Cours d'Informatique

"Bases de données"

I° année

Antoine Cornuéjols

www.lri.fr/~antoine

antoine.cornuejols@agroparistech.fr

http://www.lri.fr/~antoine/Courses/AGRO/TC/Cours-1A-BD-(v3)x2.pdf

1. Place de l'informatique

Version 2 - Janvier 2011

1.1 Nous et le monde

- **« Comprendre » le monde**
  - **Prédire**
  - **Modéliser**
    - Production d'un système causal « analogue »

Monde **complexe** :
- Beaucoup de facteurs
- Réseaux de relations complexe
- Evolution

- **Formalisation mathématique**
  - Identification de **facteurs causaux**
  - Expression des **interrelations**

Mais, **quand monde complexe**

- **Difficulté d'arriver** (directement) **à une modélisation correcte**
- Souvent, la **solution** est **non analytique**

1.1 Nous et le monde

- **« Comprendre » le monde**
  - **Prédire**
  - **Modéliser**
    - Production d'un système causal « analogue »

- **Formalisation informatique**
  - **Simulation**
  - **Construction d'un modèle**

- **Représenter** le monde
- **Calculer / inférer / raisonner**

1.2 L'informatique

- **Représenter** le monde
- **Calculer / inférer / raisonner**

1- Savoir **représenter**

- **Structures de données**
- **Représentation des connaissances**

2- **Tous les calculs ne sont pas possibles**

- Notion de **complexité**
  - Distribution / Parallélisation / « Cloud computing »

3- Il n'est **pas facile de programmer de manière fiable**

- **Méthodes de production de logiciel**
- **Vérification / Certification**

1.2 L'informatique

L'informatique est une science (comme les mathématiques ou la physique)

- **Ensemble organisé de *concepts* et d'*outils***
- Pour :
  - aider à **comprendre le monde**
  - créer des **artefacts**

1.2 Qu'est-ce que l'informatique

Science du traitement automatisé de l'information

- **Science :** un **côté théorique** (mathématique) + un **côté expérimental**
  (e.g. simulations (jeux), réseaux par paquets, multi-tâches, ...)
- **Traitement automatisé** : **algorithmes** (plus exigeant que les maths)
- **Information** : tout ce qui est **numérisable**
  (e.g. texte, images, musique, signal, ADN, ...)

1.2 Qu'est-ce que l'informatique

L'informatique concerne les abstractions ...

- **Choisir** les bonnes **abstractions**
- Utiliser **plusieurs niveaux** d'abstraction simultanément
- Définir les **relations** entre ces niveaux d'abstraction

Comme les maths

1.2 Qu'est-ce que l'informatique

L'informatique concerne les abstractions ...

... en faisant attention à :

- **L'efficacité**
  - Rapidité
  - Taille mémoire
  - Coût calcul
- Fonctionnement **correct**
  - Est-ce que cela fait ce que l'on veut ?
  - Est-ce que le programme donne une réponse ?
- **-ilité**
  - Simplicité et élégance
  - Utilisabilité
  - Modifiabilité
  - Maintenabilité
  - Coût
  - ...

Comme les sciences de l'ingénieur

1.3 L'informatique et **VOUS**

En quoi cela **vous** concerne ?

1– **Acteur direct** / en interaction directe avec l'informatique

- Programmation de simulation
- Utilisation de BD, de SIG
- Traitement d'images, télédétection
- Bio-informatique

E.g. AA :
- Croire que c'est magique
- Ne pas demander assez

2– En **interaction avec des acteurs directs**

- **Que puis-je attendre** de l'outil informatique ?
  - Qu'est-ce qui est **possible** / **difficile** / **impossible**
- Comment **interagir / dialoguer / orienter**

1.3 L'informatique et **VOUS**

- **Classes d'abstractions et de concepts**
  - Notion de **calcul** : entrée ; sortie ; spécification
  - Notion d'**algorithme** : organisation des calculs ; approximation ; heuristiques ; ...
  - **Complexité** : calculabilité ; performance
  - **Structures de données**
  - **Raisonnement** : correction ; logique ; heuristiques ; raisonnement imprécis ;
  - **Contrôle du calcul** : récursion ; itération ; non-déterminisme ; parallélisme ; distribution
  - **Communication** : information ; code ; synchrone/asynchrone ; P2P ; client-serveur ; ...
  - **Contraintes du monde physique** : tolérance aux fautes ; imprécisions ; coûts ; ...
  - etc.

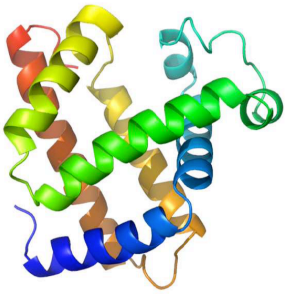
1.3 L'informatique et **VOUS**

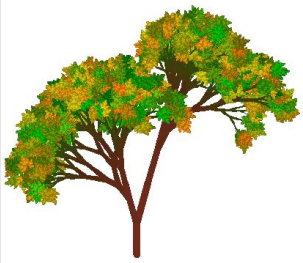
Des aspects multiples :

- **Informatique théorique** : algorithmique, automates, logique, calculabilité, complexité, théorie des graphes, . . .
- **Programmation** : plusieurs paradigmes (***impératif***, fonctionnel, logique, orienté objet, ...)
- **Réseau** : transmission de l'information (protocoles, routage, serveurs, ...)
- **Sécurité** : cryptologie, vérification de programme, ...
- **Architecture de systèmes** : code assembleur, gestion de la mémoire, ...
- **Intelligence artificielle** : raisonnement, apprentissage, représentation des connaissances, ...
- Et **beaucoup d'autres** : ***bases de données***, bio-informatique, optimisation, ...

1.4 L'informatique comme outil de pensée en biologie

- **La génétique comme un langage** : codage / décodage
- Les **algorithmes de séquencement** (shotgun) ont permis le ***déchiffrement du génome***
- Les **réseaux booléens** permettent de ***modéliser la dynamique des réseaux biologiques***
- Le **calcul des processus** permet de ***modéliser l'interaction entre molécules***
- Le **calcul sur graphe** permet de modéliser la :
  - *diffusion d'un virus*
  - *la compétition entre espèces*
- Les **systèmes multi-agents** permettent de modéliser les :
  - *insectes sociaux*
  - *bancs de poissons*
- La **récursion** permet de modéliser la ***croissance des plantes***
- ...



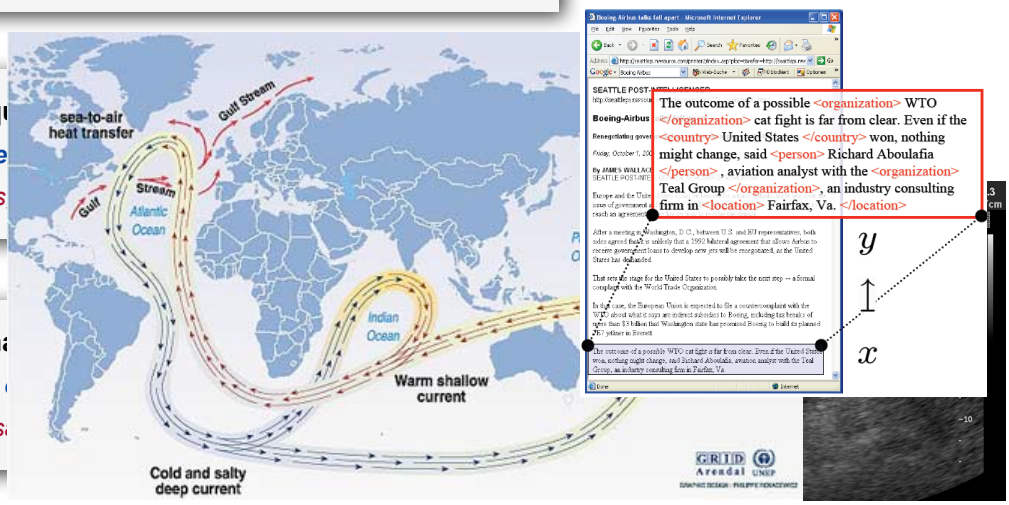


1.4 Informatique et recherche à AgroParisTech

- Étude des **échanges thermiques dans l'Atlantique Nord** :
  - ***Le Gulf-Stream est-il moins actif ?***
  - *Grosses bases de données réparties ; apprentissage artificiel ; Systèmes experts, ...*

- **Étude du risqu**
  - ***à partir de***
  - *Ontologies*

- Analyse en **ima**
  - ***détection***
  - *Apprentiss*



2. Les Bases de Données

Version 1 - Janvier 2010

2. Les Bases de Données

Contenu

2.1 Quelles données ?

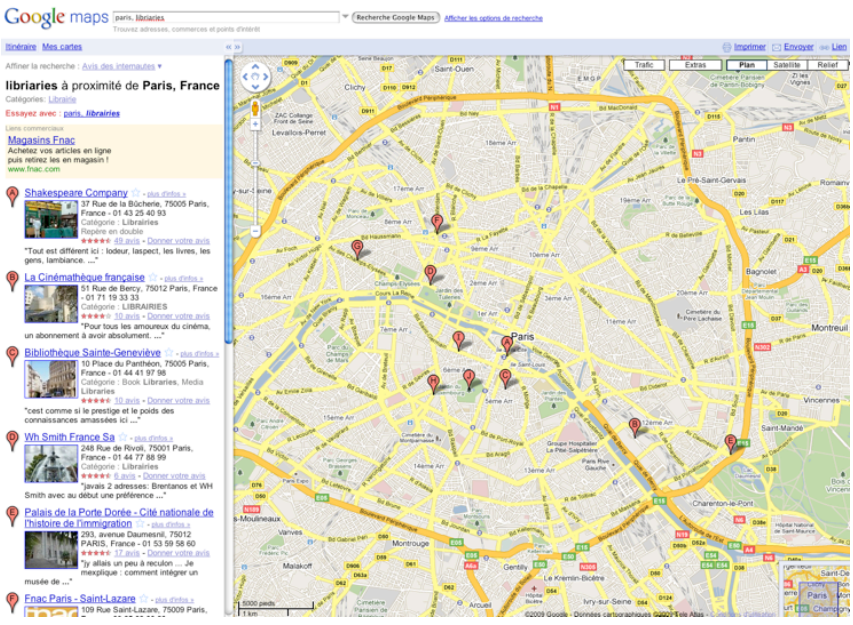
Les données sont omniprésentes autour de nous ...

- Base de données sur des **films (e.g. Allociné)**
- Base de données **bibliothécaire (e.g. BN, Amazon, ...)**
- **Marathon de New-York (Paris, Londres, du Médoc, ...)**
- **Location de voitures (e.g. Hertz, ...) / Réservation de place d'avion**
- **Sécurité sociale (carte Vital), Hôpitaux, ...**
- **Communications téléphoniques (opérateurs)**
- **Second Life**
- **Systèmes d'information géographique**
- **...**

Partie 1 - Place de l'informatique
Partie 2 - Les bases de données
Partie 3 - Conception des BD
Partie 4 - Manipulation : SQL
Partie 5 - PHP / MySQL
2.1 Des données
2.2 Comment les stocker
2.3 Comment les structurer
2.4 Utiliser une BD
1.5 Les SGBD
1.6 Notions de base

2.1 Quelles données ?

Les données sont omniprésentes autour de nous ...



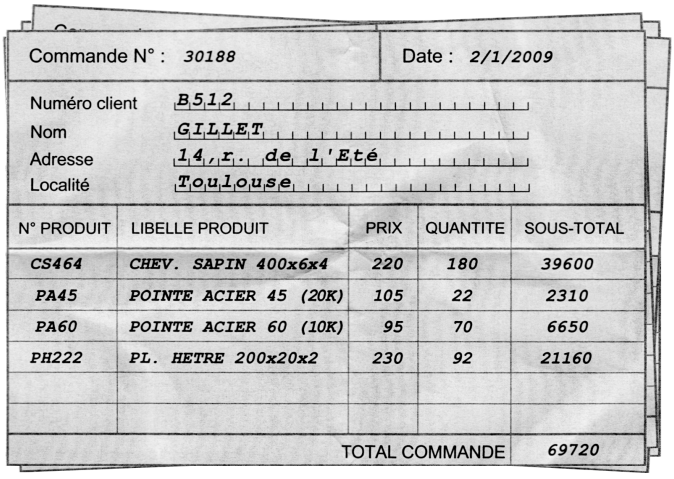
2.1 Quelles données ?

Les données sont omniprésentes autour de nous ... dans l'entreprise

- Les ***clients***
- Les ***produits***
- Les ***commandes***
- Les ***factures***
- **...**

2.1 Quelles données ?

Les données sont omniprésentes autour de nous ...



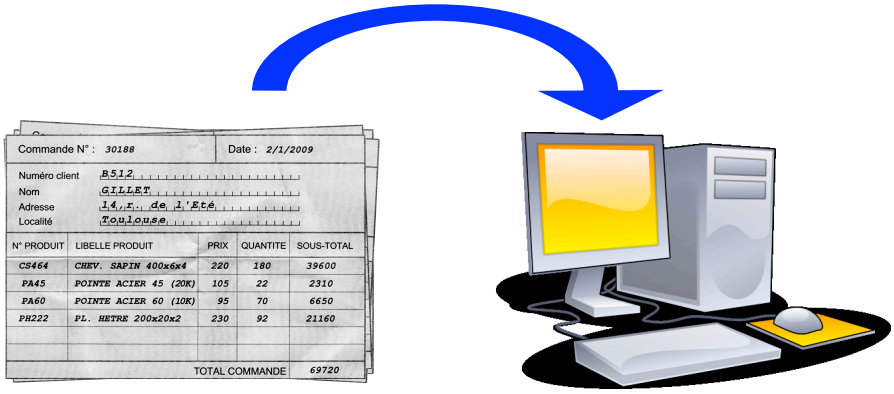
Commande N° : 30188 — Date : 2/1/2009

Numéro client B512
Nom GILLET
Adresse 14, r. de l'Eté
Localité Toulouse

N° PRODUIT	LIBELLE PRODUIT	PRIX	QUANTITE	SOUS-TOTAL
CS464	CHEV. SAPIN 400x6x4	220	180	39600
PA45	POINTE ACIER 45 (20K)	105	22	2310
PA60	POINTE ACIER 60 (10K)	95	70	6650
PH222	PL. HETRE 200x20x2	230	92	21160
			TOTAL COMMANDE	69720

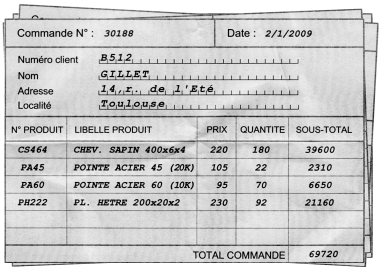
2.2 Comment les stocker ?

Comment les ranger dans un ordinateur ?



2.2 Comment les stocker ?

Comment les ranger dans un ordinateur ?



- **scanner les bons de commande [1/10]**
- **copier les données dans un tableau Word [2/10]**
- **copier les données dans une feuille Excel [4/10]**
- **copier les données dans une *base de données* [10/10]**

2.3 Comment structurer les données ?

En regardant d'un peu plus près ...

Commande N° : 30188 — Date : 2/1/2009

Numéro client : B512
Nom : GILLET
Adresse : 14, r. de l'Eté
Localité : Toulouse

données du client

N° PRODUIT	LIBELLE PRODUIT	PRIX	QUANTITE	SOUS-TOTAL
CS464	CHEV. SAPIN 400x6x4	220	180	39600
PA45	POINTE ACIER 45 (20K)	105	22	2310
PA60	POINTE ACIER 60 (10K)	95	70	6650
PH222	PL. HETRE 200x20x2	230	92	21160
	TOTAL COMMANDE			69720

données d'un détail

données de la commande

2.3 Comment structurer les données ?

Reportons ces données dans des tableaux :

données de la commande

NCOM	DATECOM	TOTAL-COMMANDE
30188	2/1/2009	69720

données du client

NCLI	NOM	ADRESSE	LOCALITE
B512	GILLET	14,r. de l'Eté	Toulouse

données des détails

NPRO	LIBELLE	PRIX	QCOM	SOUS-TOTAL
CS464	CHEV. SAPIN 400x6x4	220	180	39600
PA45	POINTE ACIER 45 (2K)	105	22	2310
PA60	POINTE ACIER 60 (1K)	095	70	6650
PH222	PL. HETRE 200x20x2	230	92	21160

Observations :

1. **les données TOTAL-COMMANDE et SOUS-TOTAL sont calculées : inutile de les conserver, on pourra les recalculer en cas de besoin**
2. **il est impossible de reconstituer le bon de commande d'origine : quel est le client de la commande, quelle est la commande d'un détail ?**

2.3 Comment structurer les données ?

Données sans redondances et avec références :

données de la commande

NCOM	NCLI	DATECOM
30188	B512	2/1/2009

données du client

NCLI	NOM	ADRESSE	LOCALITE
B512	GILLET	14,r. de l'Eté	Toulouse

données des détails

NCOM	NPRO	QCOM	LIBELLE	PRIX
30188	CS464	180	CHEV. SAPIN 400x6x4	220
30188	PA45	22	POINTE ACIER 45 (2K)	105
30188	PA60	70	POINTE ACIER 60 (1K)	095
30188	PH222	92	PL. HETRE 200x20x2	230

Observation

si plusieurs détails mentionnent le même produit, ses caractéristiques sont répétées autant de fois : on isole les données des produits dans un tableau spécifique

2.3 Comment structurer les données ?

Distribution optimale des données des bons de commande

données de la commande

NCOM	NCLI	DATECOM
30188	B512	2/1/2009

données du client

NCLI	NOM	ADRESSE	LOCALITE
B512	GILLET	14,r. de l'Eté	Toulouse

données des détails

NCOM	NPRO	QCOM
30188	CS464	180
30188	PA45	22
30188	PA60	70
30188	PH222	92

données des produits

NPRO	LIBELLE	PRIX
CS464	CHEV. SAPIN 400x6x4	220
PA45	POINTE ACIER 45 (2K)	105
PA60	POINTE ACIER 60 (1K)	95
PH222	PL. HETRE 200x20x2	230

2.3 Comment structurer les données ?

Ajoutons d'autres données : *notre première base de données*

COMMANDE

NCOM	NCLI	DATECOM
30178	K111	22/12/2008
30179	C400	22/12/2008
30182	S127	23/12/2008
30184	C400	23/12/2008
30185	F011	2/01/2009
30186	C400	2/01/2009
30188	B512	2/01/2009

DETAIL

NCOM	NPRO	QCOM
30178	CS464	25
30179	CS262	60
30179	PA60	20
30182	PA60	30
30184	CS464	120
30184	PA45	20
30185	CS464	260
30185	PA60	15
30185	PS222	600
30186	PA45	3
30188	CS464	180
30188	PA45	22
30188	PA60	70
30188	PH222	92

CLIENT

NCLI	NOM	ADRESSE	LOCALITE	(CAT)	COMPTE
B062	GOFFIN	72, r. de la Gare	Namur	B2	-3200
B112	HANSENNE	23, r. Dumont	Poitiers	C1	1250
B332	MONTI	112, r. Neuve	Genève	B2	0
B512	GILLET	14, r. de l'Eté	Toulouse	B1	-8700
C003	AVRON	8, r. de la Cure	Toulouse	B1	-1700
C123	MERCIER	25, r. Lemaître	Namur	C1	-2300
C400	FERARD	65, r. du Tertre	Poitiers	B2	350
D063	MERCIER	201, bvd du Nord	Toulouse		-2250
F010	TOUSSAINT	5, r. Godefroid	Poitiers	C1	0
F011	PONCELET	17, Clos des Erables	Toulouse	B2	0
F400	JACOB	78, ch. du Moulin	Bruxelles	C2	0
K111	VANBIST	180, r. Florimont	Lille	B1	720
K729	NEUMAN	40, r. Bransart	Toulouse		0
L422	FRANCK	60, r. de Wépion	Namur	C1	0
S127	VANDERKA	3, av. des Roses	Namur	C1	-4580
S712	GUILLAUME	14a, ch. des Roses	Paris	B1	0

PRODUIT

NPRO	LIBELLE	PRIX	QSTOCK
CS262	CHEV. SAPIN 200x6x2	75	45
CS264	CHEV. SAPIN 200x6x4	120	2690
CS464	CHEV. SAPIN 400x6x4	220	450
PA45	POINTE ACIER 45 (20K)	105	580
PA60	POINTE ACIER 60 (10K)	95	134
PH222	PL. HETRE 200x20x2	230	782
PS222	PL. SAPIN 200x20x2	185	1220

2.3 Comment structurer les données ?

Quelques leçons partielles

- On a besoin de **systèmes spécifiques** pour conserver et manipuler les données
- Les données sont représentées dans des **tables**
  - Les **lignes** sont des entités
  - Les **colonnes** sont des propriétés
- Le **découpage en tables** demande **une analyse** et une optimisation
- L'analyse doit aller au-delà des besoins immédiats (***abstraction***)

2.4 Utiliser une base de données

Que peut-on faire de ces données ?

Avant tout, **les conserver** aussi longtemps que nécessaire !

Les interroger : *quel est le numéro, le nom et l'adresse des clients de Toulouse ?*

```
select NCLI, NOM, ADRESSE
from   CLIENT
where  LOCALITE = 'Toulouse';
```

facile

ou encore : *quelles sont les commandes des clients de Toulouse ?*

```
select NCOM
from   COMMANDE
where  NCLI in (select NCLI
                from CLIENT where LOCALITE = 'Toulouse');
```

un peu plus difficile

requêtes rédigées dans le langage SQL

2.4 Utiliser une base de données

Que peut-on faire de ces données ?

Vérifier une commande lors de son enregistrement : *le client est-il connu ? son adresse a-t-elle changé ? les produits commandés sont-ils répertoriés ?*

Produire les factures

Préparer le réapprovisionnement des produits en rupture de stock

Calculer le chiffre d'affaire mensuel

Etudier la **répartition géographique des ventes**

Et mille autres applications ...

2.4 Utiliser une base de données

Que peut-on faire de ces données ?

Un dernier exemple :

calculer** la **répartition** du **chiffre d'affaire par localité** et **par produit

```
select C.LOCALITE, P.NPRO, sum(D.QCOM*P.PRIX)
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
where  C.NCLI = M.NCLI and M.NCOM = D.NCOM
       and    D.NPRO = P.NPRO
group by C.LOCALITE, P.NPRO;
```

inutile d'essayer de comprendre pour l'instant !

Cette question complexe est résolue en une seule instruction SQL de 5 lignes !

2.5 Les SGBD

La gestion d'une base de données pose des problèmes complexes. Cette gestion est assurée par des logiciels spécialisés : les ***systèmes de gestion de bases de données*** ou ***SGBD***.

- **Organisation des données :** le SGBD organise les données en tables permanentes stockées sur disque; il crée les mécanismes garantissant un accès rapide aux données; il informe les utilisateurs sur ces structures.
- **Gestion des données :** le SGBD garantit l'évolution cohérente des données; il vérifie que les contraintes (unicité, référence entre tables, etc.) sont respectées.
- **Accès aux données :** le SGBD permet l'accès aux données à la fois par l'utilisateur occasionnel et par les programmes de traitement de données.

2.5 Les SGBD

- **Protection contre les accidents :**
  le SGBD garantit l'intégrité et l'accessibilité des données en cas d'incident ou d'attaque.
- **Gestion des accès concurrents :**
  le SGBD permet l'accès simultané aux données par des centaines voire des milliers d'utilisateurs. Il contrôle rigoureusement les opérations simultanées sur les mêmes données.
- **Contrôle des accès :**
  le SGBD garantit que seuls les utilisateurs autorisés peuvent accéder aux données et les modifier.

2.5 Les défis des BDs aujourd'hui et demain

- **Multiplicité des types de données.**
  Une base de données moderne peut contenir :
  - des données **multimédias**,
  - des données **textuelles**,
  - des données **spatiales** (données GPS par exemple),
  - des données **historiques** (plusieurs lignes par entité),
  - des données **semi-structurées.**
- **Volumes et performances.**
  - Une base de données peut contenir des dizaines de milliers de tables, des **milliards de lignes**.
  - Comment garantir **l'accessibilité** de ces données, leur **protection** contre les incidents, des **temps d'accès satisfaisants** ?

2.5 Les défis des BDs aujourd'hui et demain

- **Maintenance et évolution.**
  - La structure d'une base de données peut évoluer : **ajouter** ou **supprimer** une table, une colonne, une contrainte.
  - **Comment préserver les données et les programmes** utilisateurs lors de cette évolution ?

- **Les données distribuées et nomades.**
  - Une base de données peut être répartie et/ou dupliquées sur plusieurs ordinateurs répartis géographiquement.
  - Certains de ceux-ci peuvent être des **appareils mobiles** (embarqués, portables, *smart phones*).
  - Comment garantir la **cohérence**, la **protection** et l'**accessibilité** des **données**.

2.5 Les défis des BDs aujourd'hui et demain

- **Les BD et le Web.** De nombreuses bases de données sont intimement liées au Web. En outre, le Web peut être vu comme une gigantesque base de données (largement incohérente et redondante !)

  Qu'en est-il des principes rigoureux des bases de données dans ce contexte ?

- **Les données décisionnelles.**

  Les données ne servent pas seulement à contrôler la gestion et le fonctionnement d'une entreprise au jour le jour. Elles sont aussi **souvent utilisées pour soutenir des décisions tactiques et stratégiques**.

  Caractéristiques : traitement de **très gros volumes de données complexes** pour produire une réponse courte (une heure de traitement de plusieurs téraoctets pour en extraire un seul nombre !)

  Comment structurer une base de données dans ce sens (= *entrepôts de données*) ?

2.6 Notions de base

Bases ... sur les Bases de Données

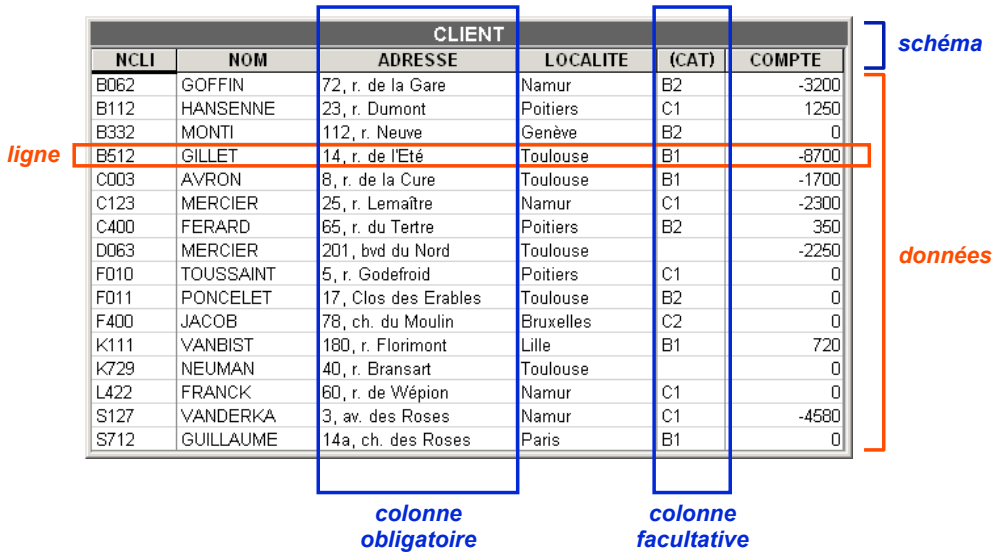
2.6 Notions de base

Base de données = ensemble de tables

- Chaque ***table*** a un **nom unique** (*ex : Film*)
- Chaque ***table*** contient les données relatives à des ***entités*** *de même nature*.
- Chaque ***ligne*** (***enregistrement***) d'une table décrit les données relatives à **une entité**.
- Chaque ***colonne*** d'une table décrit une **propriété** des entités (*ex : Prix*).
- Les lignes d'une table sont *distinctes*.
- Les **noms de table** et de **colonnes** constituent le ***schéma*** de la base.
- Les **lignes** (entités) constituent le ***contenu*** de la base.

2.6 Notions de base

Tables, lignes et colonnes



2.6 Notions de base

Le **schéma d'une table** **définit sa structure**. Il spécifie notamment :

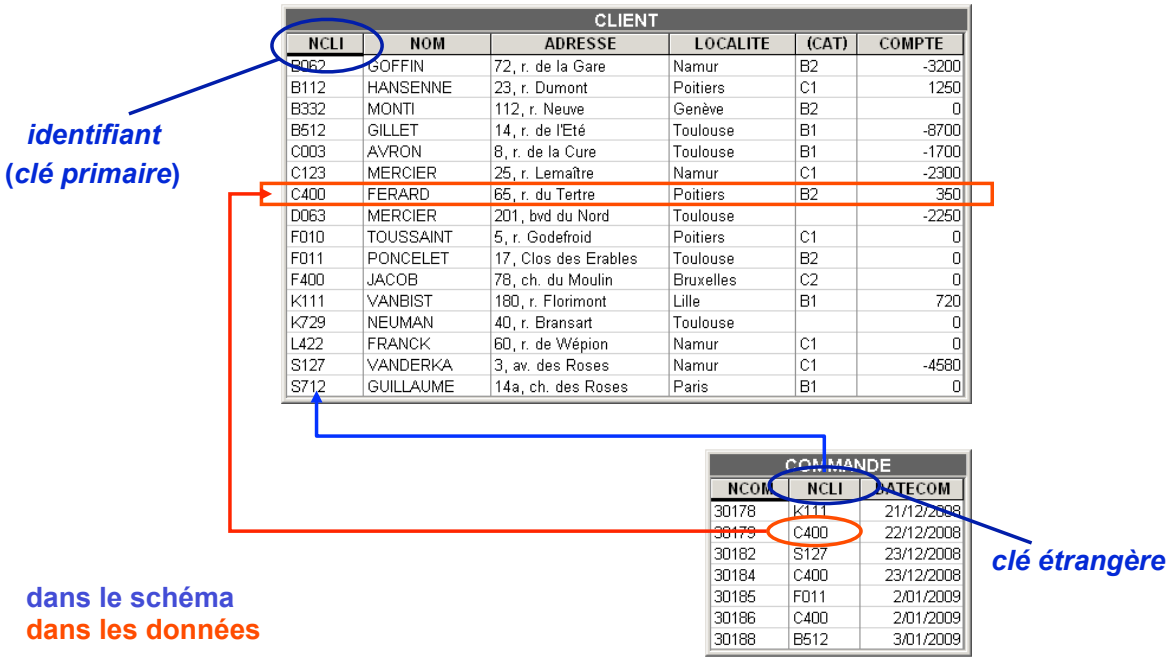
1. le **nom** de la table,
2. pour chaque **colonne**, son nom, son type, son caractère obligatoire,
3. l'**identifiant primaire** (liste de colonnes)
4. les **identifiants secondaires** éventuels (liste de colonnes)
5. les **clés étrangères** éventuelles (liste de colonnes et table cible).

Le **contenu d'une table** est formé d'un ensemble de lignes conformes au schéma.

Le **contenu** d'une table est sujet à de fréquentes modifications.

Le **schéma** d'une table peut évoluer mais moins fréquemment.

2.6 Notions de base



CLIENT

NCLI	NOM	ADRESSE	LOCALITE	(CAT)	COMPTE
B062	GOFFIN	72, r. de la Gare	Namur	B2	-3200
B112	HANSENNE	23, r. Dumont	Poitiers	C1	1250
B332	MONTI	112, r. Neuve	Genève	B2	0
B512	GILLET	14, r. de l'Eté	Toulouse	B1	-8700
C003	AVRON	8, r. de la Cure	Toulouse	B1	-1700
C123	MERCIER	25, r. Lemaître	Namur	C1	-2300
C400	FERARD	65, r. du Tertre	Poitiers	B2	350
D063	MERCIER	201, bvd du Nord	Toulouse		-2250
F010	TOUSSAINT	5, r. Godefroid	Poitiers	C1	0
F011	PONCELET	17, Clos des Erables	Toulouse	B2	0
F400	JACOB	78, ch. du Moulin	Bruxelles	C2	0
K111	VANBIST	180, r. Florimont	Lille	B1	720
K729	NEUMAN	40, r. Bransart	Toulouse		0
L422	FRANCK	60, r. de Wépion	Namur	C1	0
S127	VANDERKA	3, av. des Roses	Namur	C1	-4580
S712	GUILLAUME	14a, ch. des Roses	Paris	B1	0

COMMANDE

NCOM	NCLI	DATECOM
30178	K111	21/12/2008
30179	C400	22/12/2008
30182	S127	23/12/2008
30184	C400	23/12/2008
30185	F011	2/01/2009
30186	C400	2/01/2009
30188	B512	3/01/2009

2.6 Notions de base

Identifiants et clés étrangères

Un **identifiant** est un **groupe de colonnes** d'une table T tel qu'il ne puisse, à tout moment, exister plus d'une ligne dans T qui possède des valeurs déterminées pour ces colonnes.

La valeur de l'identifiant permet de **désigner une ligne de T**.

Une **clé étrangère** est un groupe de colonnes d'une table S tel qu'il existe, à tout moment, dans une table T, une ligne dont l'identifiant a pour valeur(s) celle(s) de ce groupe.

La valeur de la clé étrangère **dans S** sert à **référencer une ligne de la table T**.

2.6 Identifiants et clés étrangères

Une table peut posséder **plusieurs identifiants**. On choisit l'un d'eux, qu'on déclare **primaire (clé primaire)**. Les autres sont dès lors **secondaires**.

L'**identifiant primaire** est constitué de colonnes obligatoires.

Un **identifiant est minimal** si chacune de ses colonnes est nécessaire pour garantir la contrainte d'unicité.

Il est possible de déclarer une **table sans identifiant** mais ceci n'est **pas recommandé**.

Partie 1 - Place de l'informatique
Partie 2 - Les bases de données
Partie 3 - Conception des BD
Partie 4 - Manipulation : SQL
Partie 5 - PHP / MySQL
2.1 Des données
2.2 Comment les stocker
2.3 Comment les structurer
2.4 Utiliser une BD
1.5 Les SGBD
1.6 Notions de base

2.6 Identifiants et clés étrangères

Une **clé étrangère** est associée à une **contrainte référentielle**.

Une clé étrangère référence en principe l'identifiant primaire de la table cible. Elle peut référencer un identifiant secondaire mais ceci n'est pas recommandé.

Une clé étrangère et l'identifiant qu'elle référence ont la même composition : même nombre de colonnes et colonnes de mêmes types prises deux à deux.

Contrainte référentielle : il faut que chaque clé étrangère désigne une ligne existante de la table cible.

2.6 Identifiants et clés étrangères

Un **identifiant minimal** est aussi appelé **clé candidate** (***candidate key***). [*]

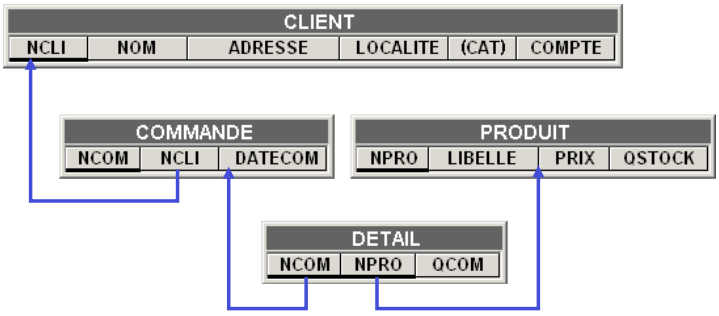
Un **identifiant primaire** s'appelle aussi **clé primaire** (***primary key***).

Clé étrangère = ***foreign key***.

[*] problème : le terme **clé** admet plus de 20 acceptions différentes dans le domaine des bases de données !

2.6 Notions de base - *Exemple*

Un schéma



2.6 Notions de base - *Exemple*

Les données

CLIENT

NCLI	NOM	ADRESSE	LOCALITE	(CAT)	COMPTE
B062	GOFFIN	72, r. de la Gare	Namur	B2	-3200
B112	HANSENNE	23, r. Dumont	Poitiers	C1	1250
B332	MONTI	112, r. Neuve	Genève	B2	0
B512	GILLET	14, r. de l'Eté	Toulouse	B1	-8700
C003	AVRON	8, r. de la Cure	Toulouse	B1	-1700
C123	MERCIER	25, r. Lemaître	Namur	C1	-2300
C400	FERARD	65, r. du Tertre	Poitiers	B2	350
D063	MERCIER	201, bvd du Nord	Toulouse		-2250
F010	TOUSSAINT	5, r. Godefroid	Poitiers	C1	0
F011	PONCELET	17, Clos des Erables	Toulouse	B2	0
F400	JACOB	78, ch. du Moulin	Bruxelles	C2	0
K111	VANBIST	180, r. Florimont	Lille	B1	720
K729	NEUMAN	40, r. Bransart	Toulouse		0
L422	FRANCK	60, r. de Wépion	Namur	C1	0
S127	VANDERKA	3, av. des Roses	Namur	C1	-4580
S712	GUILLAUME	14a, ch. des Roses	Paris	B1	0

PRODUIT

NPRO	LIBELLE	PRIX	QSTOCK
CS262	CHEV. SAPIN 200x6x2	75	45
CS264	CHEV. SAPIN 200x6x4	120	2690
CS464	CHEV. SAPIN 400x6x4	220	450
PA45	POINTE ACIER 45 (2K)	105	580
PA60	POINTE ACIER 60 (1K)	95	134
PH222	PL. HETRE 200x20x2	230	782
PS222	PL. SAPIN 200x20x2	185	1220

DETAIL

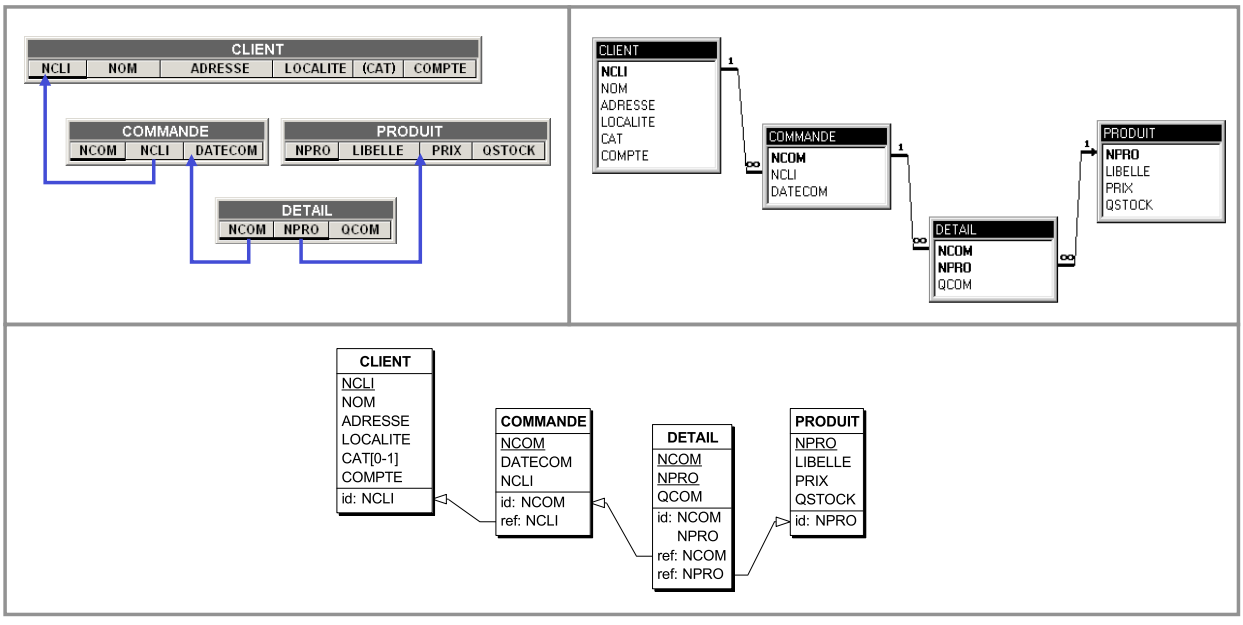
NCOM	NPRO	QCOM
30178	CS464	25
30179	CS262	60
30179	PA60	20
30182	PA60	30
30184	CS464	120
30184	PA45	20
30185	CS464	260
30185	PA60	15
30185	PS222	600
30186	PA45	3
30188	CS464	180
30188	PA45	22
30188	PA60	70
30188	PH222	92

COMMANDE

NCOM	NCLI	DATECOM
30178	K111	21/12/2008
30179	C400	22/12/2008
30182	S127	23/12/2008
30184	C400	23/12/2008
30185	F011	2/01/2009
30186	C400	2/01/2009
30188	B512	3/01/2009

2.6 Notions de base

Variantes de schéma



3 - Conception d'une base de données

Version 1 - janvier 2010

3.1 Pourquoi est-ce difficile de concevoir une BD

Comment construire une base de données ?

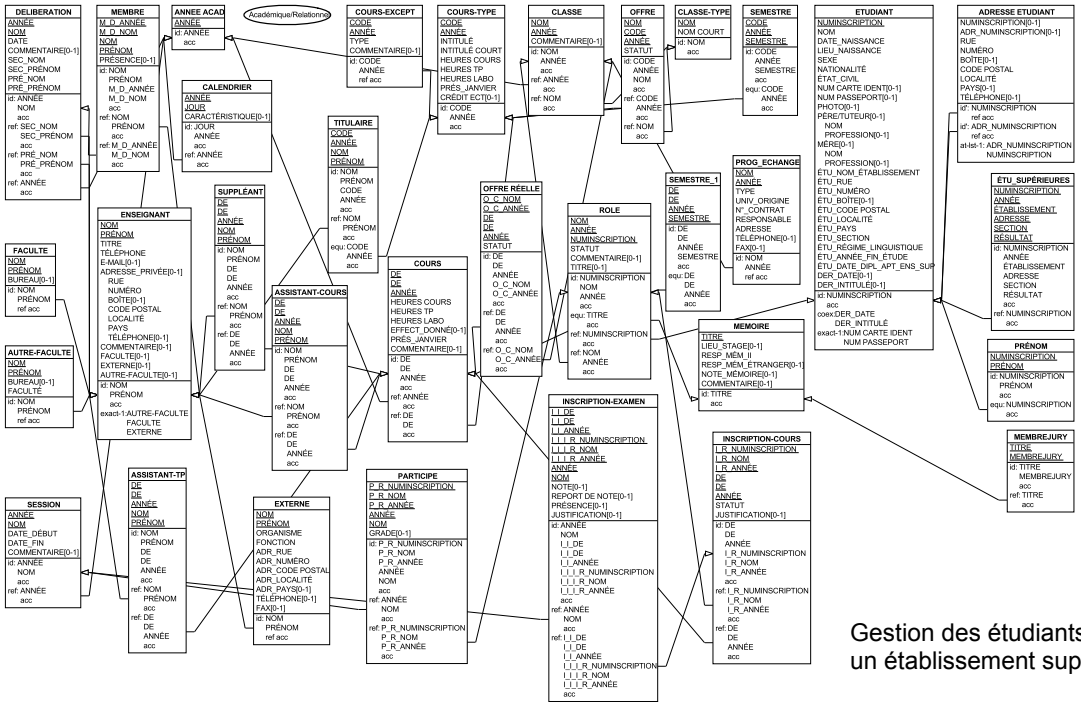
- définir les tables, les contraintes, les index, etc.
- introduire les données dans ces tables (ceci est un autre problème).

Encore faut-il que le schéma des tables satisfasse les besoins des utilisateurs des données !

On constate rapidement que les structures d'un schéma relationnel constituent un support inadéquat pour exprimer naturellement ces besoins.

L'esprit humain n'est pas conçu pour réfléchir en termes de tables !

3.1 Pourquoi est-ce difficile de concevoir une BD



Gestion des étudiants dans un établissement supérieur.

3.2 Le modèle Entité-Association (E/A)

Il faut un autre langage pour exprimer naturellement les informations que doit contenir une base de données :

le modèle Entité-association

Le **monde** est perçu comme formé d'**entités,** dotées de **propriétés** et en **associations** les unes avec les autres.

Le monde = ce dont on parle = l'univers du discours = la **partie du réel au sujet de laquelle on désire enregistrer de l'information** = **le domaine d'application**

=> *Modélisation*

3.2 Le modèle Entité-Association (E/A)

Que sont les ***besoins des utilisateurs*** **?**

- une base de données qui **contienne toutes les données** décrivant son domaine d'application **et elles seulement;**
- la **structure** de ces données doit être **simple, naturelle, expressive, sans redondance, ...**

On construit une base de données (en fait son schéma) en **deux phases :**

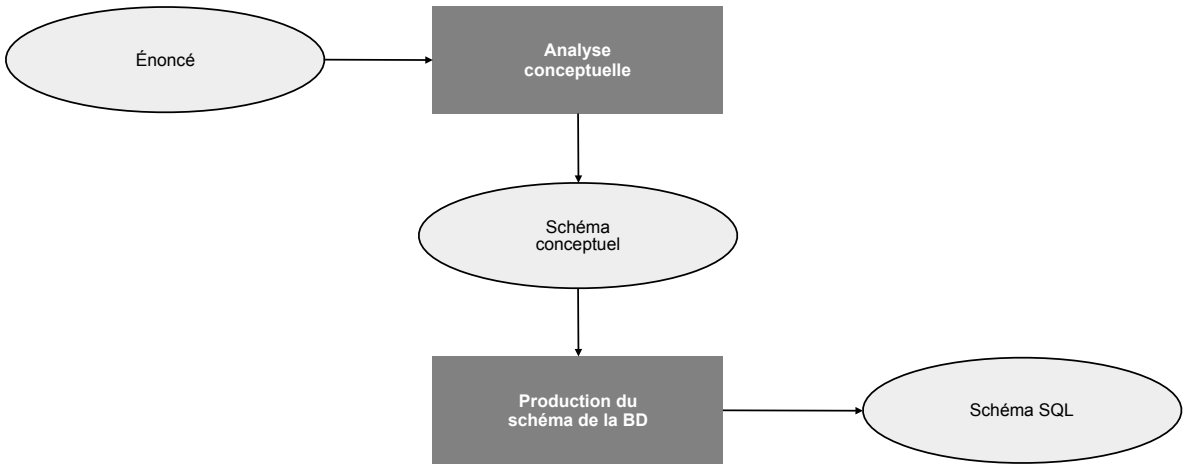
1. on **identifie** les **concepts pertinents** du domaine d'application, leurs **propriétés** et leurs **associations**

   = schéma conceptuel;

2. on **traduit** le schéma conceptuel **en structures de tables**

   = schéma de la base de données

3.2 Le modèle Entité-Association (E/A)

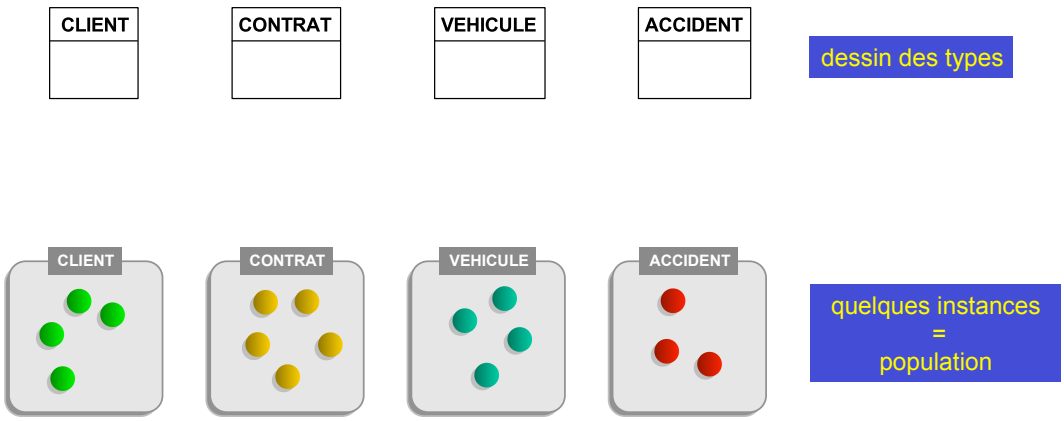


3.2 Le modèle Entité-Association (E/A)

Le **modèle Entité-association** propose une lecture du monde (le domaine d'application) simple mais limitée :

- *le **monde*** est constitué d'objets ou **entités**
- les ***entités*** sont classées en **types d'entités**
- *les **entités d'un type*** ont des **attributs** spécifiques
- *les **entités*** sont en **association** les unes avec les autres
- *les **associations*** sont classées en **types d'associations**.

3.3 Types d'entités



3.4 Attributs

CLIENT
NumClient
Nom
Adresse

CONTRAT
NumCtr
Type
DateSign

VEHICULE
NumVéh
Marque
Modèle
Année
Cylindrée

ACCIDENT
NumAcc
DateAcc
Montant

quelques attributs

⇓

NumClient = C400
Nom = FERARD
Adresse = 65, r. du Tertre

NumClient = B332
Nom = MONTI
Adresse = 112, r. Neuve

NumClient = F010
Nom = TOUSSAINT
Adresse = 5, r. Godefroid

quelques valeurs

2.3 Attributs - Type et attribut obligatoire/facultatif

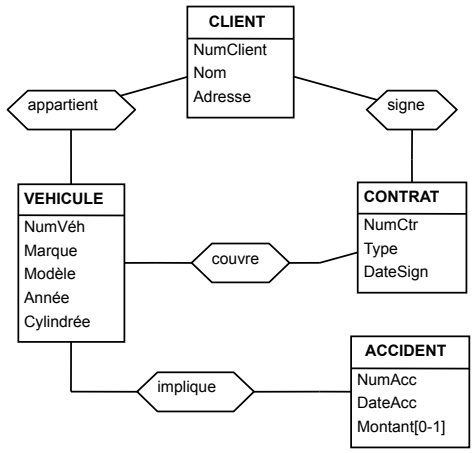
VEHICULE
NumVéh: char (16)
Marque: char (30)
Modèle: char (30)
Année: num (4)
Cylindrée: num (6)

type d'un attribut

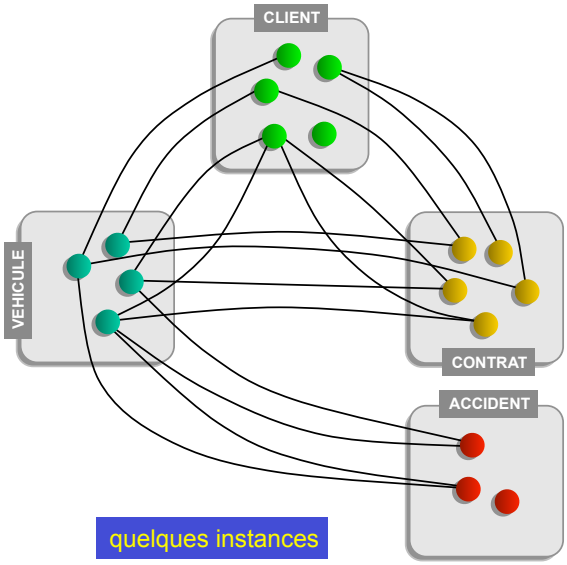
ACCIDENT
NumAcc
DateAcc
Montant[0-1]

attribut obligatoire/facultatif

3.5 Types d'associations



dessin des types



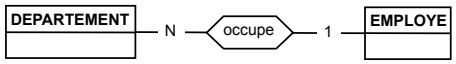
quelques instances

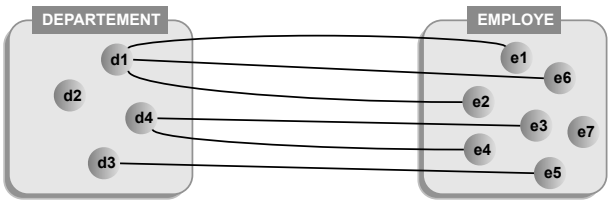
3.5 Types d'associations - Les rôles et leur nom

nom explicite : signe.signataire

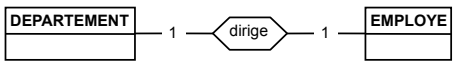
nom implicite : signe.CONTRAT

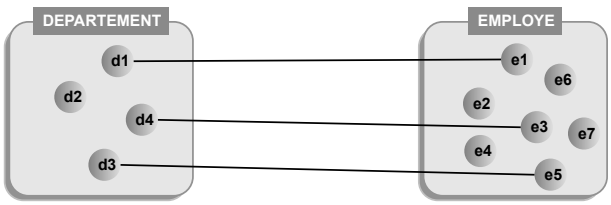
3.5 Types d'associations - Classe fonctionnelle un-à-plusieurs (1:N)



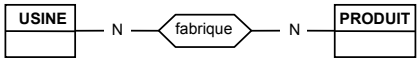


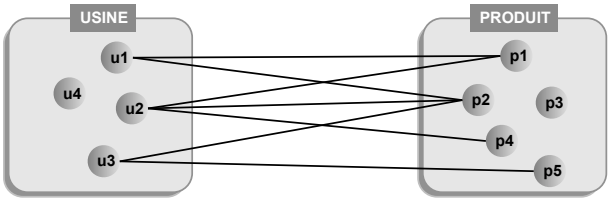
3.5 Types d'associations - Classe fonctionnelle un-à-un (1:1)



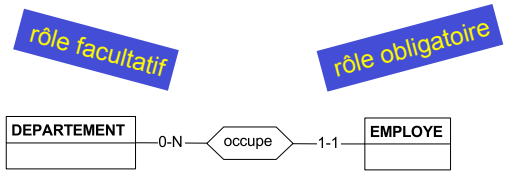


3.5 Types d'associations - Classe fonctionnelle plusieurs-à-plusieurs (N:N)





3.5 Types d'associations - Rôle obligatoire/facultatif - Cardinalité



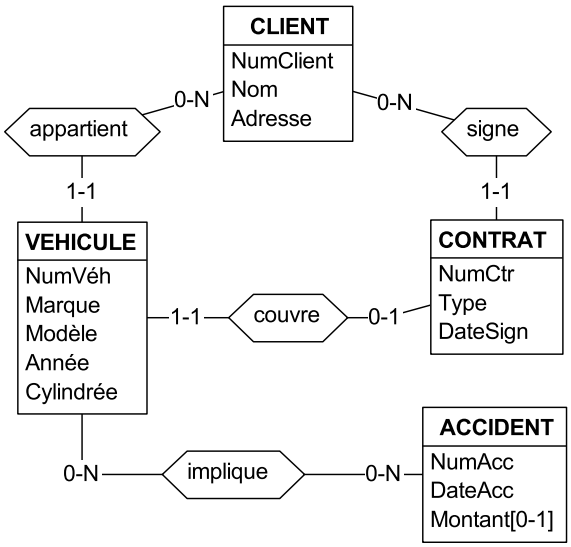
Contrainte de cardinalité

- 1-1
- 0-1
- 0-N

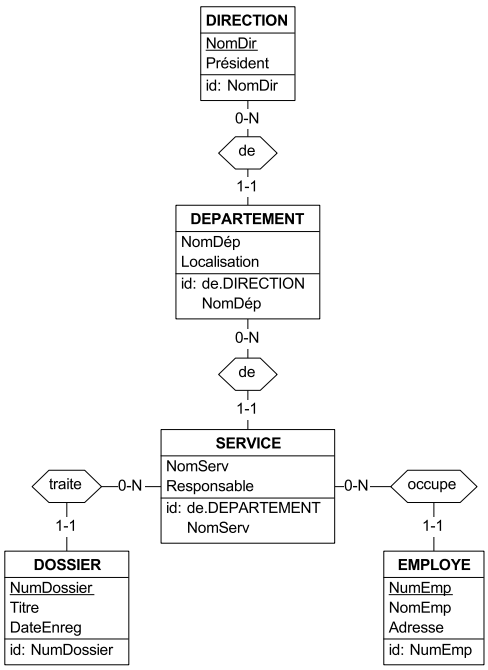
Combinaisons admises

- [0-1] [0-N]
- [1-1] [0-N]
- [0-1] [0-1]
- [1-1] [0-1]

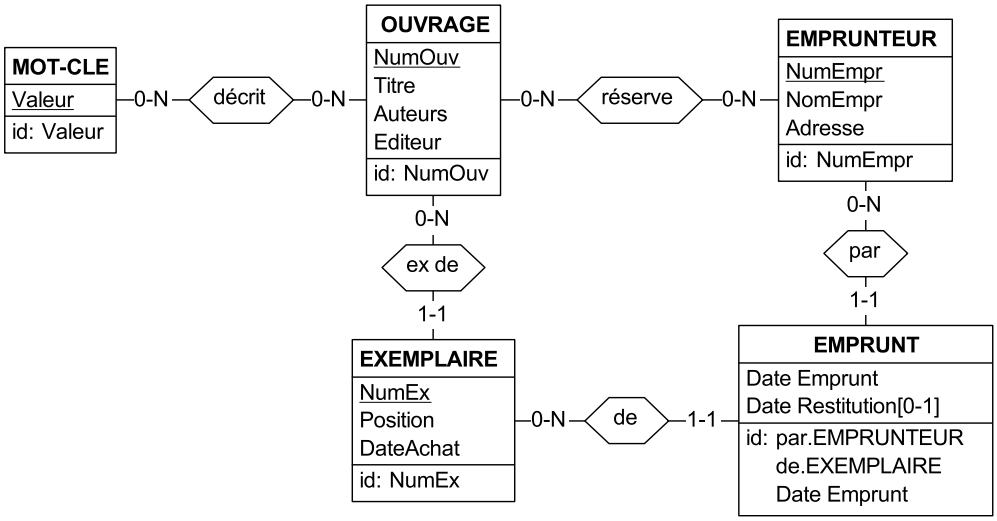
3.5 Types d'associations



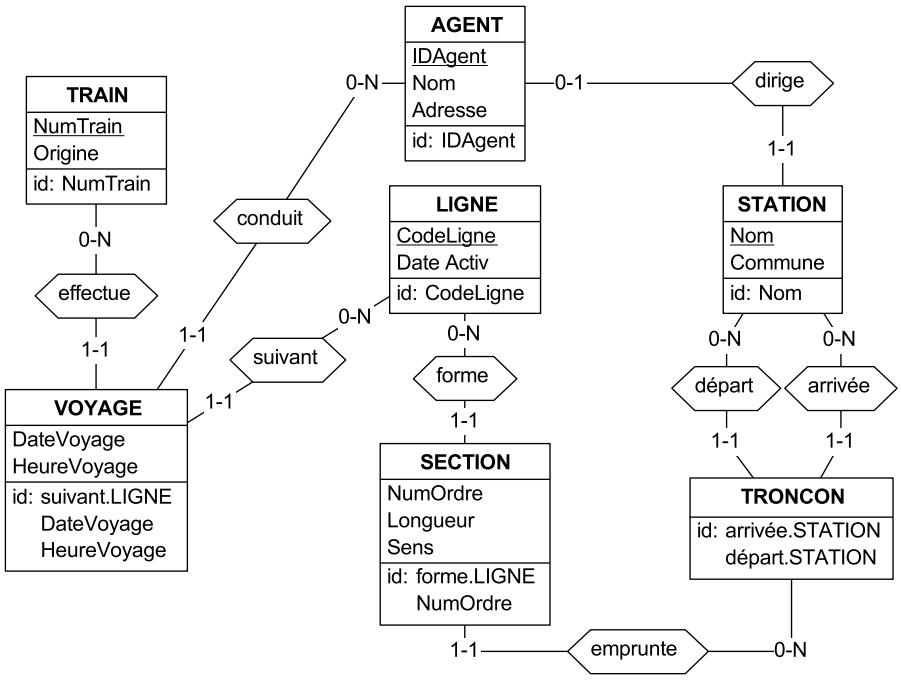
3.6 Exemples - *Structure administrative*



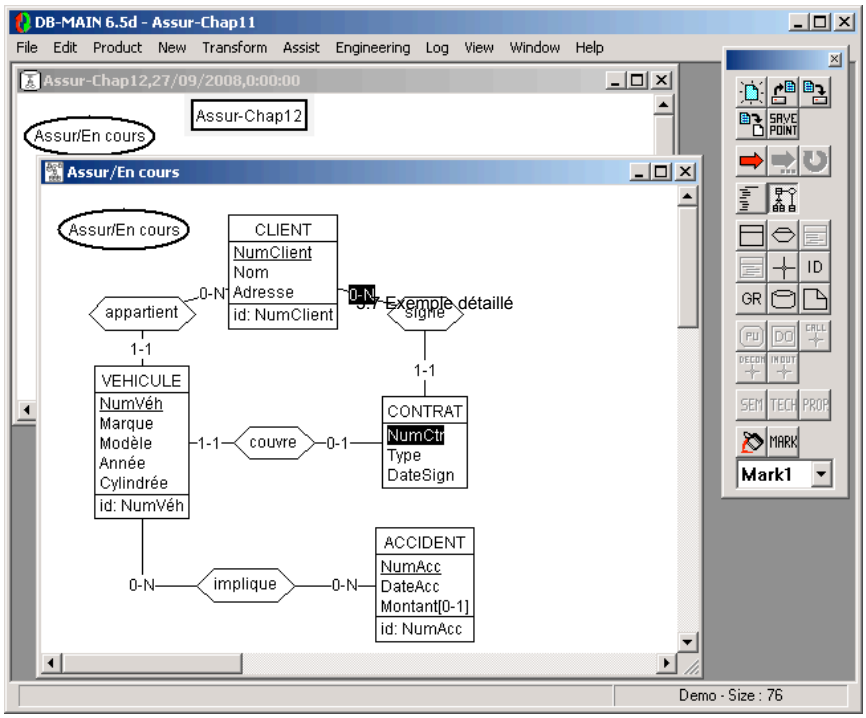
2.8 Exemples - *Gestion d'une bibliothèque*



2.8 Exemples - *Voyages en train*



Complément - Outil de dessin de schémas conceptuels



3.7 Exemple - *Conception d'une BD pour un GIE agricole*

Cahier des charges (simplifié)

- Des agriculteurs possèdent des parcelles
- Une parcelle a un et un seul propriétaire
- Un agriculteur possède une ou plusieurs parcelles
- Pour exploiter ces parcelles, les agriculteurs font appel à un GIE. Celui-ci fournit de la main d'oeuvre d'appoint, à la journée.
- Cette main d'oeuvre est assurée par des employés du GIE.
- Chaque employé du GIE a un tarif, qui constitue son salaire journalier brut.

3.7 Exemple - *Conception d'une BD pour un GIE agricole*

Cahier des charges (simplifié)

- Des **agriculteurs** possèdent des **parcelles**
- Une **parcelle** a *un et un seul* **propriétaire**
- Un **agriculteur** possède *une ou plusieurs* **parcelles**
- Pour exploiter ces parcelles, les agriculteurs font appel à un **GIE**. Celui-ci fournit de la main d'oeuvre d'appoint, à la journée.
- Cette main d'oeuvre est assurée par des **employés du GIE**.
- Chaque **employé du GIE** a un **tarif**, qui constitue son **salaire** journalier brut.

3.7 Exemple - *Conception d'une BD pour un GIE agricole*

Cahier des charges (simplifié) (2)

- Le **GIE** paie ses **employés** mensuellement, en fonction de leurs interventions.
- Chaque **intervention** concerne une **parcelle**, un **employé** et un **nombre de jours**.
- **Le système d'information désiré** doit pouvoir fournir :
  - la **liste des agriculteurs**
  - la **liste des employés**
  - la **liste des interventions par employé**
  - la **liste des interventions par agriculteur**

3.7 Exemple - *Conception d'une BD pour un GIE agricole*

Inventaire des données (simplifié)

- **Agriculteur** : Nom, prénom, lieu de résidence
- **Employé GIE** : Nom *et* prénom
- **Parcelle** : nom, superficie, lieu *et* propriétaire
- **Employé** : numéro insee *et* salaire journalier
- **Interventions** : employé, parcelle, date de début, nombre de jours

Nom	Prénom	Lieu de résidence
Dulhac	Anne-Marie	Arith
Martoz	Christian	Montargy
Carrez	François	Arith
Ferrer	Mariette	Lenoyer
Mernaz	Francine	Lescheraines
Martoz	Christian	Lescheraines

Comment identifier ?

3.7 Exemple - *Conception d'une BD pour un GIE agricole*

Clé primaire

IdAgri	Nom	Prénom	Lieu de résidence
1	Dulhac	Anne-Marie	Arith
2	Martoz	Christian	Montargy
3	Carrez	François	Arith
4	Ferrer	Mariette	Lenoyer
5	Mernaz	Francine	Lescheraines
6	Martoz	Christian	Lescheraines

3.7 Exemple - *Conception d'une BD pour un GIE agricole*

Clé primaire : pas nécessairement une seule colonne

Ville_Dep	Ville_Arr	Distance
Aiglun	Saint-Auban	25
Aix-les-Bains	Chambery	14
Rennes	Paris	342
Rennes	Saint-Malô	75
Saint-Alban	Aiglun	23

Table des distances entre villes

3.7 Exemple - *Conception d'une BD pour un GIE agricole*

Type des attributs (colonnes)

Champ	Type	Extra
Agr_id	int(11)	auto_increment
Agr_Nom	varchar(30)	
Agr_Prn	varchar(20)	
Agr_Resid	varchar(50)	

Agr-Id	Agr_Nom	Agr_Prn	Agr_Resid
1	Dulhac	Anne-Marie	Arith
2	Martoz	Christian	Montargy
3	Carrez	François	Arith
4	Ferrer	Mariette	Lenoyer
5	Mernaz	Francine	Lescheraines
6	Martoz	Christian	Lescheraines

3.7 Exemple - *Conception d'une BD pour un GIE agricole*

Les parcelles

Par_Idf	Par_Nom	Par_Lieu	Par_Superficie
1	Le Pré au Vent	Arith	350
2	Le grand Verger	Arith	300
3	Plan des Bauges	Montargy	220
4	Les Prés Rus	Arith	750
5	Lafosse	Montargy	600

Comment indiquer le propriétaire ?

3.7 Exemple - *Conception d'une BD pour un GIE agricole*

Les parcelles et leur propriétaire

Par_Idf	Par_Nom	Par_Lieu	Par_Superficie	Par_Prop
1	Le Pré au Vent	Arith	350	1
2	Le grand Verger	Arith	300	2
3	Plan des Bauges	Montargy	220	1
4	Les Prés Rus	Arith	750	**4**
5	Lafosse	Montargy	600	1

Parcelles

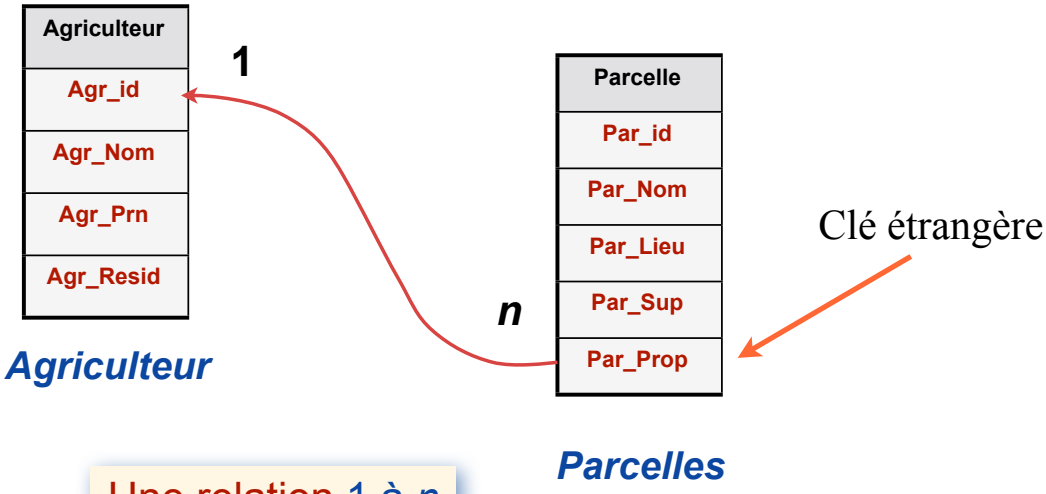
Agriculteur

Agr_Id	Agr_Nom	Agr_Prn	Agr_Resid
1	Dulhac	Anne-Marie	Arith
2	Martoz	Christian	Montargy
3	Carrez	François	Arith
4	Ferrer	Mariette	Lenoyer
5	Mernaz	Francine	Lescheraines
6	Martoz	Christian	Lescheraines

Une relation entre deux tables

3.7 Exemple - *Conception d'une BD pour un GIE agricole*

Les parcelles et leur propriétaire



Une relation 1 à *n* entre deux tables

3.7 Exemple - *Conception d'une BD pour un GIE agricole*

La table *employé*

Emp-nss	Emp_Nom	Emp_prn	Emp_tarif
1,75077E+12	Grandet	Marc	110
1,82023E+12	Barnier	Nicole	115
1,79011E+12	Pernet	Henri	119

Quelle clé primaire ?

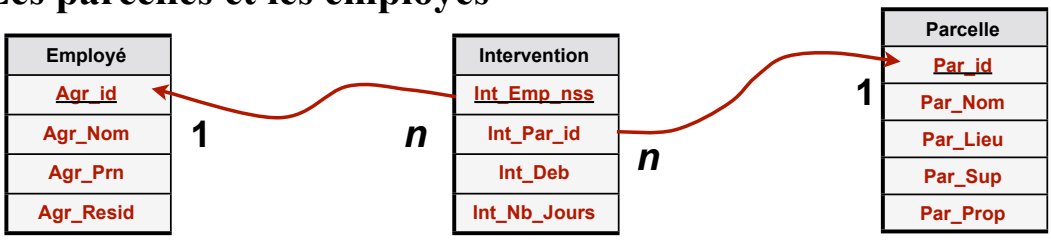
3.7 Exemple - *Conception d'une BD pour un GIE agricole*

Quelle relation entre les tables ***parcelle*** et ***employé*** ?

- **Un employé** peut être amené à travailler sur **plusieurs parcelles**
- Sur **une même parcelle** peuvent travailler **plusieurs employés** (simultanément ou l'un après l'autre)
- C'est une relation **plusieurs à plusieurs :** on la représente par **une nouvelle table**

3.7 Exemple - *Conception d'une BD pour un GIE agricole*

Les parcelles et les employés



L'employé Grandet a travaillé sur la parcelle du pré au vent, 5 jours à partir du 5 juillet 2004.

Intervention

Int_Emp_nss	Int_Par_id	Int_Deb	Int_Nb_Jours
1,75077E+12	1	2004-07-05	5

Emp_nss	Emp_Nom	Emp_prn	Emp_tarif
1,75077E+12	Grandet	Marc	110

Employé

Parcelle

Par_Idf	Par_Nom	Par_Lieu	Par_Superficie	Par_Prop
1	Le Pré au Vent	Arith	350	1

4. LE LANGAGE SQL

(*Structured Query Language*)

Version 1 - Janvier 2010

4.1 Introduction à SQL : LDD et LMD

SQL n'est pas un langage de programmation complet

- SQL permet :
  - de **définir le schéma** de la base de données (LDD)
  - de **charger** les tables relationnelles (LMD)
  - de **manipuler** les **données** stockées (LMD)
  - de **gérer** la base de données (LDD) : **sécurité**, **organisation physique**

Ici : aperçu de la partie LDD

Plus loin : le LMD

1. LE LANGAGE SQL LDD

Contenu

4.1 Introduction

Le sous-langage LDD de SQL permet de créer des structures de données et de les modifier.

Les opérations du LDD :

- **Création** d'un **schéma**
- **Création** d'une **table**
  - colonnes (obligatoire)
  - domaine
  - identifiants primaire et secondaire
  - clé étrangère
- **Suppression** d'une **table**
- **Ajout**, **suppression**, **modification** d'une **colonne**
- **Ajout**, **suppression** d'une **contrainte**
- **Ajout**, **suppression** d'un **index**
- **Création** d'un **espace de stockage**

1.3 CREATION D'UNE TABLE

Contenu

a) Tables, colonnes et domaines

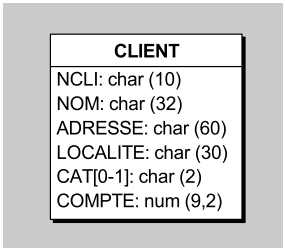
b) Les identifiants

c) Les clés étrangères

d) Suppression d'une table

4.2 Création d'une table - Tables, colonnes et domaines

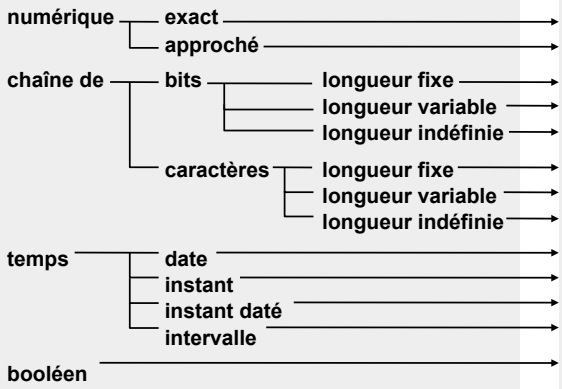
Création d'une table et de ses colonnes




```
create table CLIENT ( NCLI      char(10),
                      NOM       char(32),
                      ADRESSE   char(60),
                      LOCALITE  char(30),
                      CAT       char(2),
                      COMPTE    decimal(9,2) );
```

4.2 Création d'une table - Tables, colonnes et domaines

Les types de base



4.2 Création d'une table - Tables, colonnes et domaines

Les colonnes obligatoires

Une colonne est facultative par défaut.
Il faut déclarer explicitement les colonnes obligatoires

```
create table CLIENT ( NCLI     char(10) not null,
                      NOM      char(32) not null,
                      ADRESSE  char(60) not null,
                      LOCALITE char(30) not null,
                      CAT      char(2),
                      COMPTE   decimal(9,2) not null
                    );
```

4.2 Création d'une table - Tables, colonnes et domaines

Valeur par défaut d'une colonne

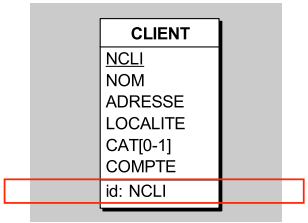
Sera assignée à la colonne si on ne spécifie pas de valeur lors de la création d'une ligne

```
create table CLIENT (
              NCLI     char(10) not null,
              NOM      char(32) not null,
              ADRESSE  char(60) not null,
              LOCALITE char(30) not null default 'Paris',
              CAT      char(2) default 'B1',
              COMPTE   decimal(9,2) not null default 0.0
                      );
```

```
create domain MONTANT decimal(9,2) default 0.0;
```


4.2 Création d'une table - Les identifiants

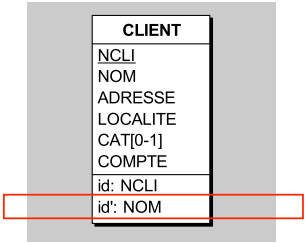
Les identifiants primaires (*primary key*)




```
create table CLIENT ( NCLI     char(10) not null,
                      NOM      char(32) not null,
                      ADRESSE  char(60) not null,
                      LOCALITE char(30) not null,
                      CAT      char(2),
                      COMPTE   decimal(9,2) not null,
                      primary key (NCLI) );
```

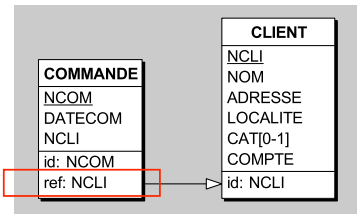
4.2 Création d'une table - Les identifiants

Les identifiants secondaires (*candidate key*)




```
create table CLIENT ( NCLI     char(10) not null,
                      NOM      char(32) not null,
                      ADRESSE  char(60) not null,
                      LOCALITE char(30) not null,
                      CAT      char(2),
                      COMPTE   decimal(9,2) not null,
                      primary key (NCLI),
                      unique (NOM) );
```


4.2 Création d'une table - Les clés étrangères ("foreign key")




```
create table COMMANDE (NCOM     char(12) not null,
                       NCLI     char(10) not null,
                       DATECOM  date not null,
              primary key (NCOM),
              foreign key (NCLI) references CLIENT);
```

Si l'identifiant cible est secondaire (*possible mais déconseillé*) :

```
              foreign key (NOM) references CLIENT(NOM)
```

4.2 Création d'une table - Les clés étrangères

Variante : contraintes de colonne

```
create table CLIENT (
             NCLI char(10) not null primary key,
             . . .
                    );
```

```
create table CLIENT (
             NCLI char(10) not null,
             NOM  char(32) not null unique,
             . . .
                    );
```

```
create table COMMANDE (
             NCOM  char(12) not null,
             NCLI  char(10) not null references CLIENT,
             . . . );
```


4.2 Création d'une table - Suppression d'une table

Suppression d'une table

```
drop table COMMANDE;
```

Attention, opération sous haute surveillance !

- la table ne doit plus être référencée par une clé étrangère;
- (E.g. la table DETAIL (ou sa clé étrangère vers COMMANDE) doit avoir été supprimée)

4.2 MODIFICATION D'UNE TABLE

Contenu

a) **Ajout, retrait** et **modification d'une colonne**

b) **Ajout** et **retrait d'une contrainte**

4.2 Modification d'une table - *Colonnes*

Ajout, retrait et modification d'une colonne

```
alter table PRODUIT add column POIDS smallint;
```
ajouter

```
alter table PRODUIT drop column PRIX;
```
supprimer

```
alter table CLIENT modify column CAT set '00';
```
modifier valeur par défaut

```
alter table CLIENT modify column CAT drop default;
```
supprimer valeur par défaut

4.2 Modification d'une table - *Contraintes*

Ajout et retrait d'une contrainte

```
alter table PROSPECT add primary key (NCLI);
```

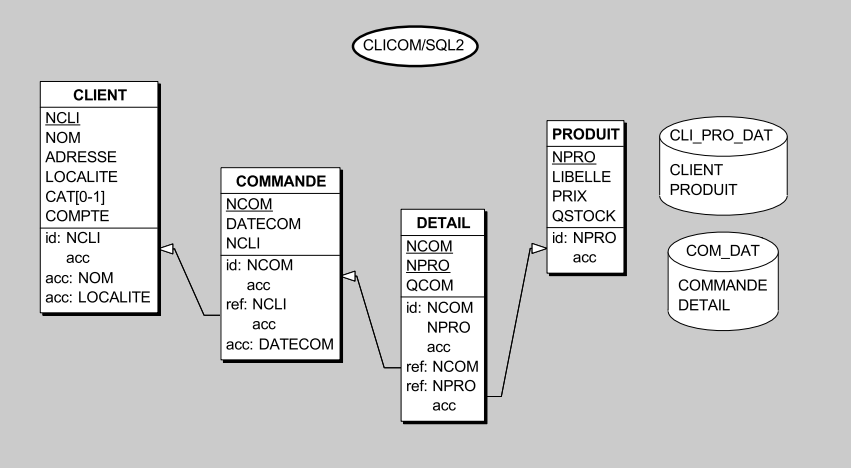
```
alter table CLIENT add unique (NOM,ADRESSE,LOCALITE);
```

```
alter table CLIENT modify CAT not null;
alter table CLIENT modify ADRESSE null;
```

```
alter table CLIENT
    add foreign key (CAT) references CATEGORIE ;
```


4.2 Exemple - Le schéma

Traduire ce schéma en SQL



4.2 Exemple - Le schéma et les espaces de stockage

```
create schema CLICOM;


create dbspace CLI_PRO_DAT;

create dbspace COM_DAT;
```


4.2 Exemple - Les tables

```
create table CLIENT (  NCLI     char(10) not null,
                       NOM      char(32) not null,
                       ADRESSE  char(60) not null,
                       LOCALITE char(30) not null,
                       CAT      char(2),
                       COMPTE   decimal(9,2) not null,
                    primary key (NCLI) ) in CLI_PRO_DAT;

create table PRODUIT ( NPRO     char(15) not null,
                       LIBELLE  char(60) not null,
                       PRIX     decimal(6) not null,
                       QSTOCK   decimal(8) not null,
                    primary key (NPRO) ) in CLI_PRO_DAT;

create table COMMANDE (NCOM     char(12) not null,
                       NCLI     char(10) not null,
                       DATECOM  date not null,
                    primary key (NCOM),
                    foreign key (NCLI) references CLIENT) in COM_DAT;

create table DETAIL (  NCOM     char(12) not null,
                       NPRO     char(15) not null,
                       QCOM    decimal(8) not null,
                    primary key (NCOM,NPRO),
                    foreign key (NCOM) references COMMANDE,
                    foreign key (NPRO) references PRODUIT) in COM_DAT;
```

2. LE LANGAGE SQL LMD (1)

Contenu

4.3 Introduction

Le sous-langage LMD de SQL permet de **consulter le contenu des tables** et de les **modifier.** Il comporte **4 verbes :**

- La requête **select** **extrait** des données des tables
- La requête **insert** **insère** de nouvelles lignes dans une table
- La requête **delete** **supprime** des lignes d'une table
- La requête **update** **modifie les valeurs** de colonnes de lignes existantes

4.3 Extraction simple - projection

Personnes

nom	*prénom*	*adresse*	*téléphone*
Martin	Pierre	7 allée des vers	258941236
Dupond	Jean	32 allé Poivrot	526389152
Dupond	Marc	8 rue de l'octet	123456789

```
SELECT nom, prénom
FROM   Personnes
```

On projette la table ***Personnes*** sur les colonnes *nom* et *prénom*.

nom	*prénom*
Martin	Pierre
Dupond	Jean
Dupond	Marc

4.3 Extraction simple - sélection

Personnes

nom	*prénom*	*adresse*	*téléphone*
Martin	Pierre	7 allée des vers	258941236
Dupond	Jean	32 allé Poivrot	526389152
Dupond	Marc	8 rue de l'octet	123456789

```
SELECT *
FROM   Personnes
WHERE nom = ''Dupond''
```

On ne sélectionne que les tuples dont l'attribut *nom* est égale à 'Dupond'.

nom	*prénom*	*adresse*	*téléphone*
Dupond	Jean	32 allé Poivrot	526389152
Dupond	Marc	8 rue de l'octet	123456789

4.3 Extraction simple - sélection

Relation de départ :

SELECT * FROM *Gens*

Gens		
Nom	*Prenom*	*Age*
Dupond	Pierre	24
Martin	Marc	48
Dupont	Jean	51
Martin	Paul	36
Dupond	Lionel	68
Chirac	Jacques	70

1

SELECT *Nom*
FROM *Gens*

Gens
Nom
Dupond
Martin
Dupont
Martin
Dupond
Chirac

2

SELECT DISTINCT *Nom*
FROM *Gens*

3

Gens
Nom
Dupond
Martin
Dupont
Chirac

4.3 Extraction simple - sélection

Gens
Nom
Chirac
Dupond
Dupont
Martin

SELECT DISTINCT *Nom*
FROM *Gens*
ORDER BY *Nom* **ASC**

4

5

Gens
Nom
Chirac
Dupond

SELECT DISTINCT *Nom*
FROM *Gens*
ORDER BY *Nom* **ASC**
LIMIT 2

6

Gens
Nom
Dupond

SELECT DISTINCT *Nom*
FROM *Gens*
WHERE *Nom* **<> 'Chirac'**
ORDER BY *Nom* **ASC**
LIMIT 2

4.4 Extraction complexe - jointure

Personnes

nom	***prénom***	*adresse*	*téléphone*
Martin	Pierre	7 allée des vers	258941236
Dupond	Jean	32 allé Poivrot	526389152

Bibliothèque

nom	***Dernierlivre***
Dupond	Robinson
Jospin	Faust
Martin	Misère

```
SELECT Personnes.prénom, dernierlivre
FROM   Personnes, Bibliothèque
WHERE Personnes.nom = Bibliothèque.nom
```

prénom	*Dernierlivre*
Jean	Robinson
Pierre	Misère

On joint les deux tables, grâce à la colonne *nom.*

Et on combine cette jointure à une projection sur les attributs *nom* et *dernierlivre*.

Attention à lever toute ambiguïté sur les noms d'attribut dans le cas où deux tables possèdent des colonnes de même nom.

4.4 Conditions plus complexes - les valeurs *null*

```
select NCLI
from   CLIENT
where  CAT = null;
```

null **ne peut être comparé à rien, même pas à lui-même !**

NCLI

```
select NCLI
from   CLIENT
where  CAT is null;
```

NCLI
D063
K729

```
select NCLI
from   CLIENT
where  CAT is not null;
```

4.4 Conditions plus complexes - *exemples*

```
select nom
from   produit
where  prix <= 100.5;
```

Liste des noms de produits dont le prix est inférieur ou égal à 100.5 €

```
select nom, prénom
from   ELEVES
where  age between 12 and 16;
```

Liste des noms et prénoms des élèves dont l'âge est dans [12,16]

```
select modèle
from   VOITURES
where  couleur in ('bleu', 'blanc', 'noir');
```

Liste des modèles de voitures dont la couleur est dans la liste : *bleu, blanc, noir*.

```
select modèle
from   VOITURES
where  couleur not in ('rose', 'violet');
```

Liste des modèles de voitures dont la couleur n'est pas dans la liste : *rose, violet*.

4.4 Conditions plus complexes - Les masques

```
select NCLI
from   CLIENT
where  CAT like 'B_';
```

'_' = un caractère quelconque

masques

```
select NPRO
from   PRODUIT
where  LIBELLE like '%SAPIN%';
```

'%' = une chaîne quelconque

Un masque définit une famille de chaînes de caractères :

'B_'	⟶	'B1' 'Bd' 'B '
'B_'	⟶	'xB' 'B' 'B12'
'%SAPIN%'	⟶	'PL. SAPIN 200x20x2' 'Boite en SAPIN' 'SAPIN VERNI'
'%SAPIN%'	⟶	'Boite en Sapin' 'Achetez S A P I N !'

4.4 Conditions plus complexes - Combinaisons logiques

```
select NOM, ADRESSE, COMPTE
from   CLIENT
where  LOCALITE = 'Toulouse' and COMPTE < 0;
```

```
select NOM, ADRESSE, COMPTE
where  COMPTE > 0
       and (CAT = 'C1' or LOCALITE = 'Paris')
```


4.4 Données extraites et données dérivées - expressions de calcul

```
select 'TVA de ', NPRO, ' = ',0.21*PRIX*QSTOCK
from   PRODUIT
where  QSTOCK > 500;
```

TVA de	NPRO	=	0,21*PRIX*QSTOCK
TVA de	CS264	=	67788
TVA de	PA45	=	12789
TVA de	PH222	=	37770.6
TVA de	PS222	=	47397

```
select NPRO as Produit, 0.21*PRIX*QSTOCK as Valeur_TVA
from   PRODUIT
where  QSTOCK > 500;
```

"Produit" est un alias de colonne

Produit	Valeur_TVA
CS264	67788
PA45	12789
PH222	37770.6
PS222	47397

4.5 Les fonctions agrégatives (statistiques)

```
select 'Namur',avg(COMPTE) as Moyenne,
       max(COMPTE)-min(COMPTE) as Ecart_max,
       count(*) as Nombre
from   CLIENT
where  LOCALITE = 'Namur';
```

Namur	Moyenne	Ecart_max	Nombre
Namur	-2520	4580	4

le résultat ne comprend qu'une seule ligne

```
select sum(QSTOCK*PRIX)
from   PRODUIT
where  LIBELLE like '%SAPIN%';
```


4.5 Les fonctions agrégatives (statistiques)

```
select count(NCLI) as Nombre,
       count(NOM) as Noms,
       count(LOCALITE) as Localités,
       count(CAT) as Catégories
from   CLIENT;
```

Nombre	Noms	Localités	Catégories
16	16	16	14

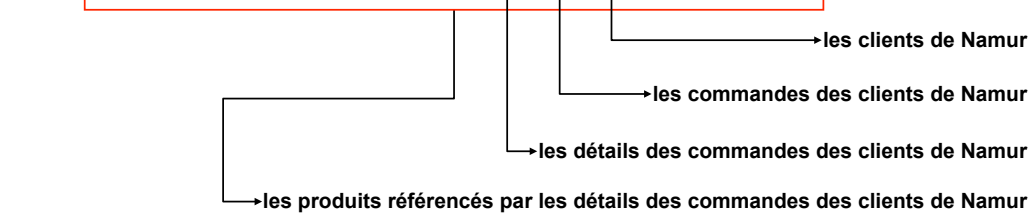
Si catégories a 2 valeurs nulles de plus que les autres attributs

```
select count(distinct NCLI) as Nombre
       count(distinct NOM) as Noms,
       count(distinct LOCALITE) as Localités,
       count(distinct CAT) as Catégories
from   CLIENT;
```

Nombre	Noms	Localités	Catégories
16	15	7	4

4.6 Les sous-requêtes - Principe

```
select *
from   PRODUIT
where  NPRO in
       (select NPRO
        from   DETAIL
        where  NCOM in
               (select NCOM
                from   COMMANDE
                where  NCLI in
                       (select NCLI
                        from   CLIENT
                        where  LOCALITE='Namur')));
```



4.6 Les sous-requêtes - Condition d'association

Une condition *in (sous-requête)* correspond le plus souvent à une condition d'association = *qui sont associés à ...*

```
select *
from   T
where  CT in (select CS
              from   S
              where  <condition>);
```

"on recherche les T qui sont associés à des S qui ..."

4.6 Les sous-requêtes - Condition d'association

Remarque : symétrie des conditions d'association

```
select *
from   COMMANDE
where  NCLI in (select NCLI
                from   CLIENT
                where  LOCALITE = 'Namur');
```

```
select *
from   CLIENT
where  NCLI in (select NCLI
                from   COMMANDE
                where  DATECOM = '12-09-2009');
```


5 - PHP / MySQL

5.1 MySQL

MySQL est un **Système de Gestion de Bases de Données**

- Se charge du **stockage**, de **l'accès**, de la **recherche**, de la **sécurité** ... des données
- Utilise le langage **SQL**
- Système **libre** (disponible sous Mac OS X, Linux, Windows, ...)
- L'un des **plus utilisé** au monde (e.g. Wikipedia, Google, YouTube, ...)
- **Multi-utilisateur**
- Utilisable depuis une grande variété de **langages de programmation** (C, C++, Java, Python, PHP, ...)
- **...**

5.3 Administration avec phpMyAdmin

- phpMyAdmin est une **application PHP** développée pour administrer une base MySQL **à distance via un navigateur web**
- Application intégrée à EasyPHP
- phpMyAdmin permet également de **visualiser les requêtes SQL**

5.3 Administration avec phpMyAdmin

Interface intuitive pour l'administration des bases de données du serveur

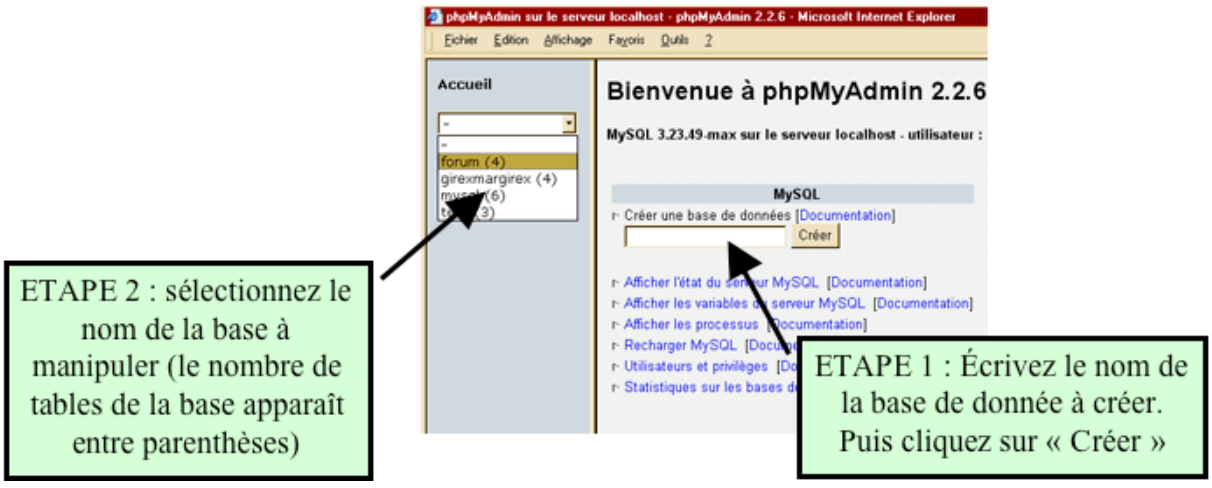
Outil pour :

- **créer** de nouvelles **bases**
- **créer** / **modifier** / **supprimer** des **tables**
- **afficher** / **ajouter** / **modifier** / **supprimer** des **tuples** dans des tables
- effectuer des **sauvegarde** de la structure et/ou des données
- effectuer n'importe quelle **requête**
- gérer les **privilèges** des utilisateurs

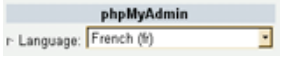
Concrètement, **PhpMyAdmin est un ensemble de pages PHP**. Ce n'est pas un programme, mais des pages PHP toutes prêtes dont on se sert pour gagner du temps.

5.3 Administration avec phpMyAdmin : *Création / sélection d'une base de données*

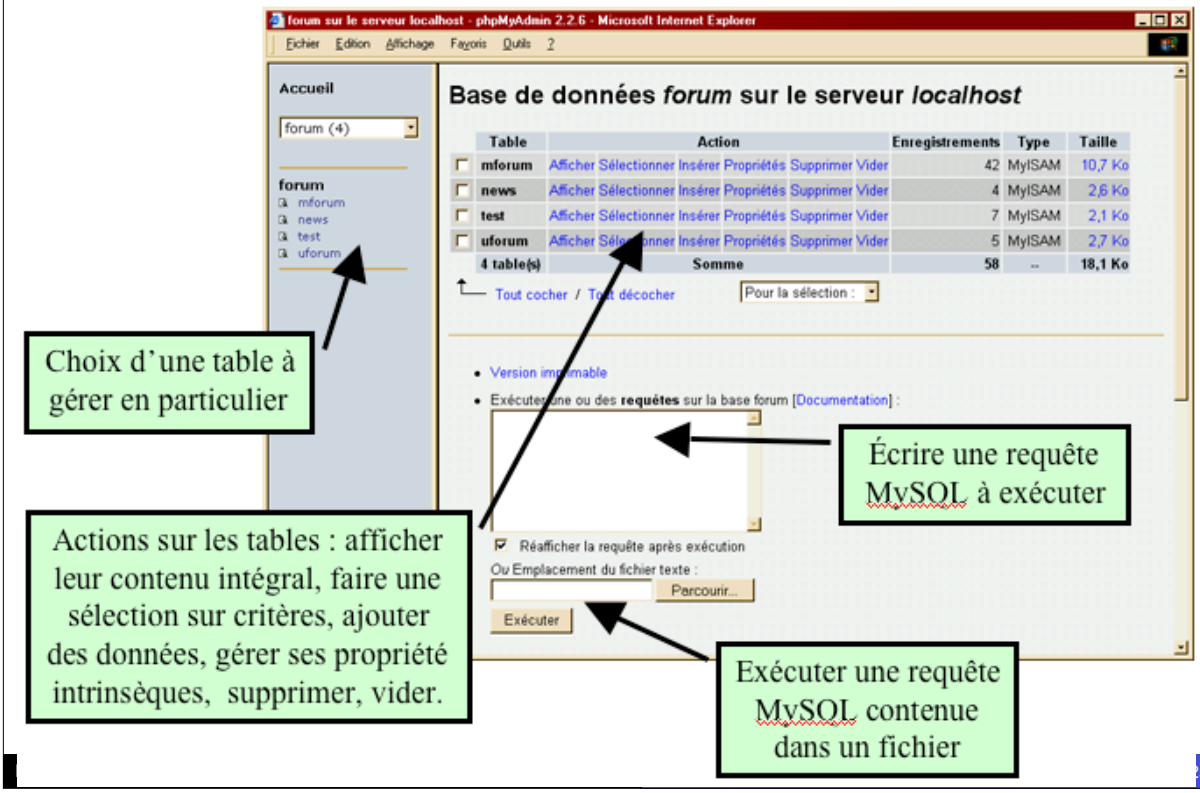
Création / sélection d'une base de données



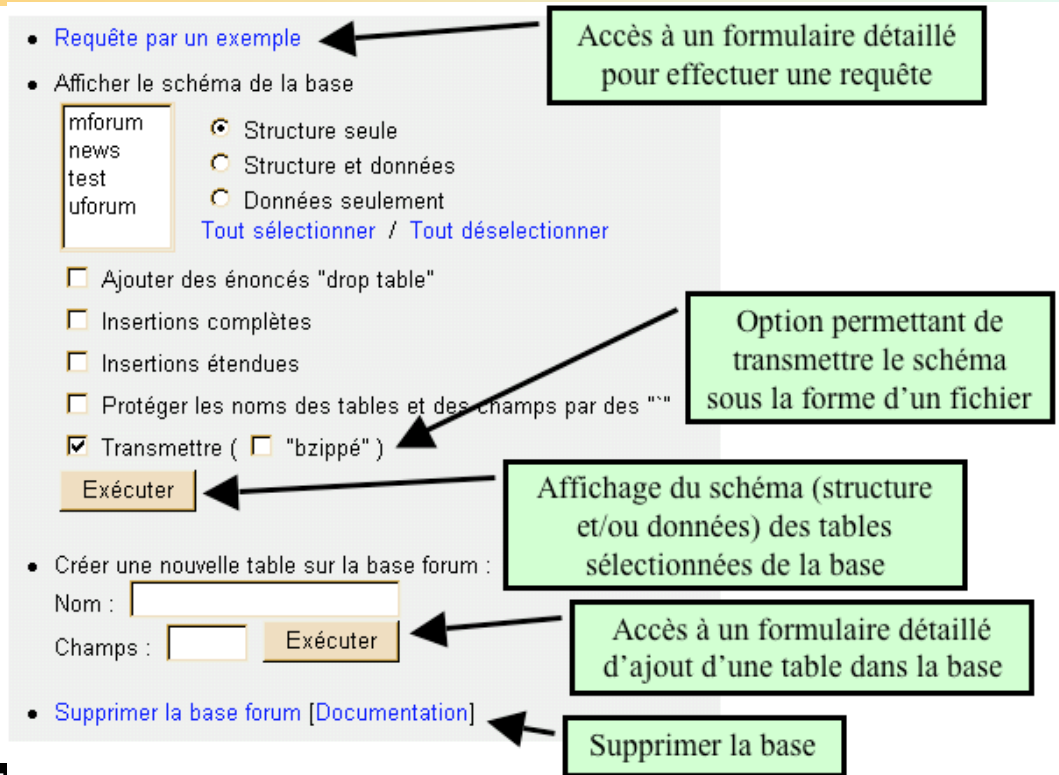
Et aussi.. choix de la langue de l'interface de phpMyAdmin



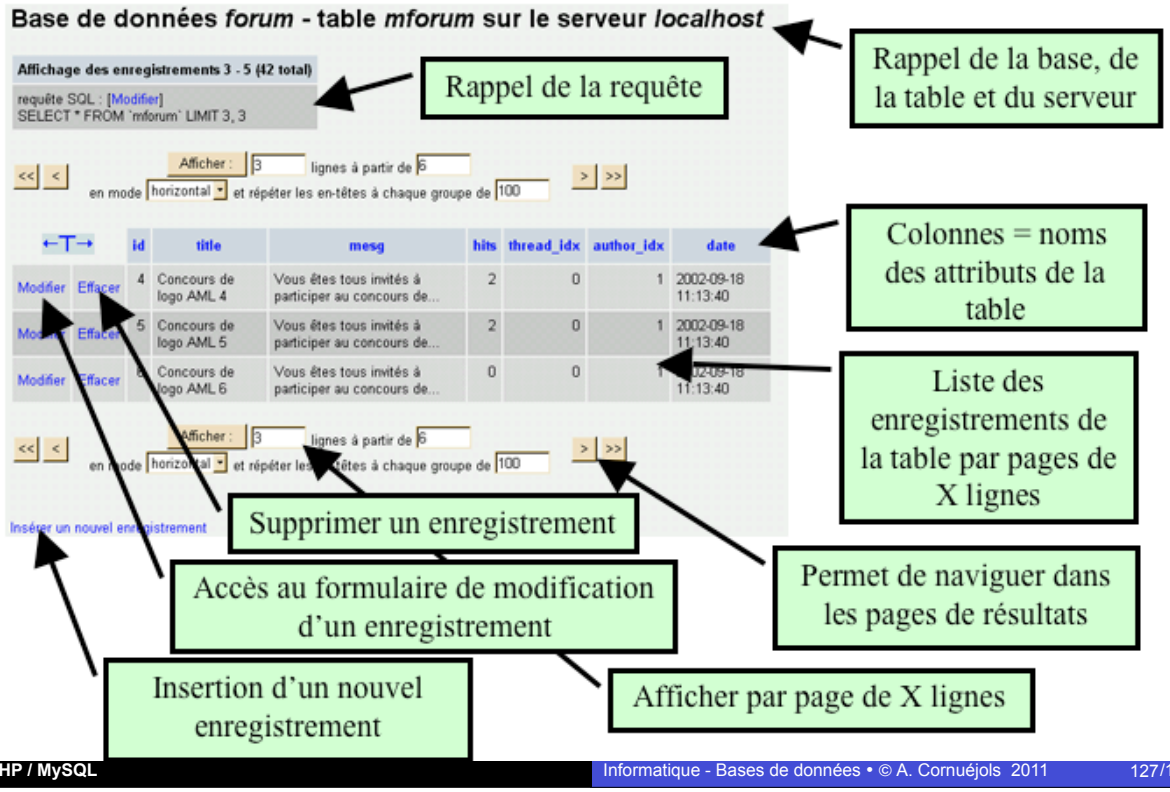
5.3 Administration avec phpMyAdmin : *Gestion de la base de données*



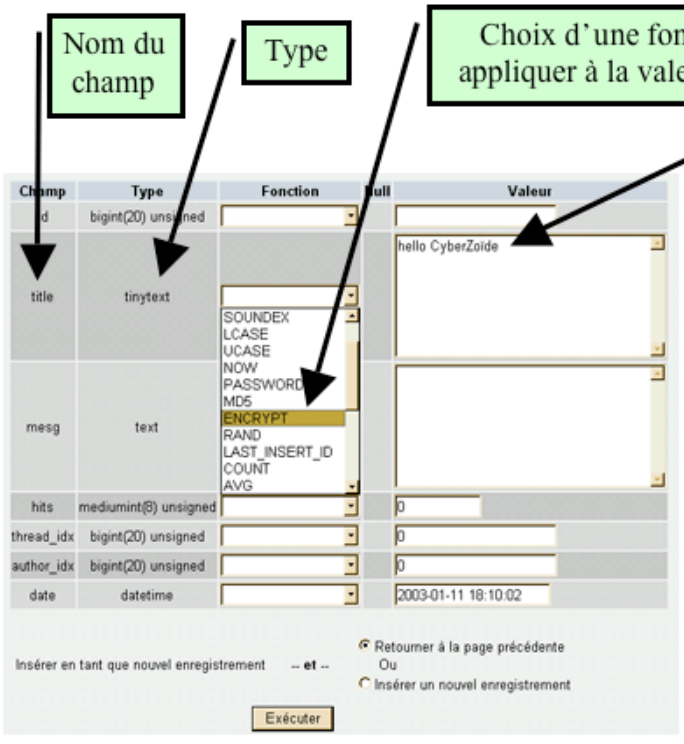
5.3 Administration avec phpMyAdmin : *Gestion de la base de données*



5.3 Administration avec phpMyAdmin : *Affichage d'une table*



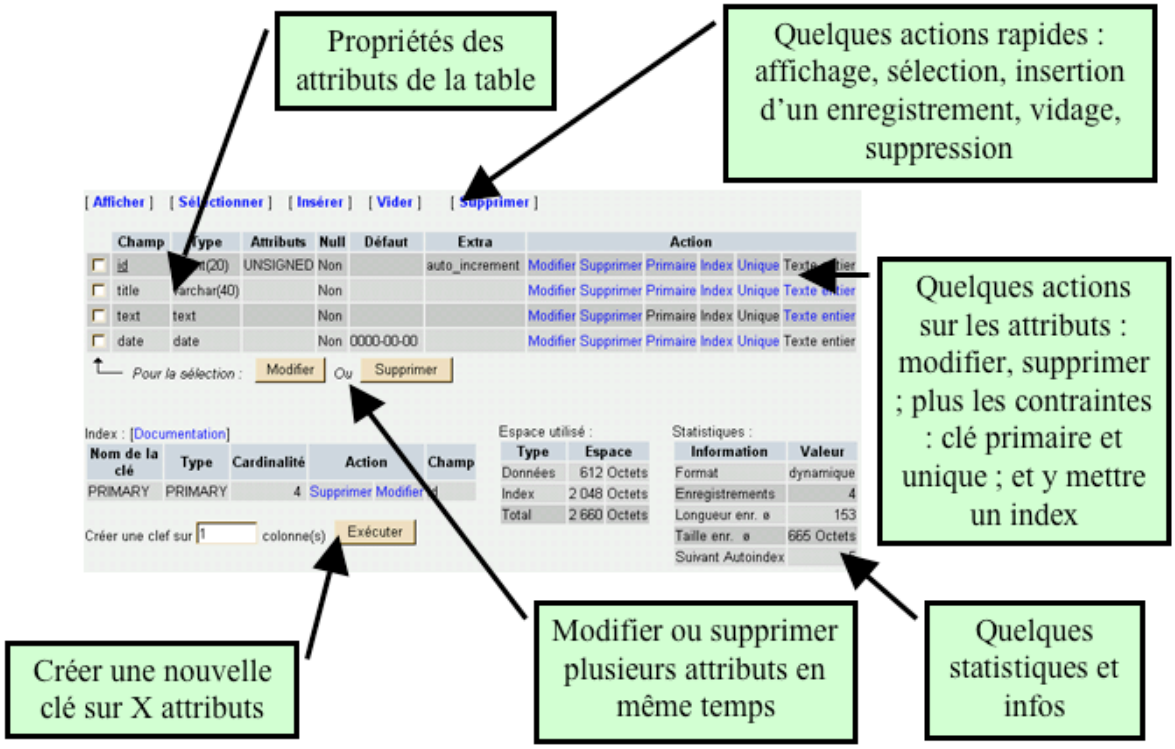
5.3 Administration avec phpMyAdmin : *Insertion / modification d'une ligne*



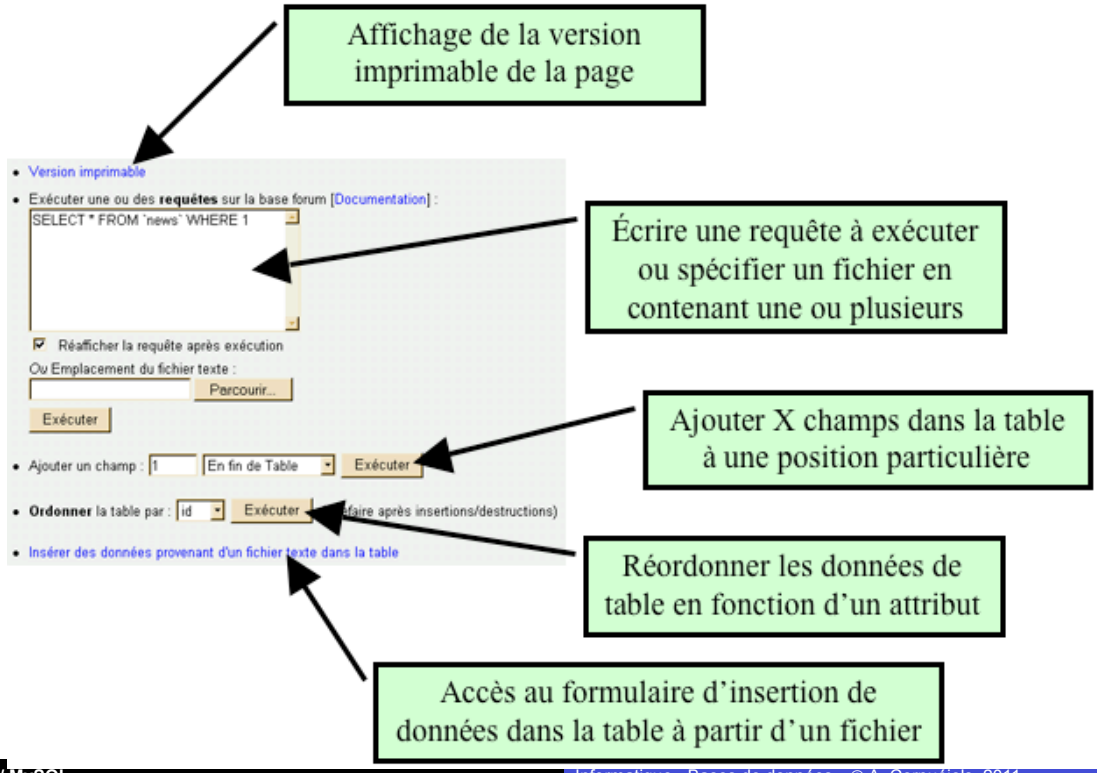
Les champs et leurs types sont définis lors de la création de la table : tous les champs sont pas forcément obligatoires…

Les formulaires d'insertion et de modification sont similaires.

5.3 Administration avec phpMyAdmin : *Gestion d'une table*

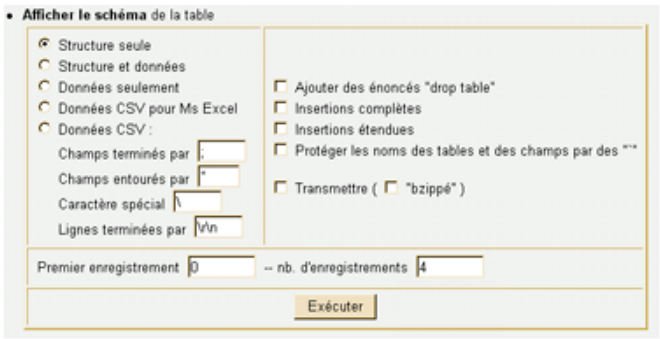


5.3 Administration avec phpMyAdmin : *Gestion d'une table*



5.3 Administration avec phpMyAdmin : *Affichage de la structure d'une table*

- **Affiche le schéma** de la table :
- **Structure** et/ou **données**



Structure :

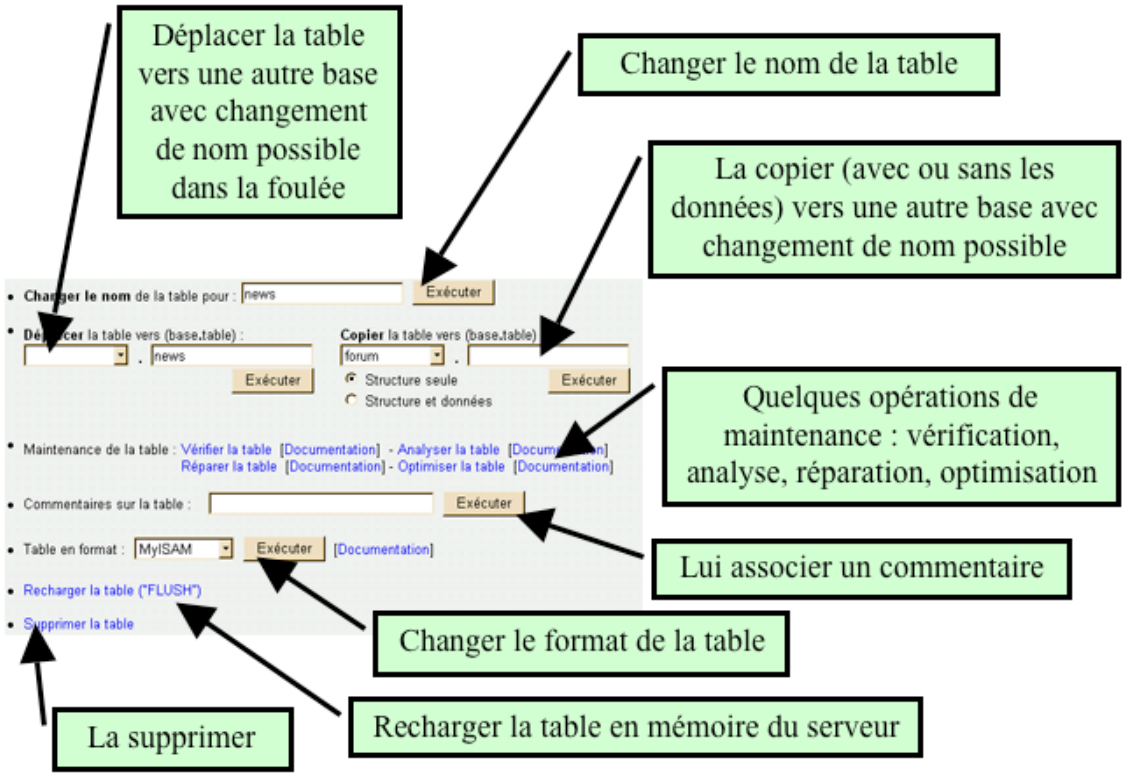
- définition des **propriétés des attributs**
- **clés**

Sortie = requêtes SQL de

- **création** de la **table**
- et d'**insertion** des **enregistrements**

Le **format CSV** est un fichier texte dont chaque ligne représente un enregistrement

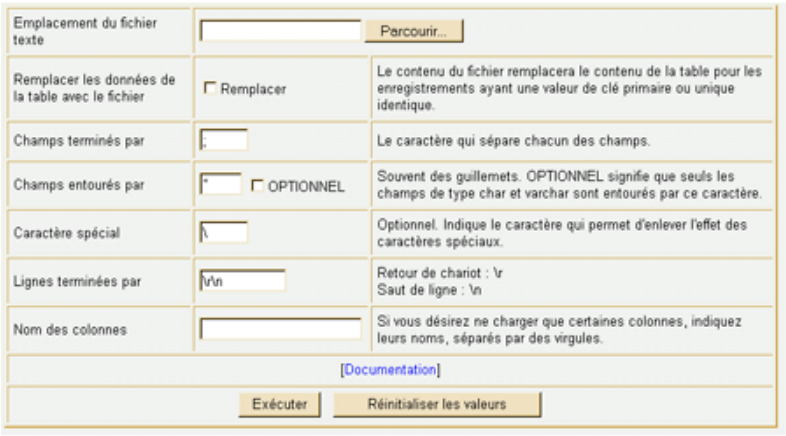
5.3 Administration avec phpMyAdmin : *Gestion d'une table*



5.3 Administration avec phpMyAdmin : *Insertion de données dans une table*

- **Accès à cet écran** par le lien "*Insérer des donnés provenant d'un fichier texte dans la table*" de la page de gestion de la table.

- Permet d'**insérer des enregistrements** dans une table à partir d'un fichier de données au format CSV.



On peut changer les valeurs par défaut des séparateurs standards du CSV.

5.3 Administration avec phpMyAdmin : *Insertion de données dans une table*

- **Accès à cet écran** par le lien "*Créer une clé sur X colonnes*" de la page de gestion de la table.

- Permet de **créer une clé** sur une ou plusieurs colonnes.

- Il faut **nommer la clé**, en **spécifier le type**, et les **attributs** sur lesquels elle s'applique.
- On peut **ajouter une autre colonne** à cette clé avant de valider l'ajout de la clé.

Conclusions

- **Base de données** = ensemble de données **structurées**, **persistantes** placé dans des **fichiers**
- **Gestion par SGBD** (*Système de Gestion de Bases de Données*)
  - **Stockage ; accès ; recherche ; sécurité ...**
  - Exemples : **MySQL, ORACLE, PostgreSQL, ...**
- **Langage spécialisé : SQL**
- On choisit une **application cliente** permettant de **dialoguer avec** un **serveur MySQL :** PHP ; phpMyAdmin ; ...

Conclusions : votre rôle

- À partir d'un **cahier des charges :**
  - **Modéliser le "monde"** (Entités / Relations)
  - **Traduire** en données et relations : **tables**
- **Choisir et mettre en oeuvre un SGBD** (*Système de Gestion de Bases de Données*)
- **Administrer** la base de données
  - **Gérer les utilisateurs ; les ressources**
  - **Entretenir la base de données**

5.4 MySQL et PHP

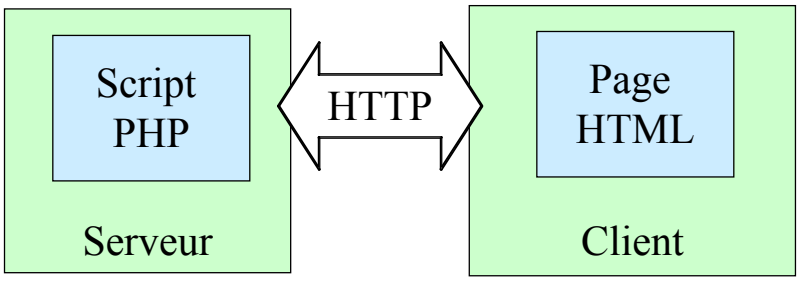
Utilisation de MySQL sous PHP

5.4 MySQL et PHP

MySQL dans l'environnement client-serveur web

- MySQL est souvent utilisé dans le cadre d'un **serveur web**
- avec le langage **PHP**

Jusqu'ici :

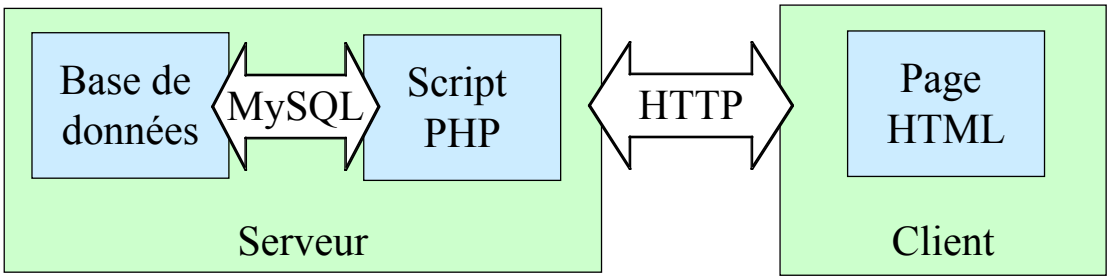


5.4 MySQL et PHP

MySQL dans l'environnement client-serveur web

- MySQL est souvent utilisé dans le cadre d'un **serveur web**
- avec le langage **PHP**

Désormais :



5.4 Interface avec PHP - *connexion*

- Pour se connecter à une base depuis un script PHP
  - spécifier un **nom de serveur**
  - un **mot de passe**
  - un **nom de base**

mysql_connect($server,$user,$password) : permet de se connecter au serveur **$server** en tant qu'utilisateur **$user** avec le mot de passe **$password**, retourne l'identifiant de connexion si succès, FALSE sinon.

mysql_select_db($base[,$id]) : permet de choisir la base **$base**, peut prendre un identifiant **$id** de connexion ; retourne TRUE en cas de succès, sinon FALSE.

5.4 Interface avec PHP - *déconnexion*

- Pour se **déconnecter** d'une base depuis un script PHP

mysql_close([$id]) : permet de fermer la connexion à un serveur de bases de données, l'argument optionnel **$id** est l'identifiant de session retourné à l'ouverture de la connexion.

5.4 Interface avec PHP - *Exemple*

```
if( $id = mysql_connect(''localhost'',''foobar'',''0478'') ) {
        if(mysql_select_db(''gigabase'') ) {
                echo ''Succès de connexion.'';
                /* code du script … */
        } else {
                die(''Echec de connexion à la base.'');
        }
        mysql_close($id);
} else {
        die(''Echec de connexion au serveur de base de données.'');
}
```

5.4 Interface avec PHP - *Interrogation*

Pour envoyer une requête à une base de donnée, il existe la fonction : **mysql_query($str)** qui prend pour paramètre une **chaîne de caractères qui contient la requête écrite en SQL** et **retourne un identificateur** de résultat ou FALSE si échec.

```
$result = mysql_query(''SELECT téléphone FROM Personnes WHERE
nom=\"$name\"'');
```

Recherche le téléphone d'une personne portant pour nom la valeur de la chaîne **$name**.

- L'identificateur de résultat **$result** permettra à d'autres fonctions d'extraire ligne par ligne les données retournées par le serveur.
- **Chaque appel à cette fonction retournera un tuple du résultat**. C'est pourquoi cette instruction pourra être utilisée au sein d'une boucle **while** qui s'arrêtera lorsque **mysql_query()** renverra FALSE.

5.4 Interface avec PHP - *Extraction de données (par tableau)*

mysql_fetch_row($result) : **retourne une ligne de résultat** (un tuple) sous la **forme d'un tableau**. Les éléments du tableau étant les valeurs des attributs de la ligne. Retourne FALSE s'il n'y a plus aucune ligne.

```
$request = ''SELECT * FROM users'';
if($result = mysql_query($request)) {
        while($ligne = mysql_fetch_row($result)) {
                $id = $ligne[0];
                $name = $ligne[1];
                $address = $ligne[2];
                echo ''$id - $name, $address <br />'';
        }
} else {
        echo ''Erreur de requête de base de données.'';
}
```

Ici, **on accède aux valeurs** de la ligne **par leur indice dans le tableau**.

5.4 Interface avec PHP - *Extraction de données (par association)*

mysql_fetch_array($result) et **mysql_fetch_assoc($result)** : retournent un **tableau associatif**. Les clés étant les noms des attributs et leurs valeurs associées leurs valeurs respectives. Retourne FALSE s'il n'y a plus aucune ligne.

```
$request = ''SELECT * FROM users'';
if($result = mysql_query($request)) {
        while($ligne = mysql_fetch_array($result)) {
                $id = $ligne[''id''];
                $name = $ligne[''name''];
                $address = $ligne[''address''];
                echo ''$id - $name, $address <br />'';
        }
} else {
        echo ''Erreur de requête de base de données.'';
}
```

Ici, **on accède aux valeurs** de la ligne **par l'attribut dans le tableau associatif.**

5.4 Interface avec PHP - *Statistiques sur une requête*

mysql_num_rows($result) : retourne le ***nombre de lignes*** retournées par la dernière requête SELECT dont on connaît l'identifiant de résultat **$result**.

```
$request = ''SELECT name FROM users WHERE birth > \''1980-05-10\''';
$result = mysql_query($request) or die(''Erreur de base de données.'');
$num = mysql_num_rows();
```

5.4 Interface avec PHP - *Fonctions sur le serveur*

mysql_create_db($base [, $id]) : création de la base **$base**.

mysql_db_name($result, $row [, $field]) : Lit les noms des bases de données. **$result** est l'identifiant de résultat issu de **mysql_list_dbs()**. **$row** est l'index dans le résultat. Retourne FALSE si échec.

mysql_db_query($base, $query [, $id]) : exécution de la requête **$query** sur la base **$base**. Retourne un identifiant de résultat si succès ou FALSE si échec.

mysql_drop_db($base [, $id]) : supprime la base de données **$base**. Retourne TRUE si succès ou FASE si échec.

```
$result = mysql_list_dbs();
$num = mysql_num_rows($result);
for ($i=0;  $i<$num; $i++)
    echo mysql_db_name($result, $i).''<br />'';
```

Affiche la liste des bases de données du serveur.

5.4 Interface avec PHP - *Gestion des erreurs*

Il est recommandé de **tester systématiquement les valeurs retournées** par les fonction de traitement sur une base de données afin d'éviter la pollution de la page web par des *Warning*.

mysql_errno([$id]) : retourne le numéro d'erreur de la dernière opération MySQL effectuée sur la connexion courante ou celle d'identifiant **$id**.

mysql_error([$id]) : retourne le message d'erreur de la dernière opération MySQL effectuée sur la connexion courante ou celle d'identifiant **$id**.

```
$request = ''DELETE FROM users WHERE name LIKE \''Martin%\'''';
if($result = mysql_query($request)) {
        …
} else {
        echo ''Erreur de base de données n°''.mysql_errno().'': ''.mysql_error();
}
```

Remerciements

- Ces transparents doivent **beaucoup** à :
  - **Hugo Etiévant**
  - **Jean-Luc Hainaut** (auteur d'un excellent livre sur les BDs)

  dont les transparents sont extraordinairement complets et bien faits.

Merci !!