de données sont fortement dépendantes du paysage informatique dans lequel elles baignent. Leurs objectifs divers et variés ont des impacts sur l’architecture technique et applicative, ainsi que sur les principes de leur modélisation.

Ces différents contextes amènent à penser la conception des bases de données de manière non dogmatique, en répondant au cas par cas (objectifs, contraintes, évolutivité, etc.).

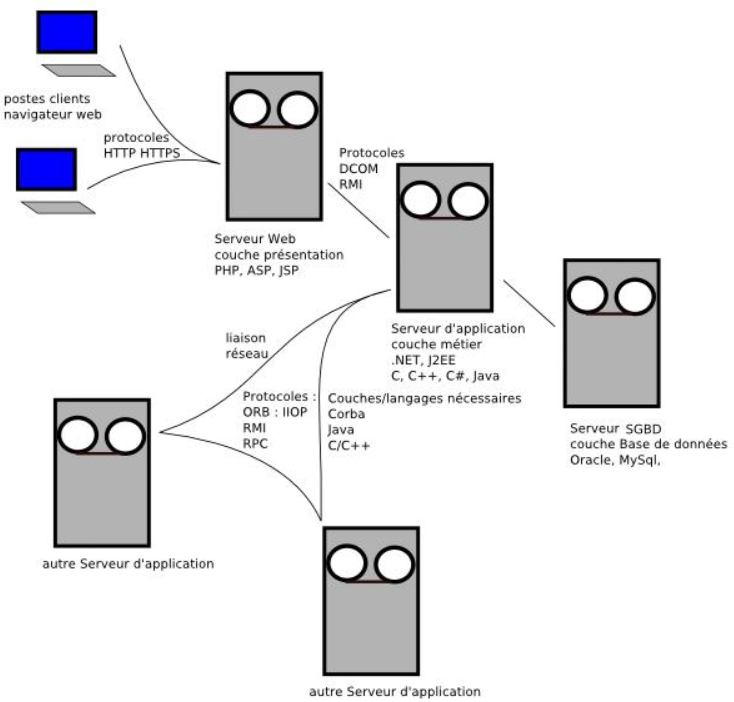
#### Architecture 3 tiers

Application du principe client/server dominant depuis les années 80. Le modèle relationnel est fortement adapté à cette structure.



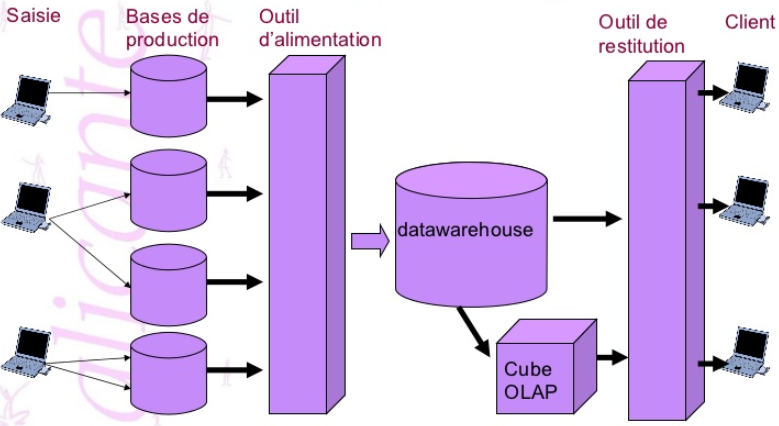
#### Architecture n-tiers

La performance des réseaux ont tendu à mutualiser les ressources, et donc à interconnecter les services applicatifs et informationnels. L’architecture n-tiers peut être vue comme une extension de l’architecture 3-tiers multi dimensionnelle.



#### Business intelligence

Le schéma relationnel est adapté dans un contexte transactionnel mais devient limité pour de l’analyse à grands volumes notamment pour l’aide à la prise de décision prospective à l’aide d’agrégation multiples et de visions selon les besoins. Cette nouvelle approche correspond à une forme de dénormalisation du modèle relationnel.



## Les normes SQL

Les bases de données de type relationnelles sont apparues dans les années 1980. Elles s’appuient sur les travaux développés par un chercheur, Mr Edgard Codd, travaillant chez IBM sur le modèle relationnel au début des années 1970. Les données sont organisées en tables distinctes sans niveau de hiérarchie. Il n’y a plus de pointeurs mais les données contenues dans les tables permettent de réaliser des liens entre les tables.

La tendance actuelle est aux systèmes qui utilisent plusieurs types de bases de données, comme SQL Server 2016. Il est ainsi possible d’écrire des requêtes qui portent sur des bases relationnelles et NoSQL.

Le langage SQL (Structured Query Language) signifie langage d’interrogation structuré. Il a été créé au début des années 1970 par IBM. C’est une start-up nommée Relational Software qui produira la première version commercialisable en 1979. Cette start-up est depuis devenue Oracle Corp.

Le langage SQL se décompose en plusieurs sous-ensembles :

* Le **DDL** pour Data Definition Language, qui regroupe les ordres utilisés pour créer, modifier ou supprimer les structures de la base (tables, index, vues, etc.). Il s’agit principalement des ordres CREATE, ALTER et DROP.
* Le **DML** pour Data Manipulation Language, qui regroupe les ordres utilisés pour manipuler les données contenues dans la base. Il s’agit principalement des ordres SELECT, INSERT, DELETE et UPDATE.
* Le **DCL** pour Data Control Language, qui regroupe les ordres utilisés pour gérer la sécurité des accès aux données. Il s’agit principalement des ordres GRANT et REVOKE.
* Le **TCL** pour Transaction Control Language, qui regroupe les ordres utilisés pour gérer la validation ou non des mises à jour effectuées sur la base. Il s’agit principalement des ordres COMMIT et ROLLBACK.

La première version de SQL normalisée par l’ANSI date de 1986.

Elle sera suivie de plusieurs versions plus ou moins importantes.

La norme SQL2 ou SQL92 est la plus importante. La majorité des SGBDR existants implémentent cette version.

Puis suivront plusieurs autres évolutions SQL-3, SQL:2003, SQL:2008 et SQL:2011 qui apportent chacune quelques fonctions complémentaires.

Une image contenant texte

Description générée automatiquement

Depuis quelques années, d’autres types de bases de données sont apparus avec les sites Internet utilisés par des millions d’utilisateurs :

* Base de données qui range les données en colonnes et non en lignes pour gérer des volumes de données énormes comme Bigtable, développée par Google, ou Cassandra, utilisée par Facebook.
* Base de données NoSQL, comme Redis, plus souple avec un schéma défini de façon imprécise, qui se préoccupe de la cohérence finale.
* Base de données de documents, similaire au type de base NoSQL. La plus connue et utilisée est MongoDB.

Chaque fournisseur de SGBDR a implémenté à sa façon le langage SQL et a ajouté ses propres extensions. Attention à la compatibilité !

Le site db-engines.com actualise régulièrement les informations sur la popularité des bases de données, globalement et par catégorie. Il explique sa méthodologie de classement.

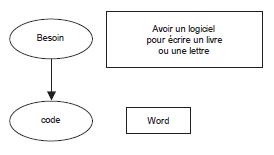
Une image contenant texte

Description générée automatiquement

# Modélisation des données

## Cycle de vie d’un projet

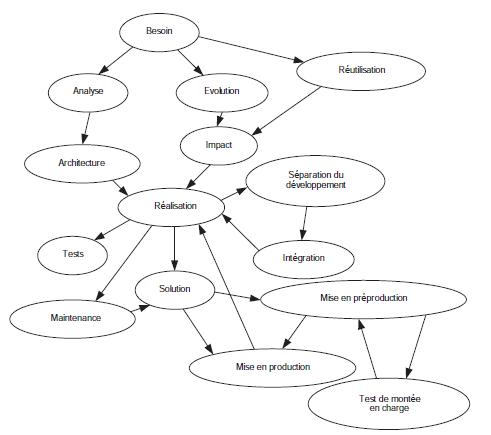
### Le début de la réflexion : Besoin -> réalisation…



### …et les problèmes surviennent

* Comment identifier correctement le besoin -> Analyse du besoin
* Le besoin s’inscrit-il dans la durée, est-il évolutif, donc impact sur le code -> Réutilisation et évolutivité du code
* Paysage applicatif et architectural de l’application -> impacts en entrée et sortie de l’application
* Le code est-il correct dans tous les cas de figure -> Tests du code (unitaires, intégration, qualité, performance)
* Les utilisateurs vont-ils modifier leurs processus avec l’application -> suivi des besoins, intégrer les évolutions et améliorations
* Comment va être géré le workflow des remontées utilisateurs (bugs) -> Processus de maintenance
* Formation utilisateur -> Documentation à différents niveaux
* Comment les utilisateurs vont-ils pouvoir piloter l’application -> Documentation d’installation de l’application et des processus courants (paramétrages)
* Quelle est la politique de mise en production et de correctifs -> directe, par pallier, patchs, montée de versions…

Ce qui nous donne au final, une fois considéré toutes ces activités lors de la durée de vie du projet, plutôt ce type de schéma :



### Solution projet : Le cycle de l’abstraction

* Etudier et comprendre l’Univers réel
* Le modéliser pour le représenter simplement
* Concevoir la solution informatique
* Implémenter la solution

En méthodologie classique, ce cycle de l’abstraction est représenté par 3 étapes majeures :

|  |  |  |
| --- | --- | --- |
| Niveaux | Données | Rôle |
| Conceptuel | Modèle Conceptuel des Données | Analyse/Conception |
| Logique | Modèle Logique des Données | Traduction en logique informatique |
| Physique | Modèle Physique des Données | Réalisation |

## Le modèle conceptuel des données = MCD

Le MCD introduit la notion d’entités, de relations et de propriétés. Il décrit de façon formelle les données utilisées par le système d’information. La représentation graphique, simple et accessible, permet à un non informaticien de participer à son élaboration. Les éléments de base constituant un modèle des données sont :

* Les propriétés
* Les entités (ou objets)
* Les relations

### Les propriétés

Les propriétés sont les informations de base du système d’information.

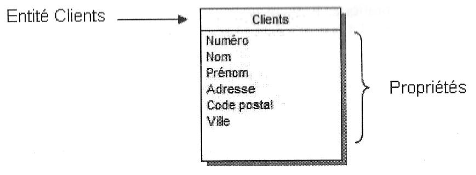
Un client possède un numéro de client, un nom, un prénom, habite à une adresse précise, etc. Ces informations élémentaires essentielles sont des propriétés.

Les propriétés disposent d’un type. Elles peuvent être numériques, représenter une date, leur longueur peut être aussi définie. Par exemple : le nom est une propriété de type alphabétique et de longueur 50, c'est-à-dire que la valeur saisie ne comportera aucun chiffre et ne dépassera pas 50 caractères. Les propriétés doivent être atomiques, relativement au système considéré.

Les types ne sont pas décrits au niveau conceptuel, car ce niveau est trop proche de la définition du système physique.

### Les entités (ou objets)

L’entité est l’objet de gestion d’un nombre de propriétés identifiées, portées par celle-ci. Comme il est aisé de le constater, les clients sont définis par certaines propriétés (numéro, nom, prénom…). Le fait de les regrouper amène naturellement à créer une entité Clients. Le symbolisme retenu est le suivant :



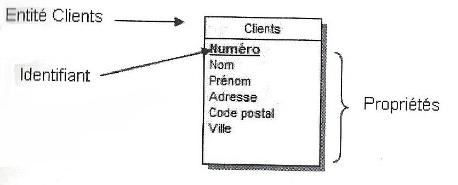
L’une des propriétés a un rôle précis, c’est l’identifiant nommé aussi la clé.

L’identifiant permet de connaitre de façon certaine et unique l’ensemble des propriétés qui participent à l’entité. Par exemple, le fait de connaitre la ville d’un client permet-il de connaitre on nom ? La connaissance du nom permet-il de connaitre la ville ? Non.

Il faut donc trouver, ou inventer, une propriété qui lorsque sa valeur est connue permet la reconnaissance de l’ensemble des valeurs qui s’y rattachent de façon formelle.

Ainsi, lorsque le numéro du client est connu, son nom, son prénom et toutes les valeurs des autres propriétés qui s’y rattachent sont connues de façon sûre et unique.

Au niveau du formalisme, cette propriété se souligne. Voici le schéma modifié de l’entité Clients :



### Les relations

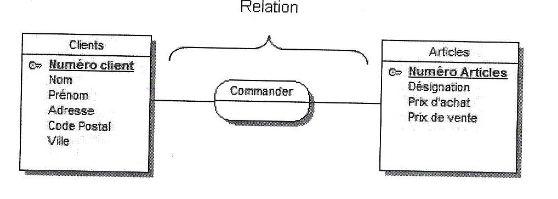
#### Définition

Les entités regroupent un ensemble d’informations élémentaires. Les entités sont la plupart du temps liées entre-elles.

Par exemple :

Un client peut commander des articles.

Si nous analysons cette phrase, on distingue 2 entités Clients et Articles) et un verbe (commander) qui indique une action les reliant. Formalisons cette phrase avec Merise.



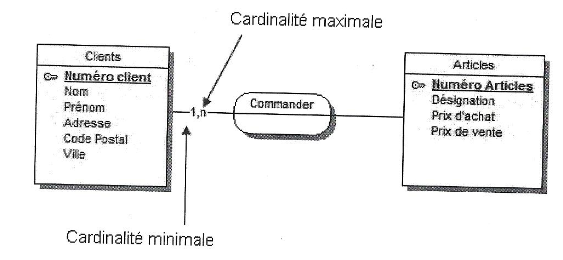
#### Les cardinalités

En reprenant la même phrase, on peut y distinguer une information supplémentaire : la cardinalité. **Un** client peut commander **des** articles.

Les cardinalités expriment le nombre de fois où l’occurrence d’une entité participe aux occurrences de la relation.

* Combien de fois au minimum un client peut-il commander un article ?
* Combien de fois au maximum un client peut-il commander un article ?

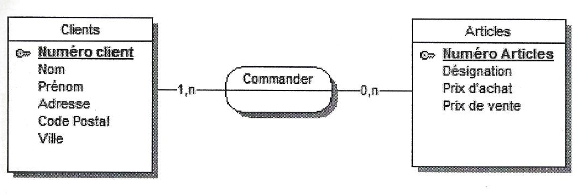
Voilà comment formaliser la réponse à ces 2 questions, selon le contexte :



De même, en considérant l’article :

* Combien de fois au minimum un article peut-il être commandé par un client ?
* Combien de fois au maximum un article peut-il être commandé par un client ?

La réponse à ces 2 questions selon le contexte :



La cardinalité minimale (0 ou 1) exprime le nombre de fois minimum qu’une occurrence d’une entité participe aux occurrences d’une relation.

La cardinalité maximale (1 ou n) exprime le nombre de fois maximal qu’une occurrence d’une entité participe aux occurrences de la relation. Si le maximum est connu, la valeur doit être inscrite au niveau du MCD.

A ce stade, il est encore souvent question d’interprétation. Ces interprétations traduisent la plupart du temps des choix fonctionnels structurants que l’équipe de développement pourra ainsi anticiper.

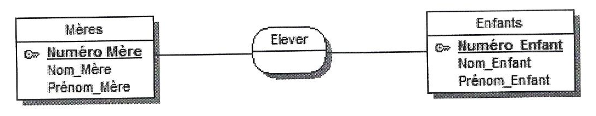
Exercice :

Une mère élève des enfants

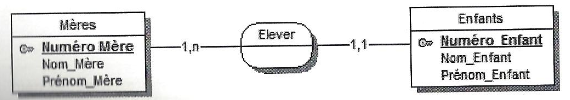
* Les 2 entités



* La relation



* Les cardinalités

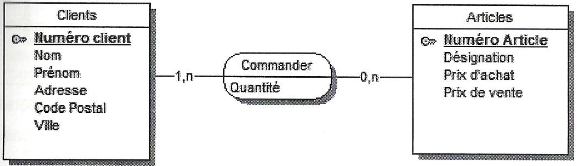


#### Les relations porteuses de données

Une relation est dite porteuse lorsqu’elle contient elle-même des propriétés.

Imaginons que l’on veuille connaitre la quantité d’articles commandés par client, la quantité est une propriété élémentaire qui dépend de la relation entre les 2 entités client et article.

Voici le MCD correspondant :

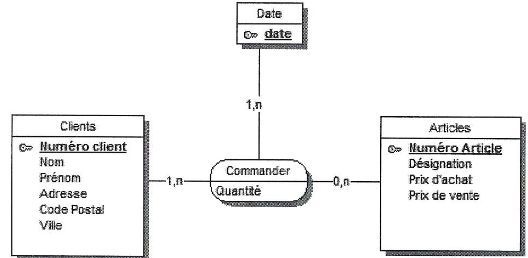


Nous pouvons donc lire sur ce schéma : le client X a commandé (action) la quantité Y d’articles Z.

Une relation faisant intervenir 2 entités est dite binaire.

Question : Où serait la propriété date d’achat ?

Dans le cas où la date serait portée par une autre entité (laquelle par exemple ?), une relation ternaire peut être la solution.

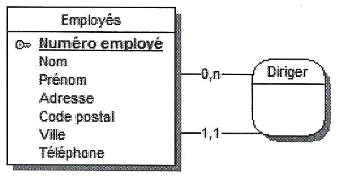


#### Les relations réflexives

Une relation réflexive est une relation d’une entité sur elle-même. L’exemple type est la hiérarchie des occurrences d’une entité sur d’autres occurrences de la même entité.

Exemple :

Un employé dirige plusieurs employés.



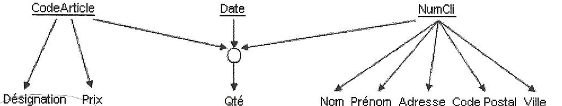
Un employé peut donc diriger zéro ou plusieurs personnes, et un employé est dirigé par un et un seul autre employé.

### Règles d’usages

* Toute entité doit comporter un identifiant
* Toutes les propriétés de l’entité dépendent fonctionnellement de l’identifiant. Ainsi, connaissant la valeur de l’identifiant, nous connaissons de façon sûre et unique la valeur des propriétés associées.
* Le nom d’une propriété ne doit apparaitre qu’une seule fois dans le MCD.
* Les propriétés résultantes d’un calcul ne doivent pas apparaître dans le MCD.

**Cas pratique**

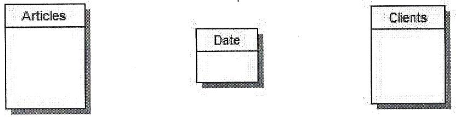
Reprenons le graphe concernant le cas du camping et réalisons le MCD le concernant.



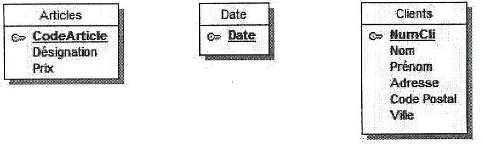
Le MCD pas à pas

Déterminons les entités. Par rapport au graphe, nous pouvons remarquer trois sources de dépendances fonctionnelles : CodeArticle, Date et Numcli.

* Chacune de ces sources peut représenter une entité :

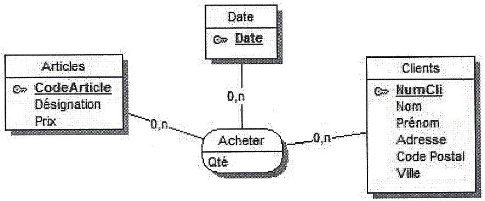


* On ajoute leurs propriétés respectives :

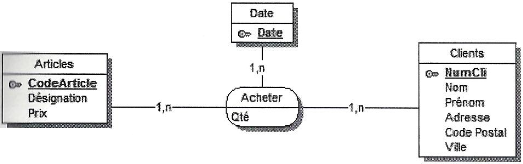


* Les relations

Dans un sens assez abstrait et très large, voici ce que cela pourrait donner :



Dans la réalité, on peut imaginer plutôt ce type de relation :



A priori, un article est acheté dans une certaine quantité, par au minimum un client à une date donnée. Toujours à voir selon le contexte…

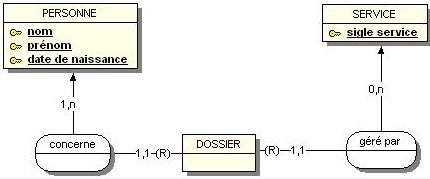
### Notions d’entité forte, d’entité faible et d’identifiant relatif

* Entité forte

Une entité forte est une entité qui, disposant de son identifiant, peut être considérée de façon isolée.

* Entité faible

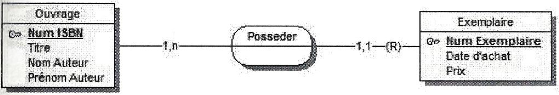
Une entité faible est une entité qui ne peut être considérée qu’en relation avec une autre entité.



* L’identifiant relatif

L’entité Dossier est une entité faible car il n’a d’existence que par les 2 entités fortes que sont le service et une personne. Au niveau du MCD, son identifiant est donc hérité des identifiants des 2 entités personnes et Service.

Exemple :



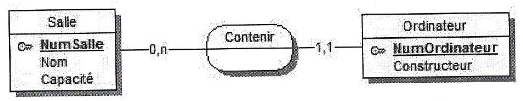
La clé d’identification de l’exemplaire est la concaténation du Num ISBN et du NumExemplaire. L’identification d’un identifiant relatif se traduit par le symbole (R) au niveau de la cardinalité.

Cela traduit une relation de hiérarchie entre les 2 entités. Si je supprime un ouvrage, je supprime alors tous ses exemplaires.

### Notion de contrainte d’intégrité fonctionnelle

Une contrainte d’intégrité fonctionnelle est définie par le fait qu’une des entités de l’association est déterminée par la connaissance d’une ou plusieurs entités participant à cette même association.

Par exemple :



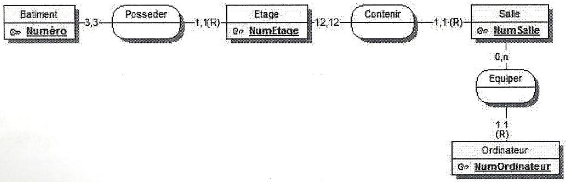
A travers le jeu des cardinalités, on peut en conclure une contrainte d’intégrité fonctionnelle :

On peut lire qu’une salle peut contenir 0 ou plusieurs ordinateurs et qu’un ordinateur existe dans une et une seule salle. Dans le cas d’une association binaire comme celle-ci, une contrainte d’intégrité fonctionnelle existe à partir du moment où une cardinalité de type **1,1** est présente.

Autre exemple :

Un ordinateur fait partie d’une salle, qui est dans un étage d’un bâtiment.

Voici le MCD correspondant :

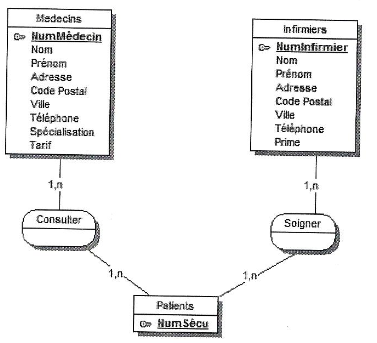
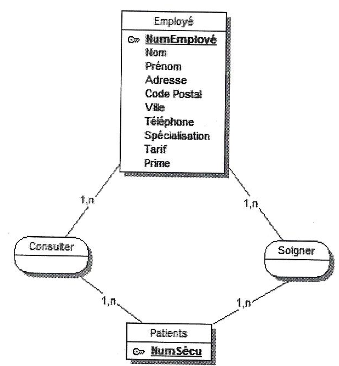


### L’héritage (ou la généralisation – spécialisation)

Commençons par un exemple.

* Une maison de santé reçoit des patients, et deux types de personnel sont salariés : les médecins qui réalisent les consultations et les infirmiers qui administrent les soins
* Les médecins ont une spécialisation (médecin du sport, gérontologie, etc.) et un tarif à l’acte.
* Les infirmiers ont des primes d’astreinte.
* Les patients sont juste référencés par leur numéro de sécurité sociale.

On peut imaginer 2 solutions de MCD a priori :

Ces 2 solutions correctes dans les faits, présentent tout de même des inconvénients structurels.

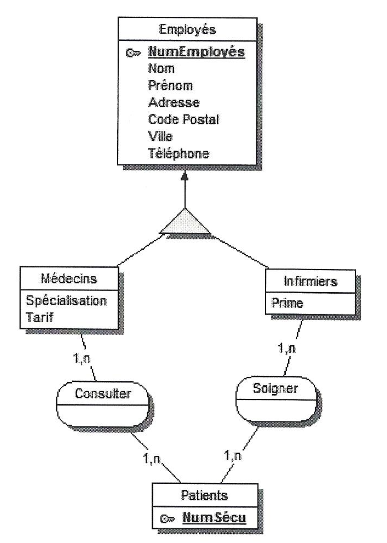
Le 1er MCD oblige à dupliquer des attributs entre les 2 entités médecins et infirmiers (nom, prénom, etc.), alors que dans le 2ème MCD certaines rubriques sont vides (spécialisation et tarifs pour les infirmiers, prime pour les médecins).

Afin de résoudre ces problèmes, nous allons utiliser le concept de l’héritage.

Le principe global de l’héritage est de factoriser les propriétés identiques dans une entité commune. Cette entité est aussi nommée entité générique ou sur-type d’entité. Les propriétés spécifiques seront contenues dans une entité spécialisée nommée aussi sous-type d’entité.

Ainsi, chaque sous-type hérite des propriétés et des associations du sur-type. La relation qui fait correspondre un sous-type à son sur-type est une généralisation. La relation inverse est une spécialisation.

Voici une formalisation de l’héritage pour l’exemple :

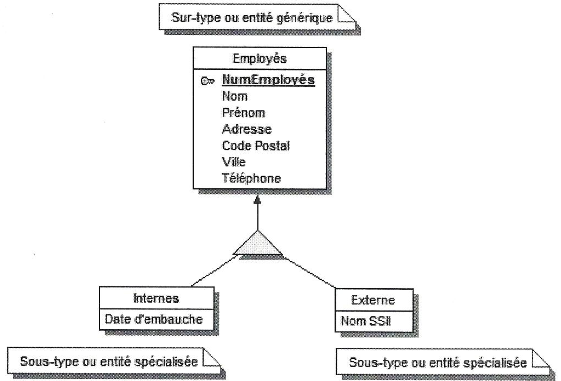


Désormais, tous les attributs communs entre médecins et infirmiers sont les propriétés de l’entité générique employés. Médecins et infirmiers sont des entités spécialisées d’employés. Chacune aura comme propriétés ses propres attributs.

L’identifiant de médecins et d’infirmiers est celui d’employés : les entités spécialisées héritant des propriétés de l’entité générique, on ne fait pas apparaître l’identifiant des entités spécialisées.

Autre exemple :

Une multinationale gère ses salariés ainsi que des salariés de SSII. Voici le MCD qui en découle :



#### Les contraintes ensemblistes

Les extensions du modèle entité-association permettent de représenter des contraintes sur des ensembles d’occurrences d’entités ou d’associations. L’entité concernée par la contrainte est appelée le pivot de la contrainte.

L’ensemble de ces contraintes est formé à partir de 2 contraintes de base :

* La contrainte de couverture
* La contrainte de disjonction

##### La contrainte de couverture

Toute occurrence de l’entité générique appartient au moins à l’un des sous-types.

##### La contrainte de disjonction

Toute occurrence de l’entité générique doit appartenir à un seul sous-type : les sous-types sont donc mutuellement exclusifs.

##### Les différentes interactions entre les contraintes

Ces 2 contraintes peuvent être présentes ou pas, selon les règles de gestion. Ainsi, nous avons 4 possibilités d’interaction entre ces 2 contraintes

###### La contrainte d’exclusion (non couverture et disjonction)

Une occurrence du sur-type appartient éventuellement à un et un seul sous-type. Il y’a donc disjonction, mais pas de couverture (formalisé par X). C’est l’un ou l’autre, mais pas les 2, au aucun.

###### La contrainte de totalité (couverture et non-disjonction)

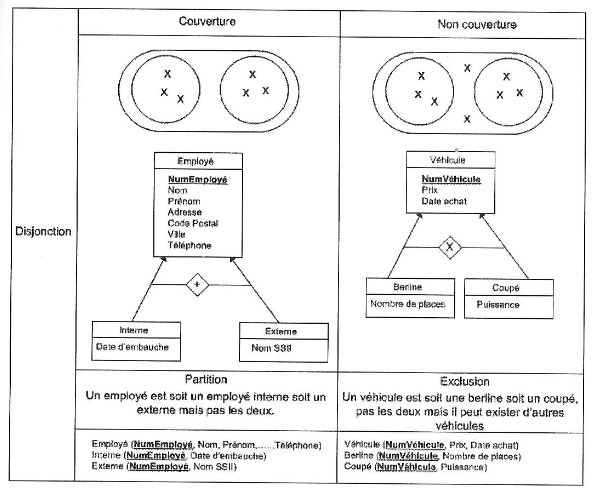
Une occurrence du sur-type appartient toujours à un sous-type, éventuellement à plusieurs. Il y’a donc couverture mais pas de disjonction (formalisé par T).

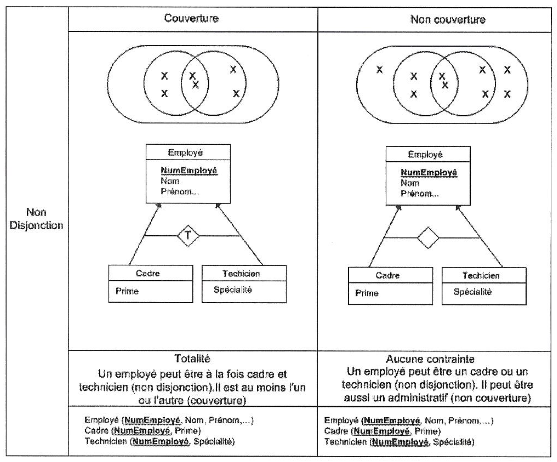
###### La contrainte de partition (couverture et disjonction)

Une occurrence du sur-type appartient toujours à un sous-type et à un seul. Il y’a donc couverture et partition (formalisé par XT ou +).

###### Pas de contrainte (pas de couverture ni de disjonction)

Une occurrence du sur-type n’appartient à aucun sous-type, à un sous-type ou à plusieurs (aucune formalisation).





#### La formalisation des contraintes entre associations

Nous retrouvons les 4 contraintes décrites pour les sous-types :

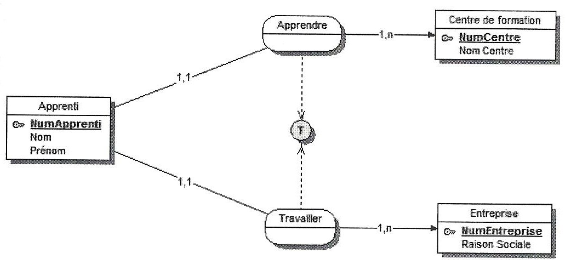
* La totalité = couverture + non-disjonction
* La partition = couverture + disjonction
* L’exclusion = non-couverture + disjonction
* Aucune contrainte = non-couverture + non-disjonction

A ces contraintes s’ajoutent 2 autres contraintes spécifiques aux associations.

* Contrainte d’égalité ou de simultanéité : toute occurrence qui participe à l’association A participe également à l’association B (formalisé par = ou S)
* Contrainte d’inclusion : toutes les occurrences d’une association A sont également occurrences d’une association B (formalisé par I).

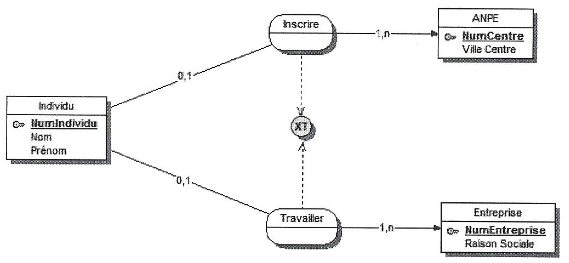
##### La contrainte de totalité (couverture + non-disjonction)

Exemple : un apprenti travaille dans une entreprise A et se forme dans un centre de formation



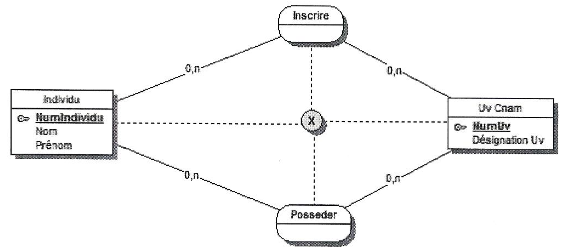
L’apprenti participe à au moins l’une des 2 associations, car il y’a couverture et non-disjonction. En fait, soit il travaille dans une entreprise, soit il apprend dans un centre de formation, soit il travaille et apprend.

##### La contrainte de partition (couverture + disjonction)



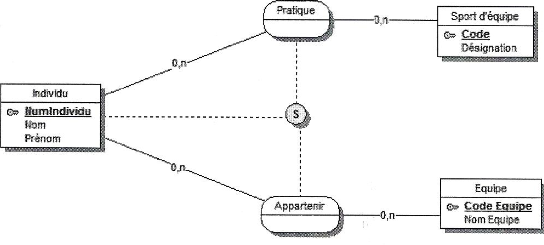
Soit une personne travaille dans une entreprise, soit elle est au chômage.

##### La contrainte d’exclusion (non-couverture + disjonction)



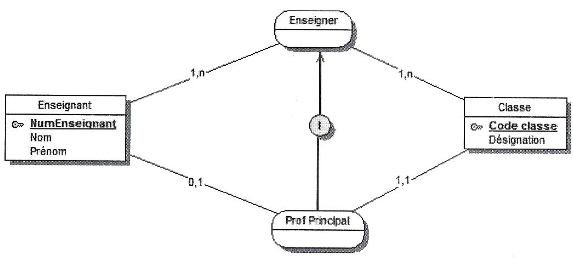
Un individu ne peut, à la fois être inscrit à une UV, et la posséder.

##### La contrainte d’égalité (=simultanéité)



Toute personne qui pratique un sport appartient à une équipe et vice versa. L’ensemble des occurrences du pivot est donc la même pour chacune des associations participant à la contrainte.

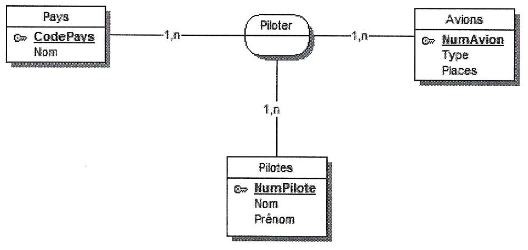
##### La contrainte d’inclusion



Toutes les occurrences d’une association sont également occurrences d’une autre. Ainsi, un enseignant ne peut être professeur principal que d’une classe où il enseigne.

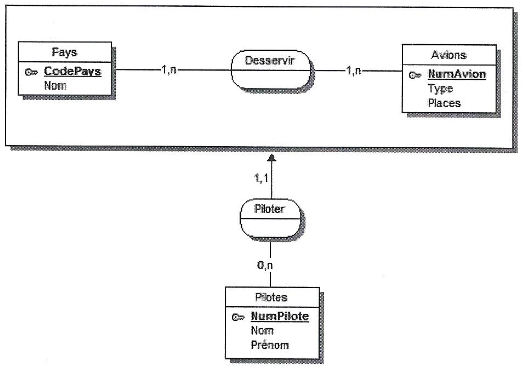
#### Agrégation sur les associations (ou associations d’associations)

Prenons le cas suivant : Dans la compagnie d’aviation « vol’toujours » , les pilotes font voler des avions dans différents pays. Mais un avion pour un pays donné n’est piloté que par un seul pilote.



Avec ce MCD, on voit que les pilotes peuvent faire voler des avions dans différents pays. Mais la règle de gestion n’est pas intégrée et difficilement intégrable sans l’utilisation de l’agrégation.

Voici comment l’intégrer :



L’agrégation consiste à percevoir comme un objet unique l’association de plusieurs autres objets. Ici ce qui est encadré représente un ensemble cohérent. D’autres représentations peuvent être proposées, selon les habitudes de formalisation.

### Cas pratique

#### Le contexte

*Votre oncle, un restaurateur, vous demande de lui réaliser un logiciel de gestion des commandes de repas. Voici les indications qu’il vous donne :*

* *Il souhaite gérer certaines informations concernant ses employés (nom, prénom, adresse complète, téléphone et diplômes).*
* *Au niveau prise d’une commande, il souhaite savoir si elle porte sur le service de midi ou celui du soir et à quelle date elle est passée.*
* *A des fins d’analyse, il souhaite savoir quelle table a passé la commande et quel serveur s’en est occupé.*
* *La carte du restaurant propose l’ensemble des plats d’entrées, principaux et desserts. Les menus proposés sont un assemblage des plats à la carte.*
* *La carte des vins propose tous les vins stockés dans la cave du restaurant. On désire connaître pour chaque bouteille son millésime, sa* date d’achat, son prix d’achat et son prix de vente. Il voudrait saisir aussi pour chaque cru les informations concernant le viticulteur (nom, prénom, adresse complète, téléphone). A l’heure actuelle votre oncle, amoureux de vin, met sur chaque goulot de chaque bouteille une étiquette contenant le prix d’achat ainsi que la date d’achat. Votre système doit pouvoir remplacer ce traitement manuel.
* Certaines boissons comme les apéritifs, les digestifs, les sodas ou les cafés sont gérés de façon simpliste juste par leur libellé et prix de vente.
* Chaque serveur prenant une commande saisit l’ensemble des informations sur une tablette qui transmet la commande via wifi sur un ordinateur central.

Réalisez le MCD en vous aidant du dictionnaire des données relevé par votre collègue fonctionnel.

#### Le dictionnaire des données

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nom | Format | Longueur | Type | |
| E | C |
| Numéro employé | Numérique |  | X |  |
| Nom employé | Alphabétique | 30 | X |  |
| Prénom employé | Alphabétique | 30 | X |  |
| Adresse employé | Alphabétique | 40 | X |  |
| Code Postal employé | Alphanumérique | 5 | X |  |
| Ville employé | Alphabétique | 40 | X |  |
| Téléphone employé | Alphanumérique | 15 | X |  |
| Diplômes des employés | Alphabétique | 30 | X |  |
| Numéro viticulteur | Numérique |  | X |  |
| Nom viticulteur | Alphabétique | 30 | X |  |
| Prénom viticulteur | Alphabétique | 30 | X |  |
| Adresse viticulteur | Alphabétique | 40 | X |  |
| Code postal viticulteur | Alphanumérique | 5 | X |  |
| Ville viticulteur | Alphabétique | 40 | X |  |
| Téléphone viticulteur | Alphanumérique | 15 | X |  |
| Numéro de la table | Numérique |  | X |  |
| Capacité | Numérique |  | X |  |
| Numéro de la boisson | Numérique |  | X |  |
| Désignation de la boisson | Alphabétique | 20 | X |  |
| Prix de vente boisson | Numérique |  | X |  |
| Numéro du vin | Numérique |  | X |  |
| Nom du vin | Alphabétique | 30 | X |  |
| Millésime du vin | Numérique |  | X |  |
| Prix de vente vin | Numérique |  | X |  |
| Numéro bouteille | Numérique |  | X |  |
| Date achat bouteille | Date |  | X |  |
| Prix achat bouteille | Numérique |  | X |  |
| Numéro menu | Numérique |  | X |  |
| Libellé menu | Alphabétique | 30 | X |  |
| Prix de vente menu | Numérique |  | X |  |
| Numéro plat | Numérique |  | X |  |
| Libellé plat | Alphabétique | 30 | X |  |
| Prix de vente plat | Numérique |  | X |  |
| Numéro de type plat | Numérique |  | X |  |
| Désignation type plat (entrée, principal, dessert) | Alphabétique | 20 | X |  |
| Type de service (midi ou soir) | Alphabétique | 4 | X |  |
| Date prise de commande | Date |  | X |  |
| Numéro commande | Numérique |  | X |  |