

# Les principaux domaines de l'informatique



... abordés dans le cadre de ce cours:



La Programmation



Les Systèmes d'Exploitation



Les Systèmes d'Information



La Conception d'Interfaces



Le Calcul Scientifique



Les Agents Logiciels



## Systemes d'information: SGBD (1)

Lorsque les données à gérer informatiquement par une entité (entreprise, université, association, ...) sont de nature diverses et possèdent de nombreux liens entre elles, les fonctionnalités fournies par un système de fichiers ne sont plus suffisantes; il convient de faire alors appel à des fonctions de gestion d'information plus sophistiquées; fournies par un *système d'information*.

L'exemple le plus courant de système d'information sont les *Systemes de Gestion des Bases de Données* (SGBD).



Un SGBD permet de gérer l'ensemble des informations nécessaires à la réalisation d'un **objectif** (tâche) **commun** au sein d'une entreprise ou de toute autre collectivité d'individus travaillant en coordination.

Exemples de tâches:

gestion des étudiants d'une université,

gestion des réservations des places d'avions,

gestion de comptes bancaires, ...



## Exemple 1:

*Catalogue de produits vendus par une entreprise:*

Base de données relativement simple

⇒ peut être gérée directement (données stockées dans un simple fichier, une feuille de tableur, ...)

## Exemple 2:

*Gestion des cours et étudiants d'une université:*

Données beaucoup plus complexes, car faisant intervenir des informations diverses, **liées entre-elles**:

- informations de type académique, sur les étudiants (matricule, date d'inscription, section, notes, ...)
- informations de type personnelles, sur les étudiants (nom, prénom, adresse, no AVS, ...)
- informations sur les cours dispensés (titre, prérequis, matière, langue, enseignant, horaire, salle, ...)
- informations sur les enseignants (nom, prénom, bureau, téléphone, statut, no AVS, ...)
- informations sur les cours dispensés (titre, matière, langue, enseignant, horaire, salle, ...)
- ...

⇒ Ensemble de données trop complexe pour être géré «manuellement»: il faut faire appel à un système d'information.



# Cycle de vie d'une base de données

Le **cycle de vie** d'une base de donnée (BD) se décompose en trois phases

- La *conception*: définition des fonctionnalités,
- L'*implantation*: réalisation effective de la base,
- L'*exploitation*: utilisation et maintenance de la base.



## Cycle de vie d'une BD: la conception

La **phase de conception** est une phase d'analyse, qui aboutit à **déterminer le futur contenu** de la BD.

L'ensemble des concepteurs et des utilisateurs potentiels doivent se mettre d'accord sur la nature et les caractéristiques des informations qui devront être manipulées.

La description obtenue (qui ne fait généralement référence à aucun système de SGBD particulier) utilise un **langage formel** basé sur des concepts bien établis, comme les *objets*, les *liens* et les *propriétés*. Cette description est appelée:

*schéma conceptuel (des besoins)*.

L'ensemble des concepts utilisés par le langage formel de description choisi est appelé le *modèle conceptuel*.

## Cycle de vie d'une BD: schéma conceptuel



Un **schéma conceptuel** se décompose généralement en deux parties:

- Une partie *statique* décrivant la structure des données;
- Une partie *dynamique* décrivant les opérations sur les données

De plus, il faudra pouvoir décrire des *contraintes* (règles) pesant sur les données (comme par exemple: «il ne doit pas y avoir plus de 20% d'écart entre les salaires des employés d'une même service et d'une même catégorie»).

Ces contraintes, appelées *contraintes d'intégrité*, seront décrites dans un langage permettant d'exprimer des règles compatibles avec le modèle conceptuel choisi.

Le modèle conceptuel illustré dans le cours est le modèle **entité-association**.



## Cycle de vie d'une BD: implantation

La **phase d'implantation** est une phase qui consiste:

- à définir le modèle choisi pour la base de données (à l'aide d'un langage symbolique de description des données –LDD– spécifique au SGBD choisi), et
- à entrer les premières données, i.e. construire une première version de la BD.



## Cycle de vie d'une BD: schéma logique

La phase d'implantation nécessite  
la **traduction du *schéma conceptuel*** dans un schéma utilisant  
les concepts du modèle employé par le SGBD (appelé *modèle logique*).

Le nouveau schéma obtenu est appelé

le **schéma logique**

(ou aussi quelquefois [malheureusement] «schéma conceptuel»).

Le modèle logique illustré dans le cours est le ***modèle relationnel***.



## Cycle de vie d'une BD: schéma interne

Pour l'implémentation effective des données,  
il faut encore effectuer les choix relatifs à leur stockage et leur  
structuration sur les mémoires secondaires, sous la forme  
d'un **ensemble de fichiers**.

Ces choix sont consignés dans ce qu'on appelle  
le *schéma interne* de la base de données, qui repose sur le *modèle interne*,  
dont les concepts sont ceux du **système de fichiers** utilisé.



## Cycle de vie d'une BD: exploitation

En phase d'exploitation, l'utilisation de la BD se fait au moyen d'un *langage de manipulation de données* –LMD– qui permet d'exprimer aussi bien des *requêtes d'interrogation* (pour obtenir des informations contenues dans la base) que des *requêtes de mise à jour* (pour modifier le contenu de la base).

Le langage de manipulation de données illustré dans le cours est:

**SQL** (*Structured Query Langage*) , version 92.



## Cycle de vie d'une BD: schémas externes

Lors de son interaction avec la BD,  
chaque utilisateur (ou groupe d'utilisateurs) n'est généralement intéressé  
que par **une partie des données stockées** dans la base.

On lui associe donc un *schéma externe* (aussi appelé *vue*)  
décrivant le sous-ensemble de la base auquel il a accès,  
structuré de façon à répondre à ses besoins spécifiques.

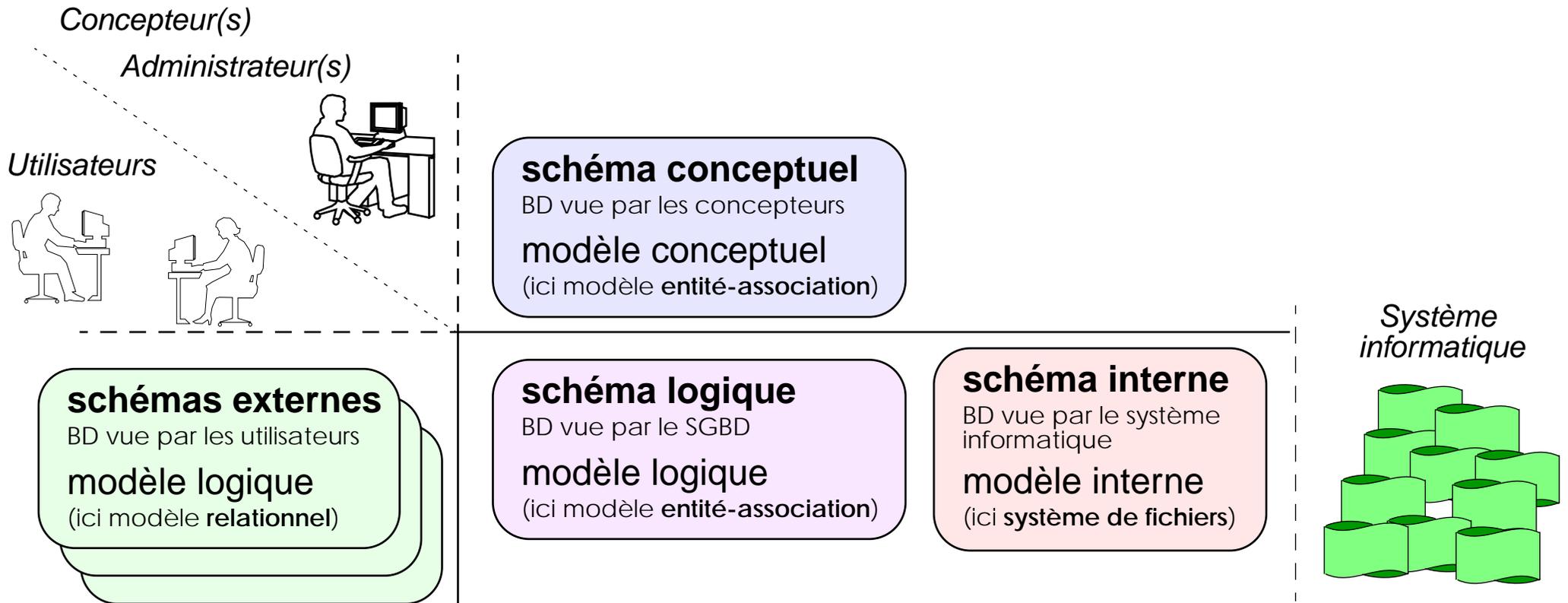
Dans les SGBD actuels, le modèle utilisé pour décrire les schémas externes  
est le même que celui du schéma logique.



# Les schémas d'une base de données

La description complète d'une base de données est donc réalisée à l'aide de 4 types de schémas, dont 3 sont directement utilisés par le SGBD.

Il sont organisés de la façon suivante:

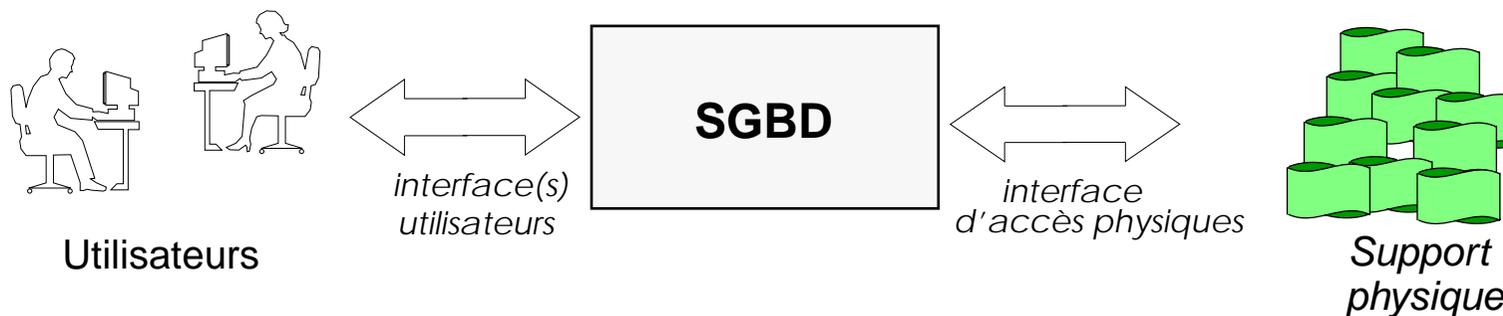




## Principe de fonctionnement d'un SGBD (1)

L'objectif fondamental de l'organisation d'un SGBD est d'assurer **l'indépendance programmes/données**:

- D'une part, un utilisateur doit pouvoir modifier sa vue de la base, sans avoir à se soucier des choix opérés au niveau interne en matière de fichiers;
- d'autre part, un administrateur de système doit avoir la possibilité de modifier ces choix sans que cela ait un impact sur les utilisateurs



De plus un SGBD étant utilisé simultanément par plusieurs utilisateurs, il doit pouvoir résoudre les problèmes internes de coordination des actions, de cohérence (intégrité) des données, et contrôler le bon déroulement continu de ses activités.

## Principe de fonctionnement d'un SGBD (2)

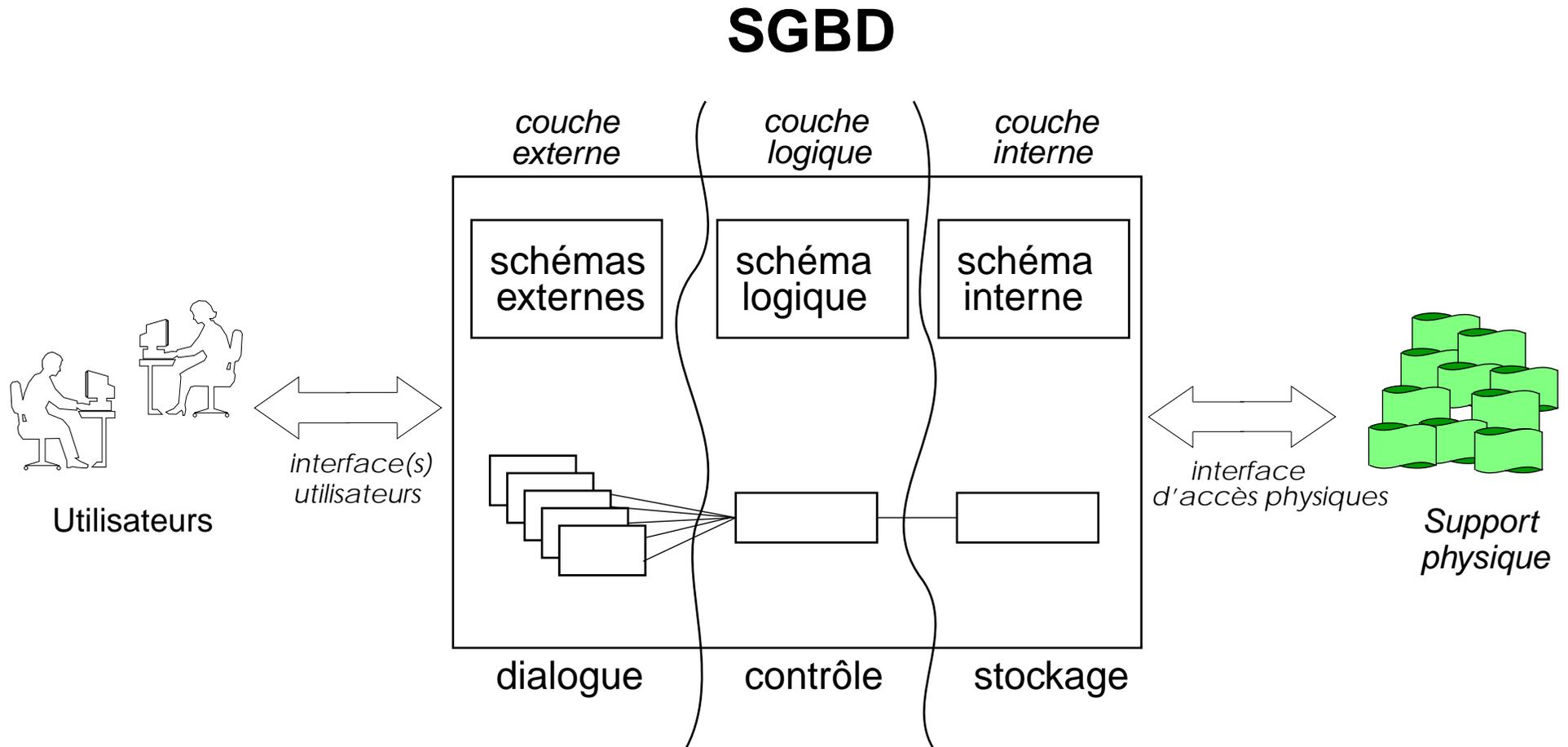


En conséquence, un SGBD est habituellement organisé en trois couches :

- **La *couche externe*:**  
qui prend en charge l'interface avec les utilisateurs  
(analyse des requêtes –interrogation, modification de la BD–,  
contrôle des droits d'accès, présentations des résultats, ...)
- **La *couche intermédiaire* (ou *logique*):**  
qui assure les fonctions de contrôle global  
(optimisation des requêtes, gestion des conflits d'accès, contrôle de  
la cohérence globale de la base, garantie du bon déroulement des  
actions en cas de panne, ...)
- **La *couche interne*:**  
qui s'occupe du stockage des données sur les supports physiques et  
de la gestion des fichiers et des accès (index, clés, ...).



# Principe de fonctionnement d'un SGBD (3)





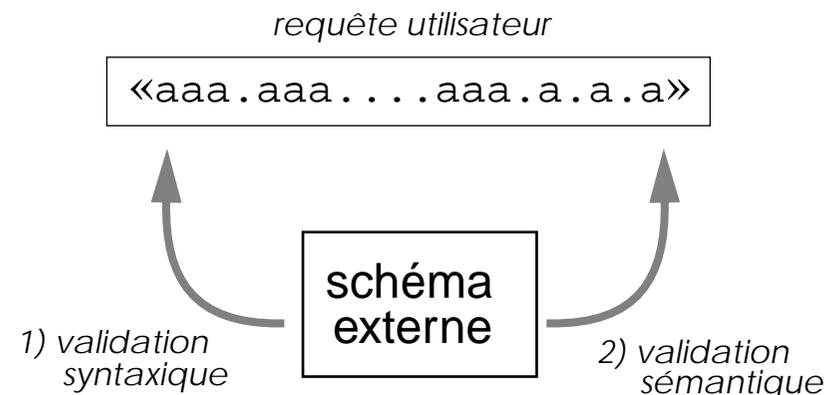
## Principe de fonctionnement d'un SGBD (4)

Avec la structuration qui vient d'être définie,  
le principe de fonctionnement d'un SGBD est le suivant:

1. une requête, exprimée par l'utilisateur dans le LMD accepté par le SGBD est d'abord

**validée** du point de vue **syntactique**  
(conformité à la grammaire du langage)

2. puis validée du point de vue **sémantique**  
(les objets cités doivent être connus dans le schéma externe de l'utilisateur).



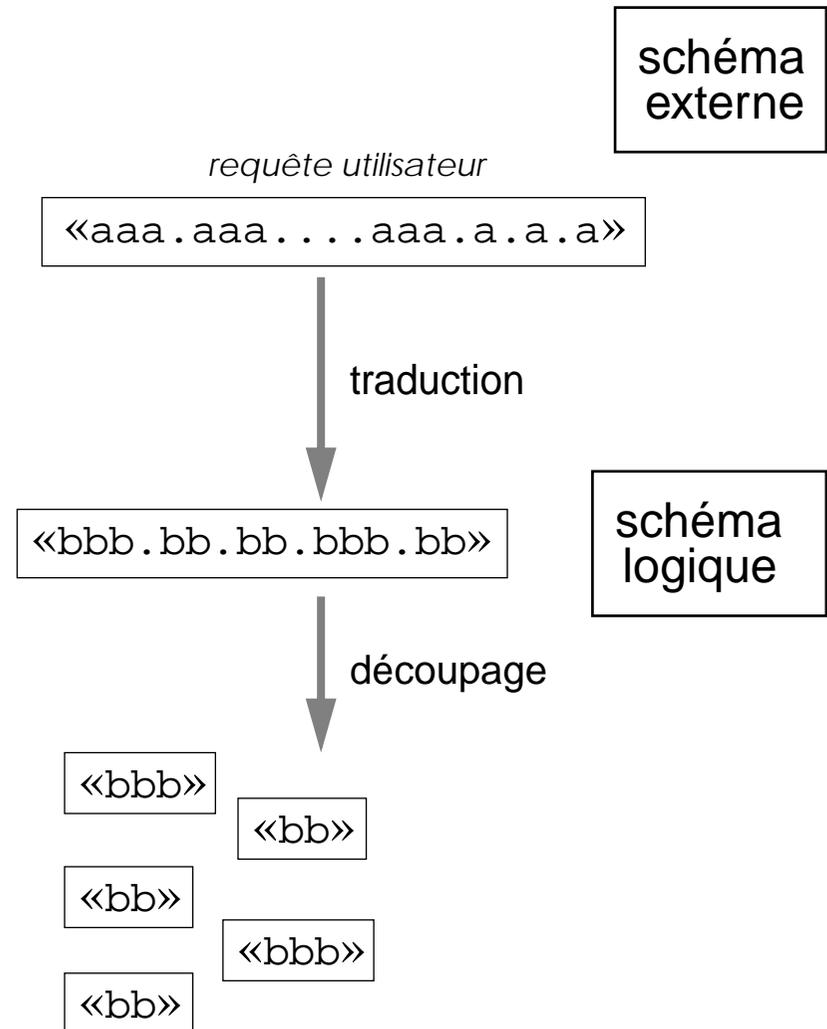


3. Après la validation faite dans la couche externe, on utilise les règles de correspondance entre schéma externe et schéma logique (établies au moment de la définition du schéma externe) pour:

**traduire la requête dans le modèle logique.**

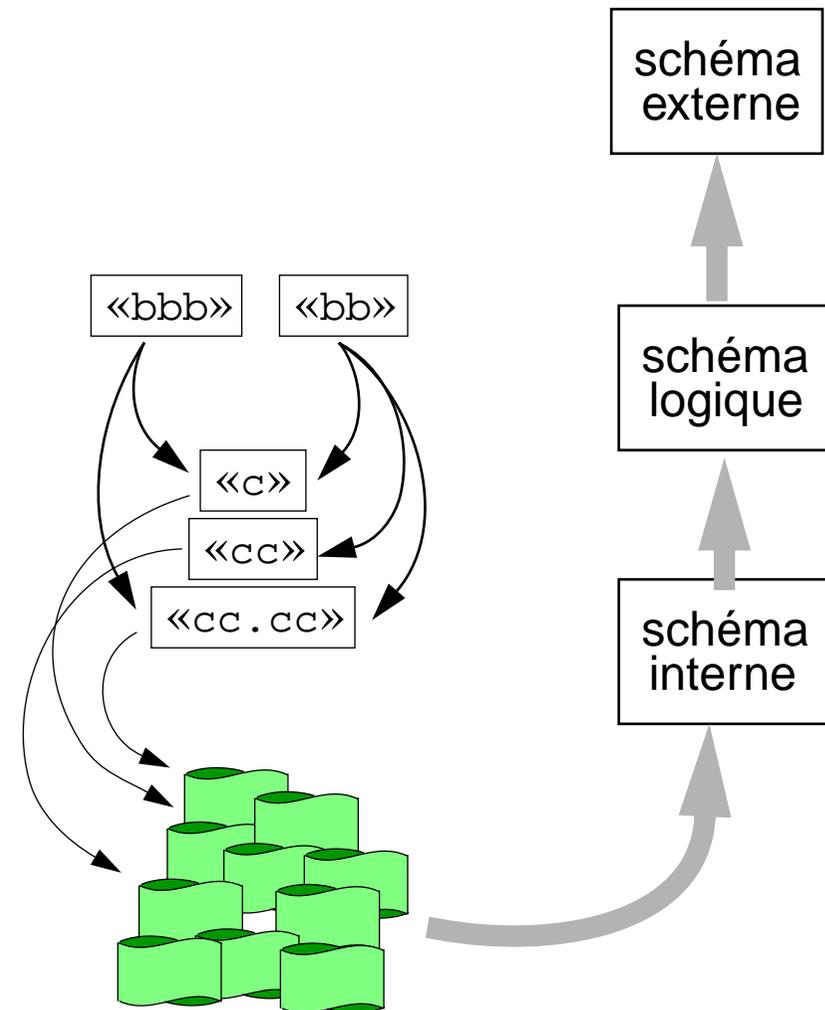
Cette traduction se fait dans la couche logique, et est accompagnée des contrôles sur la confidentialité, la concurrence d'accès, ...

4. Si la requête est acceptée, elle est optimisée puis **découpée en sous-requêtes plus élémentaires**, qui sont transmises à la couche interne; sinon elle peut être mise en attente ou refusée.





5. Au niveau interne, chaque sous-requête est **traduite en une ou plusieurs requêtes physiques**, en fonction des informations contenues dans le schéma interne; le SGBD réalise ensuite l'accès physique aux données (extraction ou modification).
6. Les éventuelles données extraites sont passées à la couche logique, puis à la couche externe, où elles sont ré-organisées en fonction du schéma externe de l'utilisateur.





## Modèle «entité-association» (1)

Le modèle *entité-association*  
(aussi appelé **modèle entité-relation**)  
est le modèle conceptuel de description statique  
utilisé dans la plupart des méthodes et outils d'aide à la  
conception de base de données (MERISE, IDA, ...)

Les concepts de base de ce modèle sont

- Les *entités*
- Les *associations*
- Les *attributs*



## Modèle «entité-association» (2)



Les *entités* permettent de représenter les objets (concrets ou abstraits) du monde réel à propos desquels on veut enregistrer des informations.

Les entités sont structurées par le biais de **types** qui regroupent des ensembles d'entités perçues comme similaires et ayant les mêmes caractéristiques.



Les *associations* permettent de représenter les liens entre entités, liens au sein desquels les entités peuvent jouer des *rôles* spécifiques (agent, objet, producteur, ...).

De même que les entités, les associations peuvent être typées, i.e. regroupées en ensembles homogènes d'associations (associations liant des entités de même type, avec les mêmes rôles et possédant les mêmes propriétés).



Les *attributs* permettent de représenter des propriétés associées à un type d'entité (TE), un type d'association (TA) ou participant à la définition d'un attribut complexe.

L'ensemble des attributs d'un TE ou TA représente l'ensemble des informations inhérentes que l'on souhaite conserver sur les entités ou les associations.

## Modèle «entité-association» (3)



### Exemples d'entités:

- un étudiant
- un enseignant
- un cours

### Exemples d'associations:

- suivre (un *étudiant* suit un *cours*)
- dispenser (un *enseignant* dispense un *cours*)

### Exemples d'attributs:

- nom, prénom, adresse,
- no de téléphone, no bancaire,
- cycle, matricule, nombre d'heures
- date de naissance, horaire,
- statut, état civile,...



## Identifiants



Un *identifiant* de TE (respectivement de TA) est un **ensemble minimum d'attributs** tel qu'il n'existe pas deux occurrences de TE (respectivement de TA) partageant **simultanément** les mêmes valeurs pour cet ensemble d'attributs.

Un identifiant permet donc de référencer de façon **non ambiguë** une occurrence de TE ou de TA.

Par exemple, le **matricule** peut servir d'identifiant pour les *étudiants*



# Attributs

Un *attribut* est décrit par les spécifications suivantes:

- Le nom (unique) de l'attribut
- Définition sous forme de texte libre
- Ses **cardinalités minimale et maximale**, i.e. les nombres minimum et maximum de valeurs autorisées pour cet attribut dans une occurrence de TE ou de TA, ou dans l'attribut dont il est un composant.
- Si l'attribut est **composé**, la description de ses attributs composants; sinon (i.e. l'attribut est **simple**) son **domaine de valeurs** (l'ensemble des valeurs autorisées)

Exemple:

<b>Attribut «Date de naissance»</b>	
<b>nom</b>	date de naissance
<b>définition</b>	Indique les jour, mois et année de naissance d'une personne
<b>cardinalités</b>	min=1, max=1 ( <i>toute personne à exactement une date de naissance</i> )
<b>type:</b>	jour ( <i>fait l'objet d'une définition séparée</i> )
	mois ( <i>définition séparée</i> )
	année ( <i>définition séparée</i> )

## Interprétation des cardinalités

<b>min = 0</b>	l'attribut est <b>facultatif</b> ( <i>exemple: un numéro de téléphone privé</i> )	<b>max = 1</b>	l'attribut est <b>monovalué</b> ( <i>exemple: une date de naissance</i> )
<b>min = 1</b>	l'attribut est <b>obligatoire</b> ( <i>exemple: une date de naissance</i> )	<b>max = n</b>	l'attribut est <b>multivalué</b> ( <i>exemple: les prénoms d'une personne</i> )



## Récapitulatif des types d'attributs

:

- ⇒ *attribut simple*: (type = **simple**)  
un attribut qui n'est pas composé (et est donc associé à un domaine de valeurs).  
Exemple: salaire (encore que...), couleur, ...
- ⇒ *attribut complexe*: (type = **composé**)  
un attribut composé  
Exemple: date (jour+mois+année), ...
- ⇒ *attribut monovalué*: (max = 1)  
un attribut qui ne peut prendre qu'une seule valeur par occurrence  
Exemple: date de naissance, nom, ...
- ⇒ *attribut multivalué*: (max = n)  
un attribut qui peut prendre plusieurs valeurs par occurrence  
Exemple: prénom, n° de téléphone, ...
- ⇒ *attribut facultatif*: (min = 0)  
un attribut qui peut ne pas prendre de valeur dans une occurrence  
Exemple: salaire, n° de téléphone, ...
- ⇒ *attribut obligatoire*: (min = 1)  
un attribut qui doit prendre au moins une valeur par occurrence  
Exemple: nom, couleur, ...



Un *type d'entité* (TE) est décrit par les spécifications suivantes:

- Le nom (unique) du TE
- Une définition sous la forme d'un texte libre
- La description des éventuels attributs du TE
- La composition des éventuels identifiants du TE.

**Exemple:**

<b>Type Entité «cours»</b>	
<b>nom</b>	cours
<b>définition</b>	regroupe les informations gérées par le SAC sur les cours dispensés à l'EPFL
<b>attributs:</b>	attribut-1: nom_cours
	attribut-2: cycle_section
<b>identifiant</b>	attribut-1+attribut-2



Un *type d'association* (TA) est décrit par les spécifications suivantes:

- Le nom (unique) du TA
- Une définition sous la forme d'un texte libre
- Les noms des TE participant au TA, avec le nom du rôle les associant au TA
- Pour chaque rôle, sa cardinalité minimale et maximale, i.e. le nombre min et max d'occurrences du TE qui peuvent, à un instant donné, être liés par ce rôle à une occurrence du TA
- La description des éventuels attributs du TA
- La composition des éventuels identifiants du TA.

**Exemple:**

<i>Type association «suit»</i>	
<b>nom</b>	suit
<b>définition</b>	définit les cours suivis par un étudiant sous la forme: <i>un étudiant suit un cours</i>
<b>TE participants</b>	<étudiant (1:n),>, <cours (0:n)>
<b>attributs:</b>	attribut-1: identifiant de étudiant
	attribut-2: identifiant de cours
<b>identifiant</b>	attribut-1+attribut-2

## Interprétation des cardinalités

<b>min = 0</b>	le rôle est <b>facultatif</b> ( <i>exemple: un cours peut n'être suivi par aucun étudiant</i> )	<b>max = 1</b>	le rôle est <b>exclusif</b> ( <i>un cours ne peut être donné que dans 1 seule salle</i> )
<b>min = 1</b>	le rôle est <b>obligatoire</b> ( <i>un cours doit être donné par au moins 1 professeur</i> )	<b>max = n</b>	le rôle est <b>duplicable</b> ( <i>un cours peut être suivi par plusieurs étudiants</i> )



## Occurrence et population

 On appelle *occurrence* d'un TE (respectivement d'un TA), toute entité (respectivement association) appartenant à l'ensemble décrit par le TE (respectivement le TA).

 On appelle *population* du TE (respectivement du TA), l'ensemble des occurrences du TE (respectivement du TA).

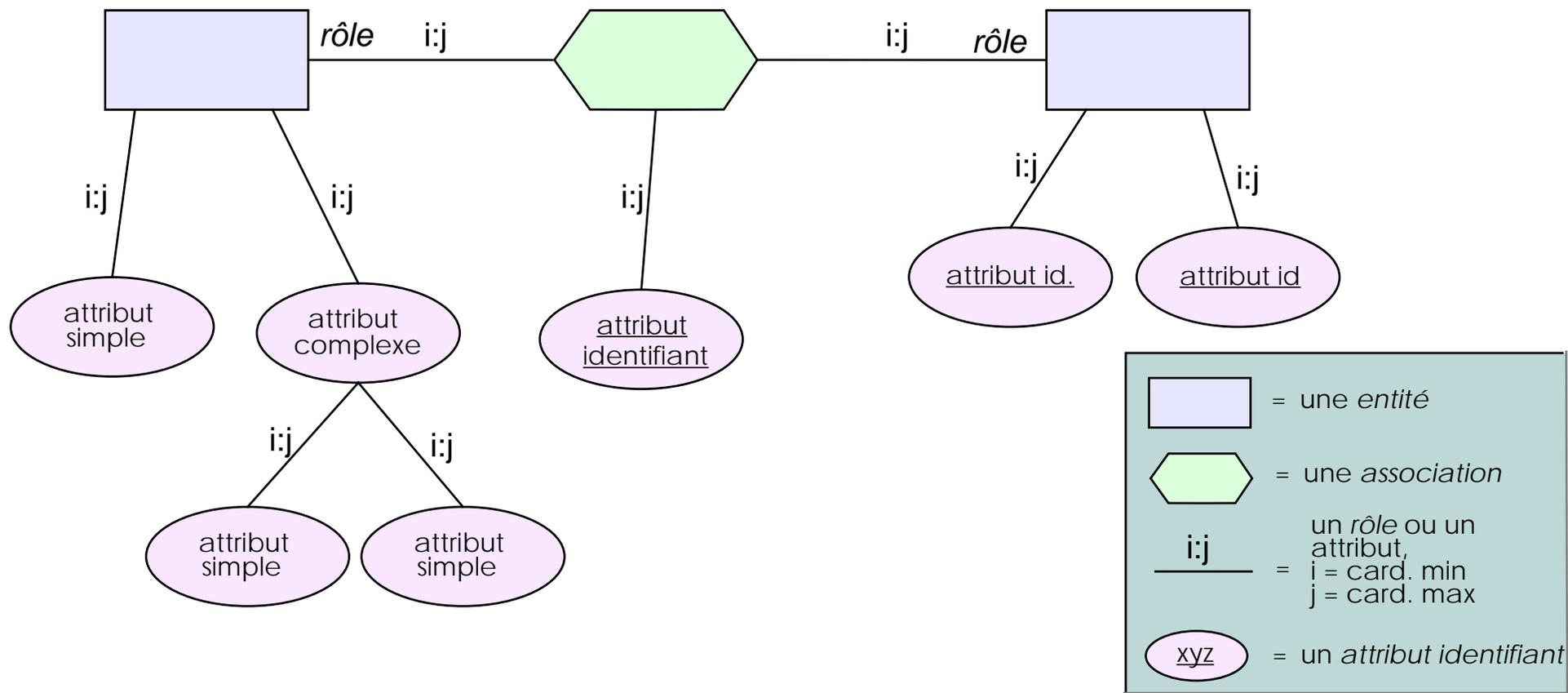
 La **base de données** décrite par un schéma entité-association est donc **l'ensemble des populations des TE et des TA** apparaissant dans le schéma.

 Les définitions des TE et des TA peuvent, de plus, être structurées au sein d'une **hiérarchie de généralisations**.



# Diagramme entité-association

Le modèle *entité-association* permet un **représentation graphique** [plus lisible] – appelée *diagramme* – du schéma d'une base de données.





## Contraintes d'intégrité

Les concepts d'entités, d'associations et d'attributs ne suffisent généralement pas pour décrire tout ce qui caractérise les données associée à un schéma.

On peut être amené à exprimer des règles additionnelles pour restreindre la combinatoire des occurrences autorisées par la description statique.

De telles règles, souvent exprimées dans un formalisme inspiré de la logique du 1<sup>er</sup> ordre sont appelées:  
*contraintes d'intégrité* (CI)

### Exemple de CI



Un cours  $c_1$  ne peut pas être prérequis pour un cours  $c_2$  s'il appartient à un cycle postérieur à celui de  $c_2$ :

$$\forall c_1 \in \text{Cours}, \forall c_2 \in \text{Cours}$$

$$(\text{Prérequis}(c_1, c_2) \Rightarrow c_2 \cdot \text{cycle} \geq c_1 \cdot \text{cycle})$$

Les arguments  $c_1$  et  $c_2$  de l'association «Prérequis( $c_1, c_2$ )» ont ici respectivement pour rôle: «*est-un*» et «*pour*»  
«Prérequis( $c_1, c_2$ )» signifie donc:  $c_1$  est-un Prérequis pour  $c_2$

Les contraintes d'intégrité les plus fréquentes limitent les valeurs possibles d'un attribut à certaines valeurs du domaine sous-jacent.



## Modèle logique: modèle relationnel

Le **modèle relationnel** [inventé en 1960],  
est actuellement le modèle logique le plus répandus  
parmi les SGBD du marché.

Son principal avantage est sa **grande simplicité**  
(ce qui lui permet d'être bien étudié sur le plan théorique, et facilement implantable) ...

... mais c'est également son principal défaut,  
car sa simplicité en fait un **outil sémantiquement trop pauvre**  
pour pouvoir correctement modéliser la complexité du monde réel  
(pour cela, d'autres modèles plus sophistiqués ont été développés,  
tels que les modèles orientés objets)



## Constituants d'un modèle relationnel

Dans le modèle relationnel, les *types d'entités* et les *types d'associations* sont représentés par un concept unique: *la relation*.

La relation représentant un TE ou un TA est un **tableau à deux dimensions**, usuellement appelé *table*, dont les colonnes sont associées aux **attributs** du TE ou du TA, et les lignes correspondent aux **occurrences** (également appelées *tuples*) du TE ou du TA.

<u>Matricule</u>	Nom	Prénom	Age
136	Dupont	Jean	19
141	Dupond	Annie	20
245	Duval	Annie	19
341	Dumond	Marc	19
...	...	...	...



Usuellement, le ou les attributs identifiant de la relation sont soulignés.



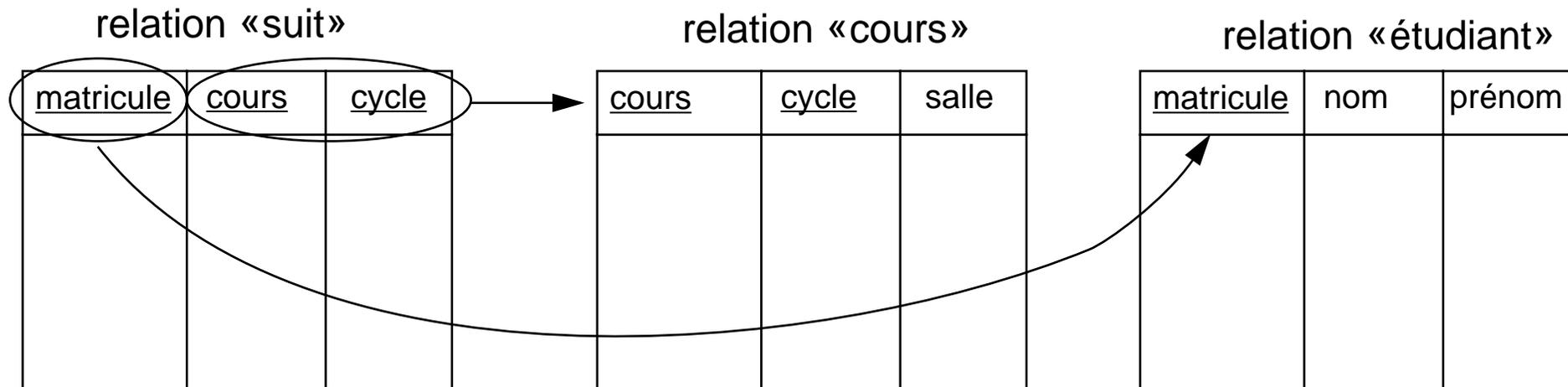
# Identifiants et identifiants externes

Comme dans le cas du modèle entité-association,

un *identifiant* de relation est un **ensemble minimal d'attributs**, tel qu'il n'existe pas deux tuples de la relation ayant des mêmes valeurs pour ces attributs.

Certains ensembles d'attributs d'une relation peuvent correspondre aux identifiants d'une autre relation.

Ces ensembles d'attributs sont alors appelés *identifiants externes*.





# Définition d'une relation

Une *relation* est décrite par les spécifications suivantes:

- le nom (unique) de la relation
- une définition, sous la forme d'un texte libre
- une liste d'attributs, chacun associé à un **domaine**
- le(s) **identifiant**(s)
- le(s) éventuel(s) **identifiant**(s) **externe**(s)

### Exemple de domaines:

Dnom: chaîne (caractères) de lng. maximal 30

Dnum: entier compris entre 0 et 99999

Dcol: {bleu, vert, rouge}

### Exemple:

<i>Relation «étudiant»</i>			
<b>nom</b>	Etudiant		
<b>définition</b>	ensemble des informations gérées par le SAC concernant les étudiants de l'EPFL		
<b>attributs</b>	matricule	<i>Domaine</i>	Dnum
	nom	<i>Domaine</i>	Dnom
<b>identifiants</b>	matricule		
<b>id. externes</b>	∅		



## Population, schéma et contraintes



La *population* d'une relation est l'ensemble de ses tuples.



Le **schéma** (logique) d'une base de données relationnelle est l'ensemble des schémas (logiques) de ses relations.



### **Contraintes imposées par le modèle relationnel:**

- Les attributs sont tous simples et monovalués (les notions d'attributs complexes, multivalués ou facultatifs n'existent pas dans le modèle relationnel).
- Toute relation a nécessairement au moins un identifiant.



# Règles de modélisation (1)

La prise en compte des attributs complexes du modèle entité-association peut se faire de trois façons:

- Les valeurs composites d'un attribut complexe sont considérées comme des valeurs atomiques (p.ex. de type chaîne de caractères); l'attribut est conservé, mais les valeurs de ses constituants ne sont plus individuellement accessibles (*perte d'informations structurelles*)
- L'attribut complexe est représenté par plusieurs attributs simples (i.e. plusieurs colonnes dans la table): les valeurs des constituants sont individuellement accessibles, mais l'attribut complexe n'existe plus en tant que tel (*perte d'informations structurelles*)
- L'attribut complexe est représenté par une relation (aggrégation des attributs élémentaires). La table représentant l'attribut complexe possède un identifiant (propre), et l'ensemble des identifiants externes des tables matérialisant les attributs élémentaires): cela revient à expliciter une relation de type: *est-constituant-de*. (*pas de perte d'information, mais complexification notable du schéma*)

date de naissance
"21 mars 1975"
"18 décembre 1961"
"21 mars 1975"

Jour	Mois	Année
21	3	1975
18	12	1961
21	3	1975

<u>date</u>	Jour	Mois	Année
d1	21	3	1975
d2	18	12	1961

date de naissance
d1
d2
d1



## Règles de modélisation (2)

La prise en compte des attributs multivalués du modèle entité-association se fait par la création d'une relation supplémentaire, associant les valeurs d'un identifiant de l'entité ou de l'association décrite par un attribut multivalué aux valeurs multiples de l'attribut.

<u>Matricule</u>	Nom	Prénoms
231	Dupont	Pierre, Claude
428	Durand	Claude, Jean



<u>Matricule</u>	Nom
231	Dupont
428	Durand



<u>Matricule</u>	<u>Prénom</u>
231	Pierre
231	Claude
428	Claude
428	Jean