

Introduction aux bases de données

Pierre Lefebvre

PLAN DU COURS

- **Partie 1 : Pratique du langage SQL**
 - Introduction + définitions des données
 - Manipulation des données
 - Recherche des informations
 - Les requêtes : les opérateurs de base
 - Recherche des informations
- **Partie 2 : Conception de bases de données**
 - Conception de bases de données
 - Etudes de cas
 - Manipulation d'un outil de conception
- **Partie 3: Programmation d'une base de données**
 - Programmation procédurale
 - Etudes de cas
 - Les procédures stockées et triggers
 - Accès d'une base de données avec Java

Bibliographie

- *Bases de données*, J-L Hainaut, Dunod, 2012
- *Administrez vos bases de données avec MySQL*, Chantal Gribaumont, Le livre du zéro, 2012
- *Apprendre SQL avec MySQL*, Christian soutou, Eyrolles, 2006
- *SQL*, Ryan Stephens, Ronald Plew, Arie Jones, Pearson, 2013
- ...

Des données ? Est ce important pour vous ?

- Des relevés de banques, de cartes de crédit
- Des carnets d'adresses
- La consommation de téléphone
- Des inscriptions à des clubs, associations,
- Des papiers utiles
- Des horaires de transport
- Des programmes de télé
- ...

Des données ? Est ce important pour vous ?

- Assurer l'accès aux données
- Assurer la sécurité de ces données
 - Confidentialité
 - Authentification
 - Intégrité
- Le tout efficacement, rapidement, partout etc...
- **C'est important pour vous...**
- **C'est impératif pour les entreprises !**

Pourquoi des bases de données ?

- stocker de **gros volumes** d'informations
- **partager** des informations par une communauté de personnes
- **gérer l'accès** à ces informations
- gérer des **informations cohérentes et non-redondantes**

Les limites à l'utilisation des fichiers

- **L'utilisation de fichiers impose à l'utilisateur de connaître :**
 - le mode d'accès (séquentielle, indexée, ...)
 - la structure physique des enregistrements
 - et la localisation
- Toute modification de la structure des enregistrements (ajout d'un champ par exemple) entraîne la réécriture de tous les programmes qui manipulent ces fichiers.

Historique

- 1ère génération (années 70): réseau, hiérarchique (CODASYL, IMS,...)
 - LMD navigationnel
- 2ième génération (années 80) :relationnel (Oracle, Ingres, DB2, SQL server,...)
 - LMD non procédural
- 3ième génération (années 90)
 - objet (Object Store, Versant,...)
 - relationnel objet (Oracle8, DB2 UDS, Informix US)
- 4ième génération ?
 - déductif (« Datalog »), entrepôt de données, data mining, support du WEB (XML, XMLQL)
- Modèles pour conception
 - E/R, UML

Contexte des bases de données

- Les Bases de Données sont nécessaires à tous les domaines d'activité
industrie, commerce, services, recherche scientifique, ...
- Leur succès est lié aux avancées scientifiques et technologiques en gestion de l'information et des communications
- Les Systèmes de Gestion des Bases de Données (SGBD) sont parmi les systèmes logiciels les plus complexes
- Poids économique: par exemple le SGBD Oracle a un CA 14,4 Milliards de \$ et 65000 collaborateurs, utilisé par 98 entreprises du classement Fortune 100

Définition d'une Base de données

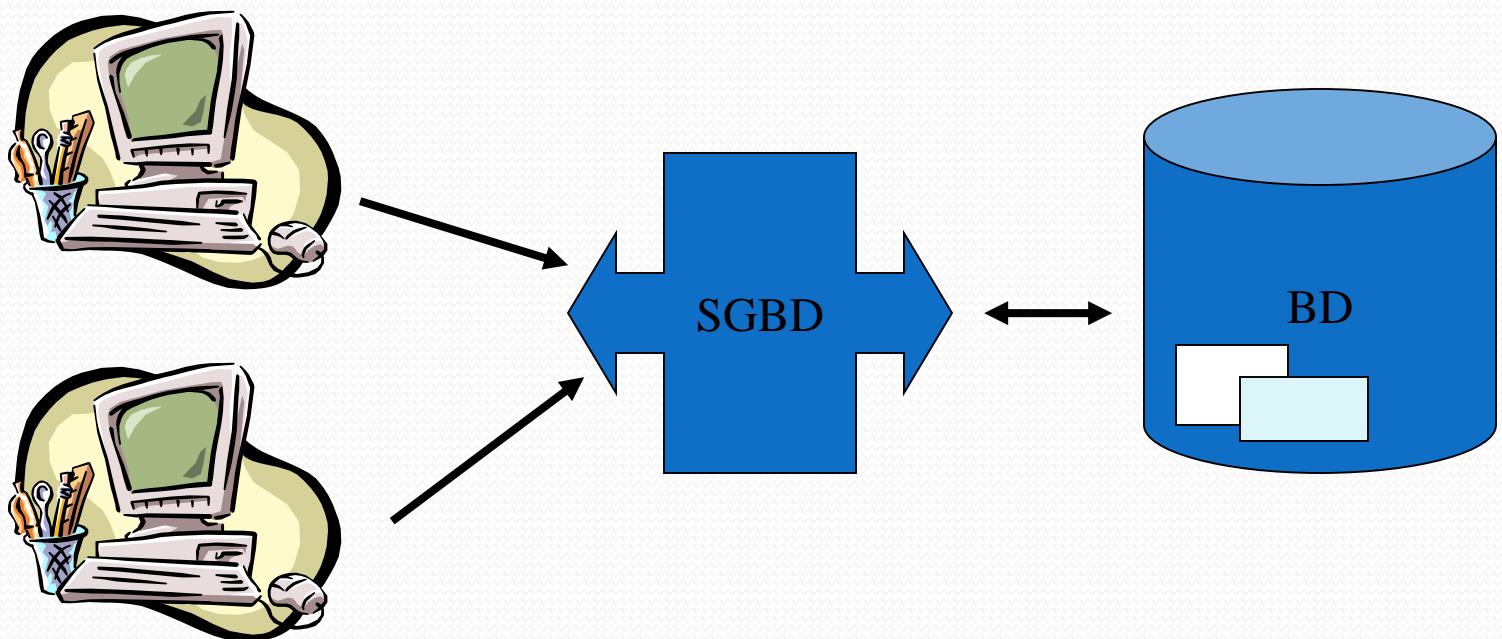
- **Définition** : on peut considérer une **Base de Données** (BD) comme une grande quantité de données (ou ensemble d'informations), centralisées ou non, servant pour les besoins d'une ou plusieurs applications, interrogables et modifiables par un groupe d'utilisateurs travaillant en parallèle.
- **Exemples d'application**
 - Système Socrate : SNCF
 - Annuaire électronique
 - Catalogue électronique d'une bibliothèque

Définition d'une base de données

- **Définition** : une base de données est un ensemble structuré de données (1) enregistrées sur des supports accessibles par l'ordinateur (2) pour satisfaire simultanément plusieurs utilisateurs (3) de manière sélective (4) en un temps opportun (5).
 - (1) : Organisation et description de données
 - (2) : Stockage sur disque
 - (3) : Partage des données
 - (4) : Confidentialité
 - (5) : Performance

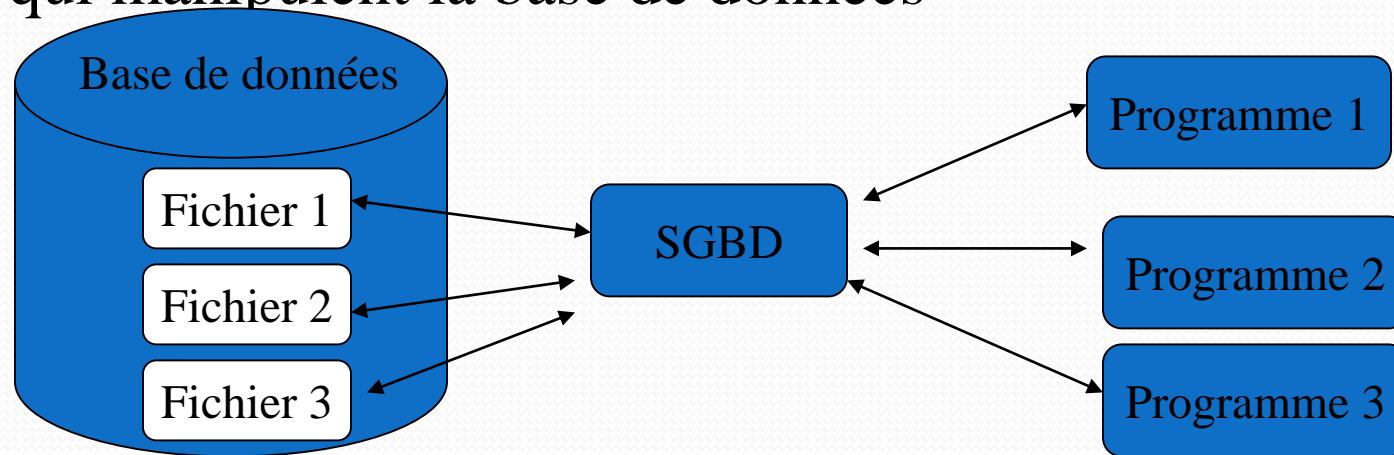
SGBD

- Définition : Le logiciel qui permet d'interagir avec une BD est Système de Gestion de Base de Données (SGBD)



SGBD

- Un SGBD est un intermédiaire entre les utilisateurs et les fichiers physiques
- Un SGBD facilite
 - la gestion de données, avec une représentation intuitive simple sous forme de table
 - la manipulation de données. On peut insérer, modifier les données et les structures sans modifier les programmes qui manipulent la base de données



Objectifs des SGBD (1)

- Indépendance physique (1) : Plus besoin de travailler directement sur les fichiers physiques (tels qu'ils sont enregistrés sur disque). Un SGBD nous permet de décrire les données et les liens entre elles d'une façon logique sans se soucier du comment cela va se faire physiquement dans les fichiers. On parle alors d'image logique de la base de données, (ou aussi description logique ou conceptuelle ou encore de schéma logique). Ce schéma est décrit dans un modèle de données par exemple le modèles de tables, appelé le modèle relationnel



Objectifs des SGBD (2)

- **Indépendance physique (2)** : La manipulation des données doit être faciliter en travaillant directement sur le schéma logique. On peut insérer, supprimer, modifier des données directement sur l'image logique. Le SGBD va s'occuper de faire le travail sur les fichiers physiques.
- **Indépendance logique** : Un même ensemble de données peut être vu différemment par des utilisateurs différents. Toutes ces visions personnelles des données doivent être intégrés dans une vision globale.
- **Manipulations des données par des non informaticiens**. Il faut pouvoir accéder aux données sans savoir programmer ce qui signifie des langages « quasi naturels ».
- **Efficacité des accès aux données** : Ces langages doivent permettre d'obtenir des réponses aux interrogations en un temps « raisonnable ». Il doivent donc être optimisés et, entre autres, il faut un mécanisme permettant de minimiser le nombre d'accès disques. Tout ceci, bien sur, de façon complètement transparente pour l'utilisateur.

Objectifs des SGBD (3)

- **Administration centralisée des données** : Des visions différentes des données (entre autres) se résolvent plus facilement si les données sont administrées de façon centralisée.
- **Cohérence des données**. Les données sont soumises à un certain nombre de contrainte d'intégrité qui définissent un état cohérent de la base. Elles doivent pouvoir être exprimées simplement et vérifiées automatiquement à chaque insertion, modification ou suppression de données, par exemple :
 - l'âge d'une personne supérieur à zéro
 - Salaire supérieur à zéro
 - Etc

Dès que l'on essaie de saisir une valeur qui ne respecte pas cette contrainte, le SGBD le refuse.

Objectifs des SGBD (4)

- **Non redondance des données** : Afin d'éviter les problèmes lors des mises à jour, chaque donnée ne doit être présente qu'une seule fois dans la base.
- **Partageabilité des données** : Il s'agit de permettre à plusieurs utilisateurs d'accéder aux mêmes données au même moment. Si ce problème est simple à résoudre quand il s'agit uniquement d'interrogations et quand on est dans un contexte mono-utilisateur, cela n'est plus le cas quand il s'agit de modifications dans un contexte multi-utilisateurs. Il s'agit alors de pouvoir :
 - Permettre à deux (ou plus) utilisateurs de modifier la même donnée « en même temps »;
 - Assurer un résultat d'interrogation cohérent pour un utilisateur consultant une table pendant qu'un autre la modifie.

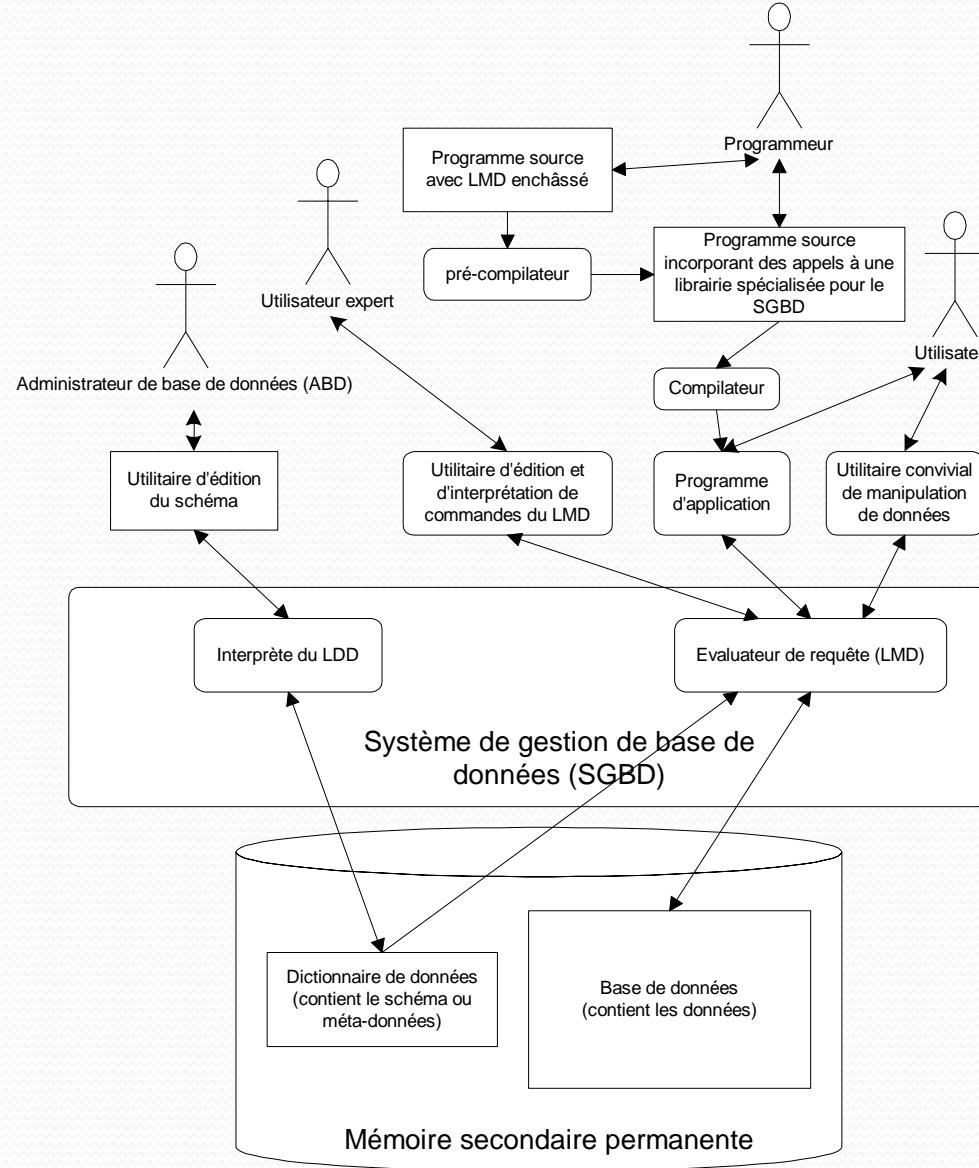
Objectifs des SGBD (5)

- **Sécurité des données**: Les données doivent pouvoir être protégées contre les accès non autorisés. Pour cela, il faut pouvoir associer à chaque utilisateur des droits d'accès aux données.
- **Résistance aux pannes** : Que se passe-t-il si une panne survient au milieu d'une modification, si certains fichiers contenant les données deviennent illisibles? Les pannes, bien qu'étant assez rares, se produisent quand même de temps en temps. Il faut pouvoir, lorsque l'une d'elles arrive, récupérer une base dans un état « sain ». Ainsi, après une panne intervenant au milieu d'une modification deux solutions sont possibles : soit récupérer les données dans l'état dans lequel elles étaient avant la modification, soit terminer l'opération interrompue.

Le marché des SGBD

- Les bases de données sont souvent livrées sous cette forme :
 - Un moteur de gestion du stockage, ce moteur peut être un serveur, ou une librairie.
 - Des applications d'utilisation et de manœuvre du moteur :
 - Un client « console »
 - Une application « standalone » à interface utilisateur évoluée.
 - Des applicatifs d'administration pour gérer les sauvegardes, les copies, les restaurations
 - Des ateliers de fabrication d'applications de gestion basées sur le moteur

Types d'utilisateur

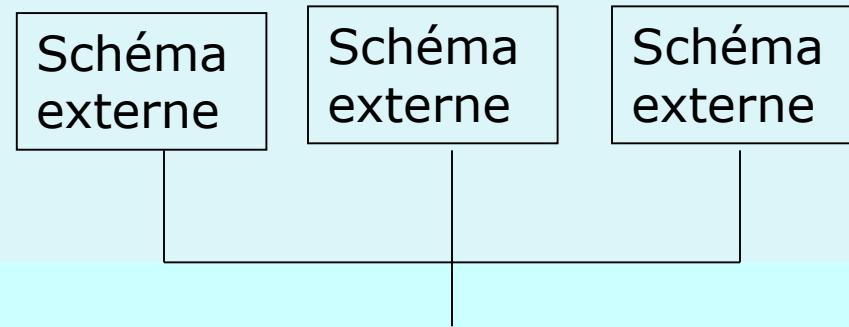


Description d'une base de données

Architecture ANSI/SPARC

Niveau externe

Vue utilisateur



Niveau conceptuel

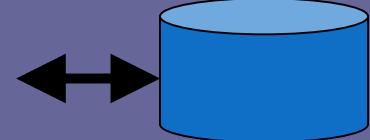
*Description des entités et
des relations*

Schéma conceptuel

Niveau interne

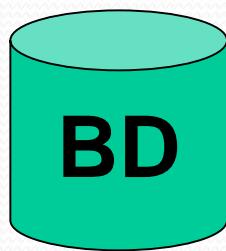
*Description de la façon dont
les données sont stockées*

Schéma interne

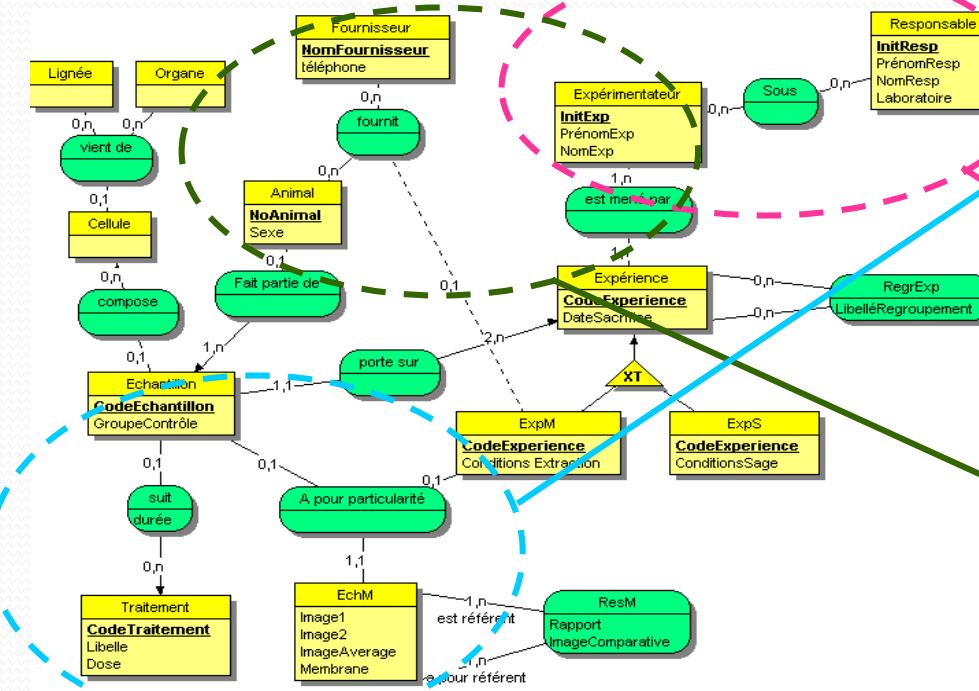


Les trois niveaux de description

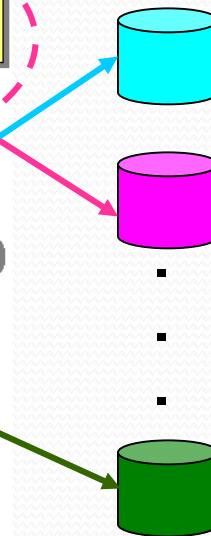
Niveau interne
Stockage



Niveau conceptuel ou logique

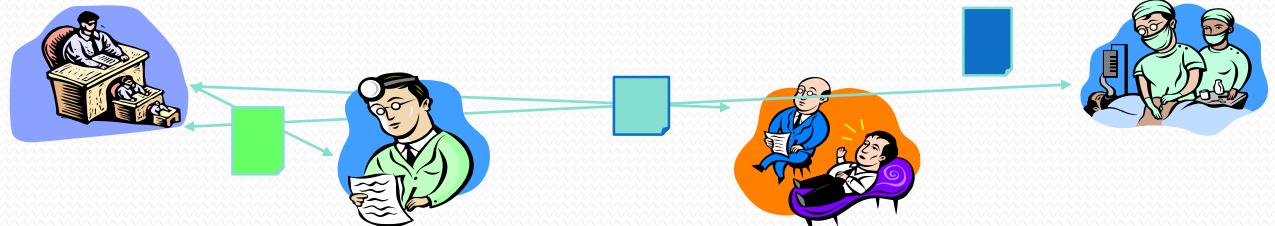


Vues



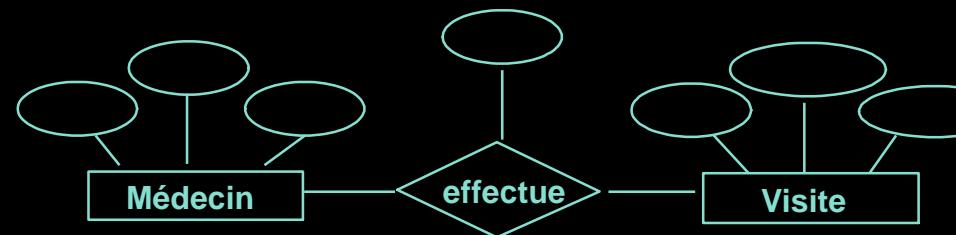
Les modèles en bases de données

Réel



Modèle conceptuel

- Indépendant du modèle de données
- Indépendant du SGBD



Modèle logique

- Dépendant du modèle de données
- Indépendant du SGBD

Codasyl

Relationnel

Objet

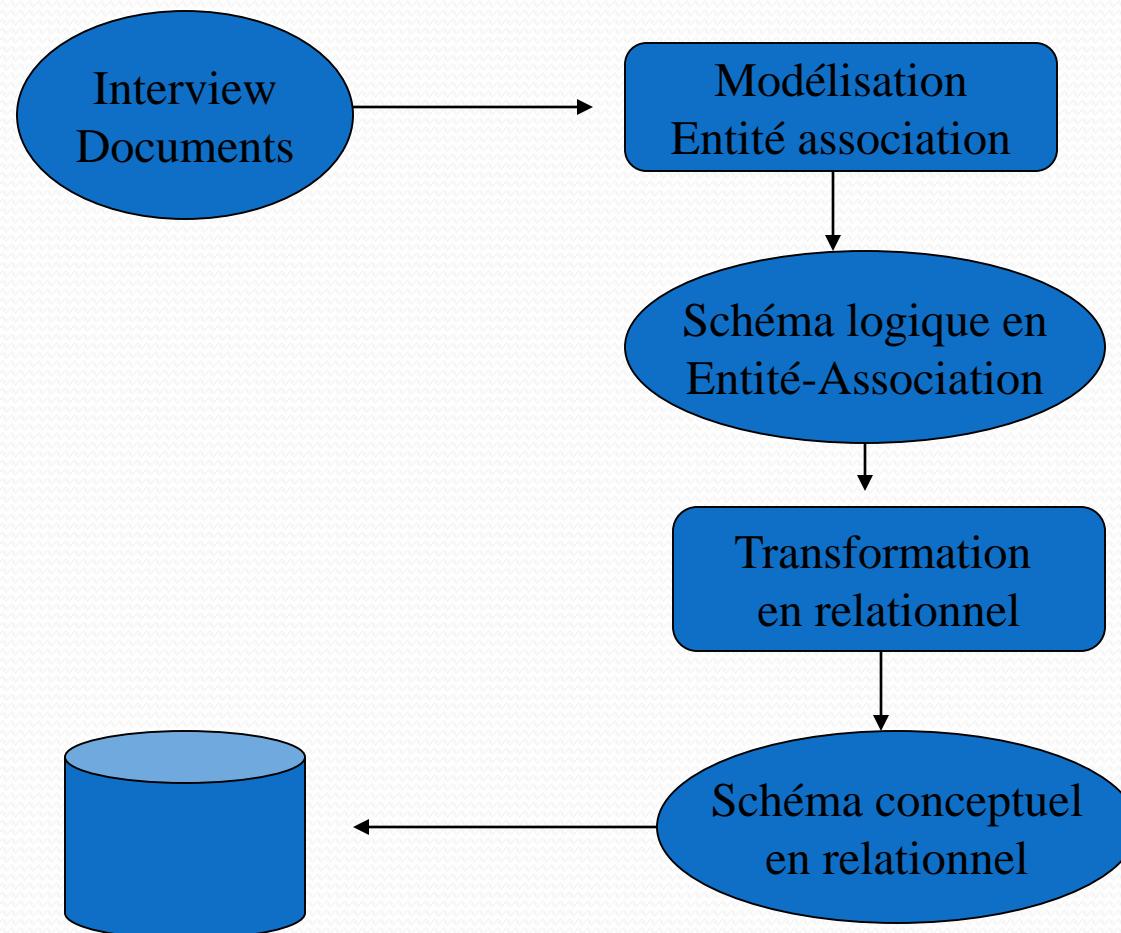
XML

Modèle Physique

- Dépendant du modèle de données
- Dépendant du SGBD

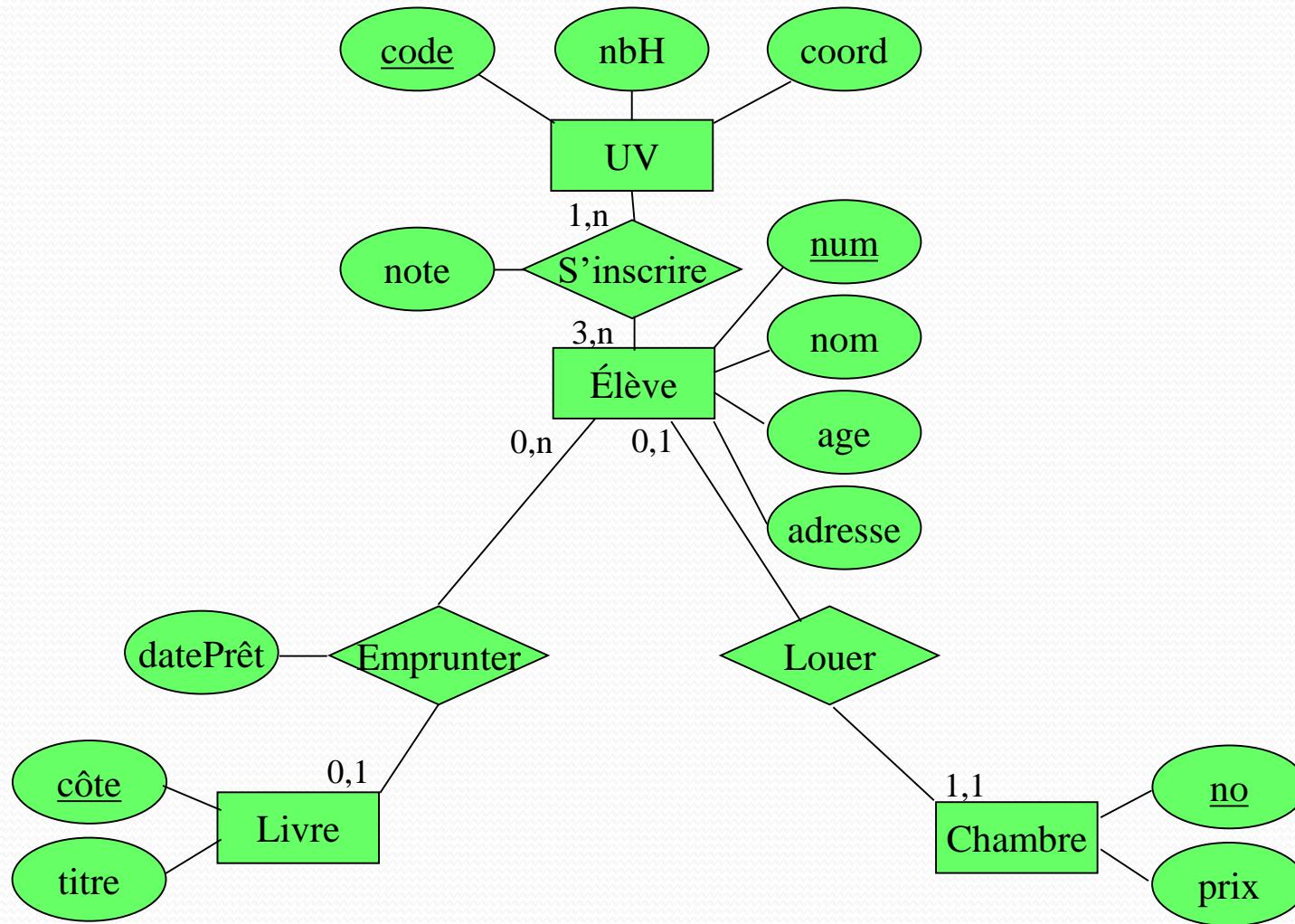
- Organisation physique des données
- Structures de stockage des données
- Structures accélératrices (index)

Démarche de construction d'une BD relationnelle

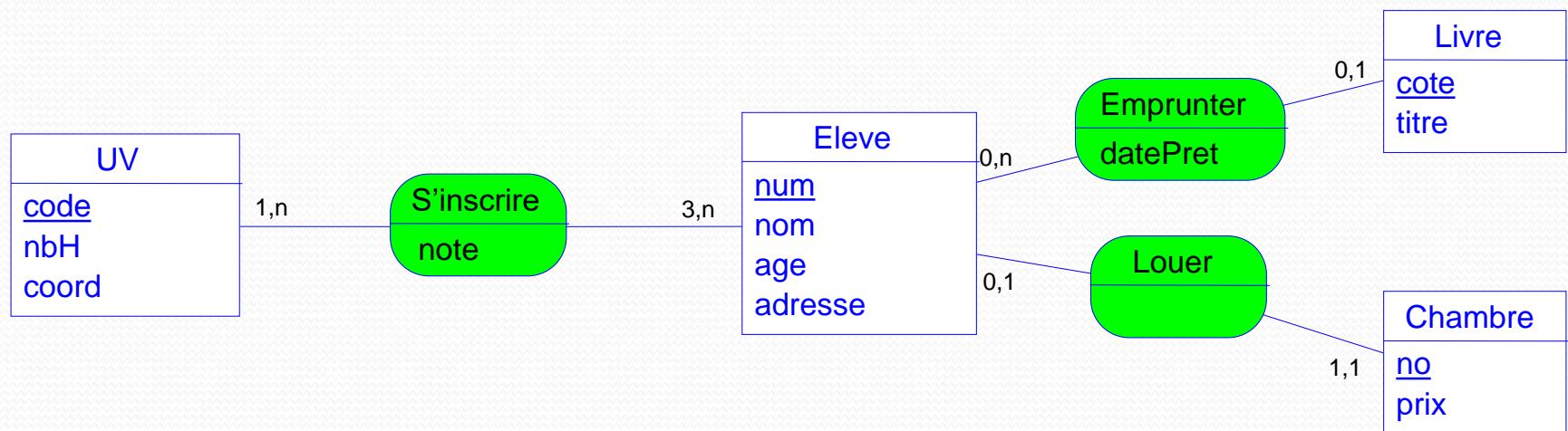


Modèle Entité/Association

[Chen 76]



Modèle Entité/Association



Définition et description des données niveau logique

- **Permet la description**
 - Des objets : exemple OUVRAGES, ETUDIANTS
 - Des propriétés des objets (attributs) : exemple Titre de OUVRAGES
 - Des liens entre les objets : un OUVRAVE peut être emprunté par un ETUDIANT
 - Des contraintes : le nombre d'exemplaires d'un OUVRAVE est supérieur à zéro
- Cette description est faite selon un modèle de données.
- **Un modèle de données** est un ensemble de concepts permettant de décrire la structure d'une base de données. La plupart des modèles de données incluent des opérations permettant de mettre à jour et questionner la base. Le modèle de données le plus utilisé est le modèle relationnel
- Cette description va donner lieu à un schéma de base de données. Un schéma de base de données se compose d'une description des données et de leurs relations ainsi que d'un ensemble de contraintes d'intégrité.

Modèle conceptuel / BDD Relationnelle

- **Modèle conceptuel**
 - entités
 - attributs
 - relations
- **Base de Données relationnelle**
 - tables
 - colonnes
 - clés

Définition et description des données niveau physique

- Description informatique des données et de leur organisation : en terme de fichiers, d'index, de méthodes d'accès, ...
- Passage du modèle logique au modèle physique tend à être assisté par le SGBD : transparent et/ou semi-automatique
- Objectifs : optimiser les performances

Manipulation et restitution des données

- Afin de réaliser les opérations suivantes
 - Insertion : saisir des données
 - Supprimer
 - Modifier
 - Interroger : rechercher des données via des requêtes

La manipulation des données est mise en œuvre à l'aide d'un Langage de manipulation de Données (LMD). SQL (Structured Query Language) est le langage standard de manipulation de BD

Modèles de SGBD

- **Quelques modèles logiques :**
 - Modèle hiérarchique
 - Modèle réseau
 - Modèle relationnel
 - Modèle objet
- **Quelques SGBD (relationnels du marché)**
 - Micro : ACCESS, Paradox, Dbase, PostgreSQL, MySQL, ...
 - Gros système : DB2, ORACLE, SYBASE, ...

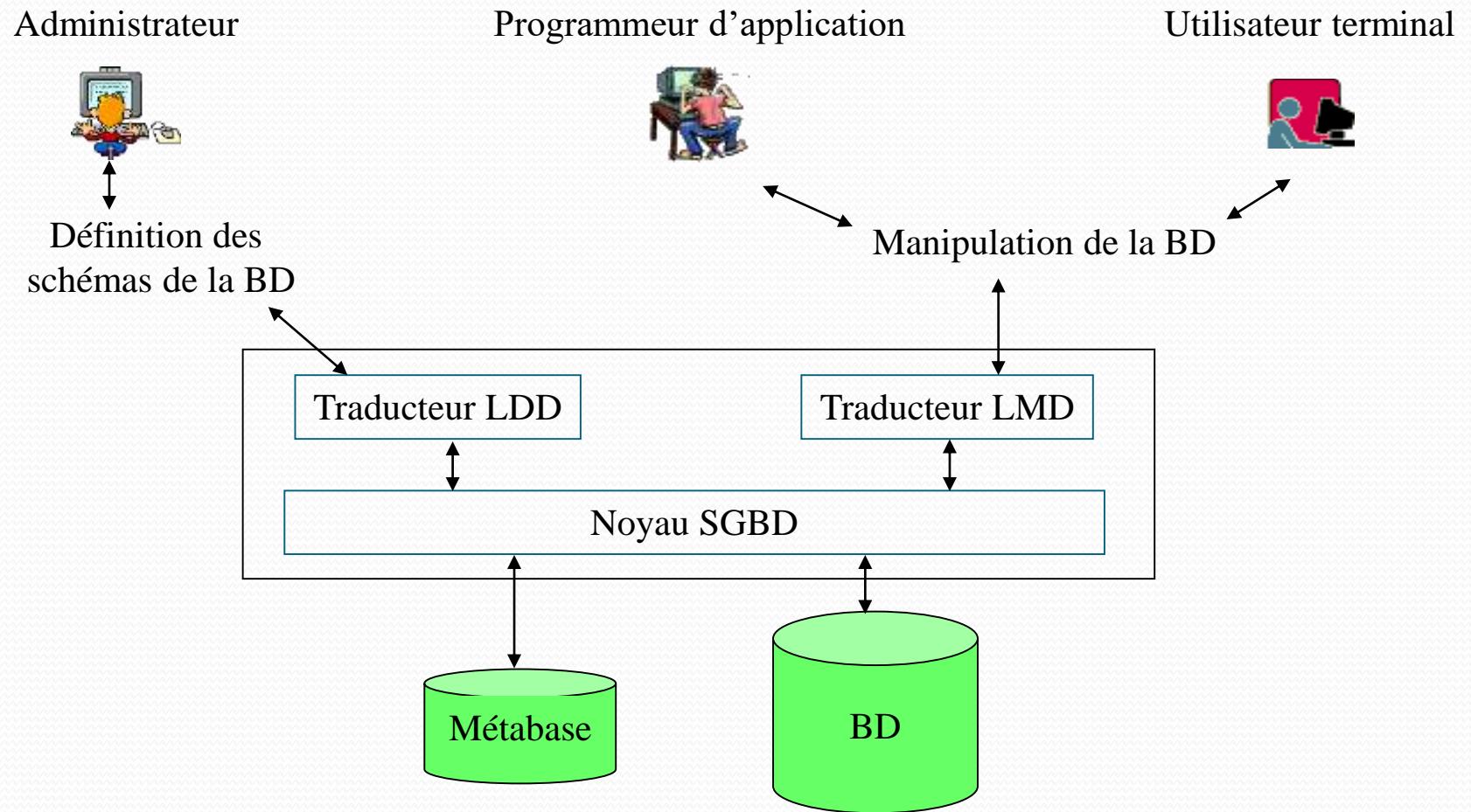


Les objets dans une base de données

- **Tables**
 - Une table enregistre des enregistrements qui décrivent une instance d'une entité
- **Vues**
 - Les vues sont des résultats d'exploration de données que l'on fait apparaître comme une table.
- **Index**
 - Un index est une table d'encodage qui optimise l'accès aux données
- **User**
 - Un objet User représente un utilisateur des données.

Architecture des SGBD

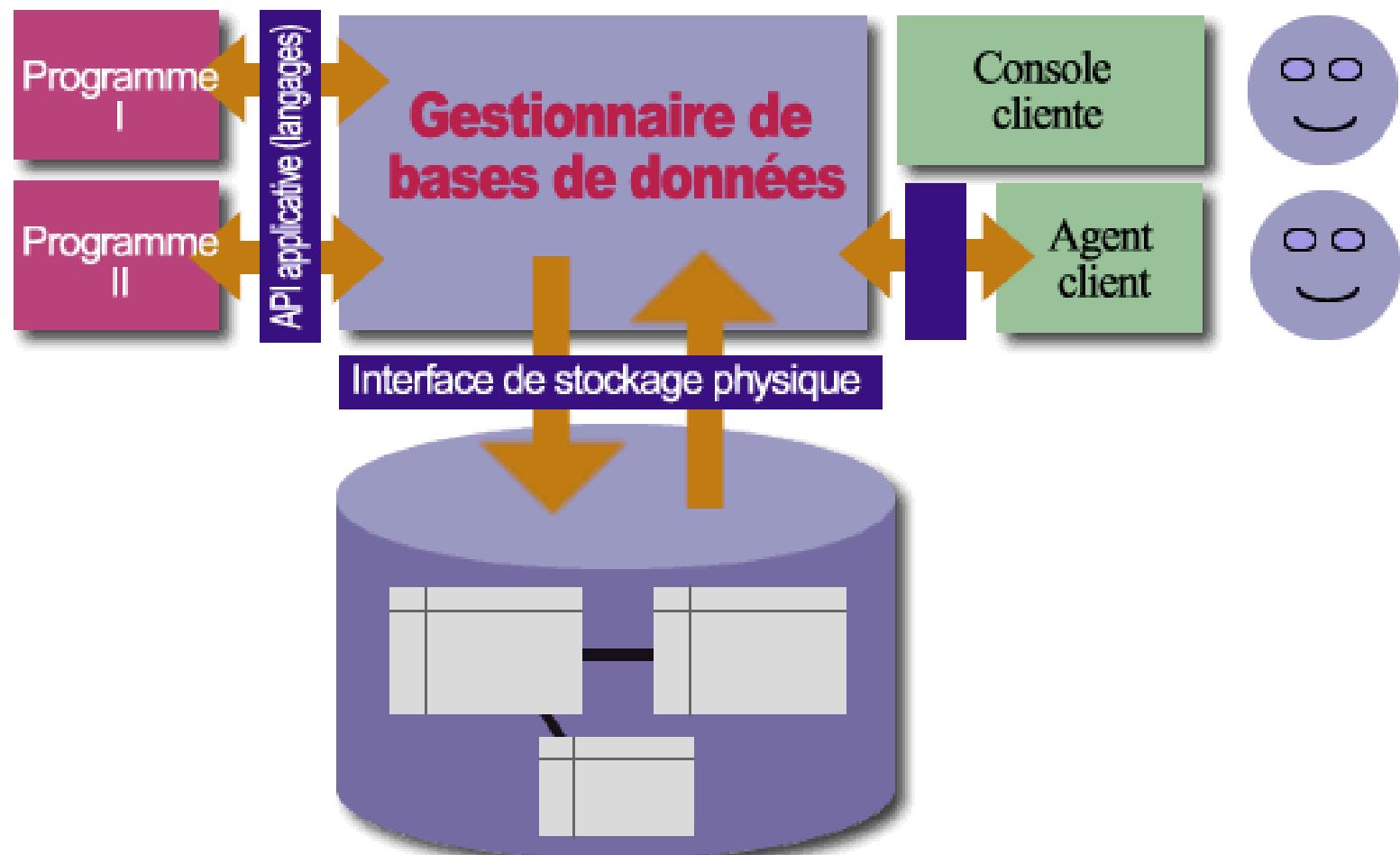
Architecture d'un SGBD



L'architecture des SGBD

- Données sur le serveur partagées entre N clients
- Interfaces graphiques sur la station de travail personnelle
- Communication par des protocoles standardisés
- Clients et serveurs communiquant par des requêtes avec réponses

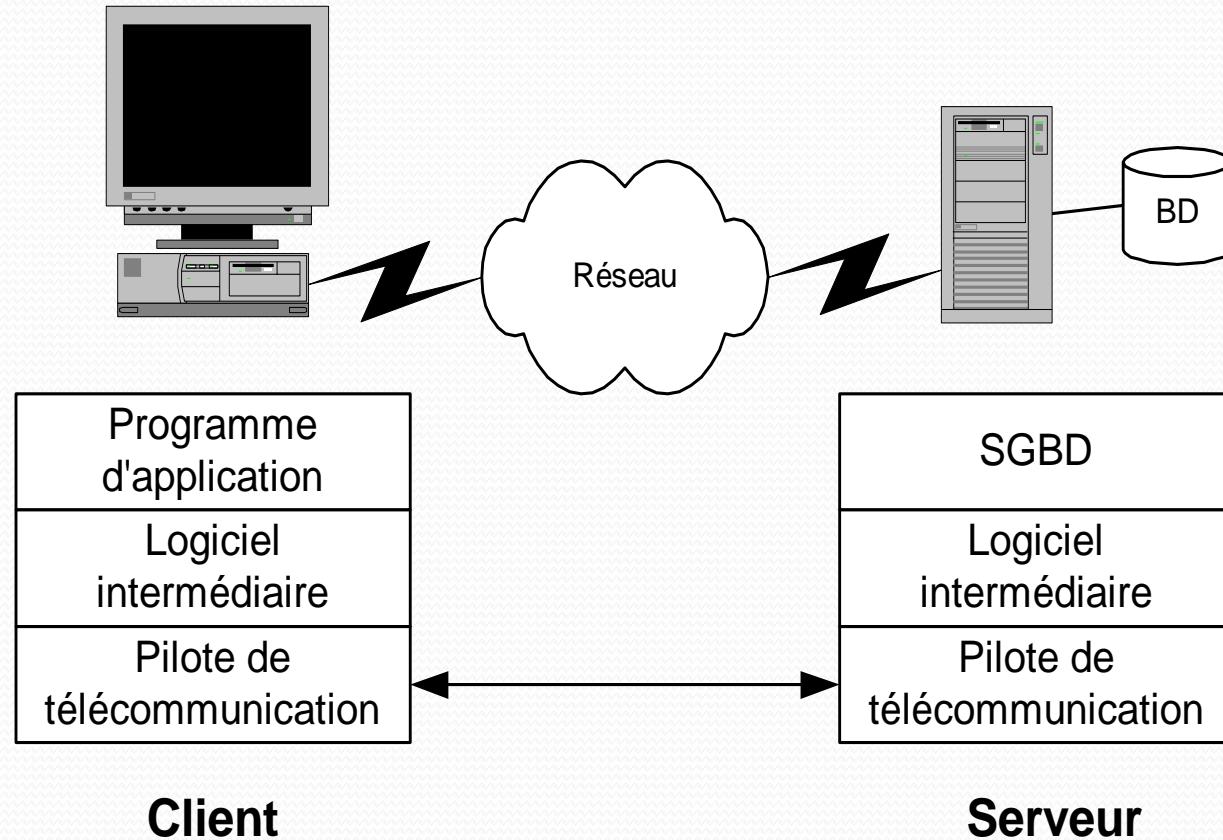
Structure d'une base de données



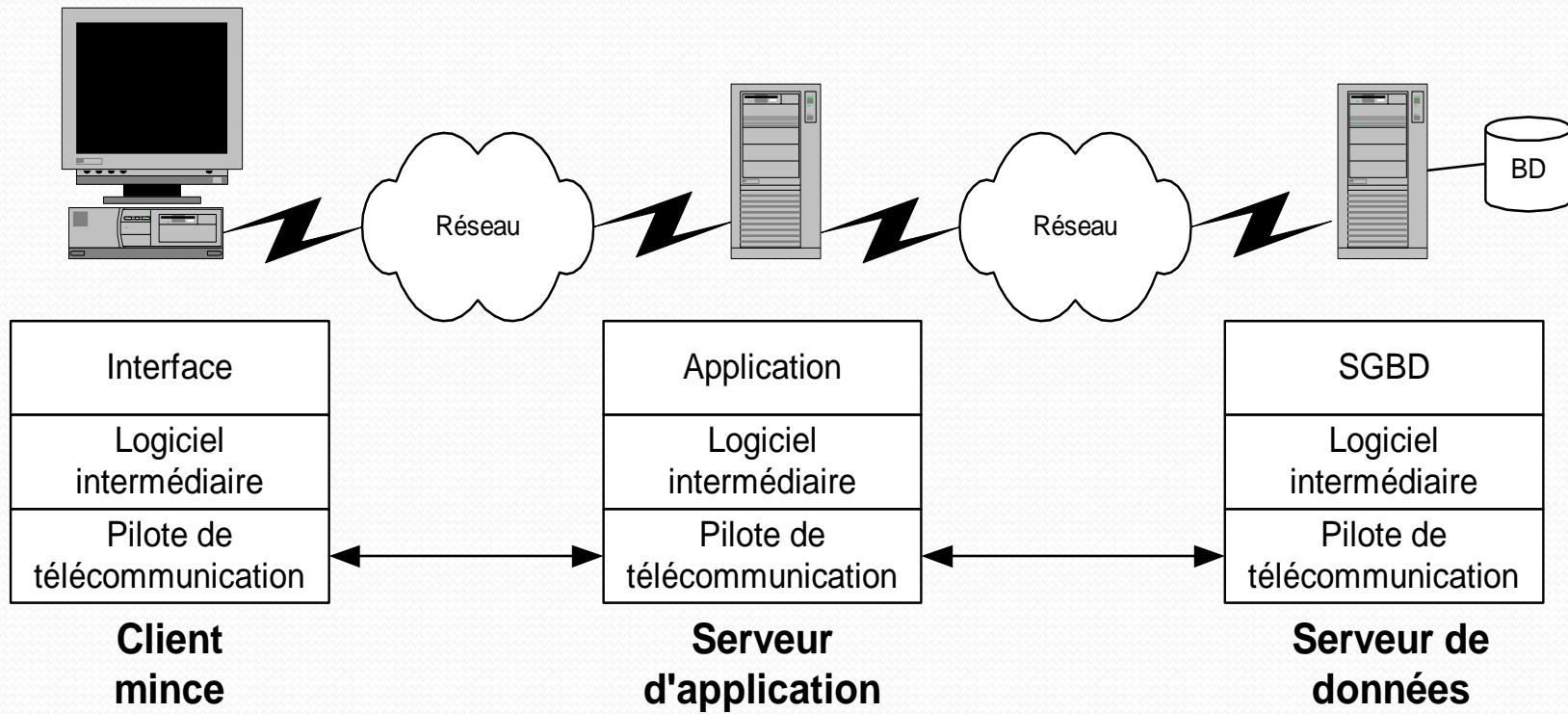
Architecture des bases de données

- *Architecture centralisée*
 - programme d'application et SGBD sur même machine (même site)
 - premiers systèmes
- *Architecture du type client-serveur*
 - programme d'application = *client*
 - interface (« GUI ») + traitement du domaine d 'application
 - SGBD = *serveur* (*de données* « *data server* »)
 - machines (sites) différentes
 - *deux couches, niveaux* (“*two tier* ”)

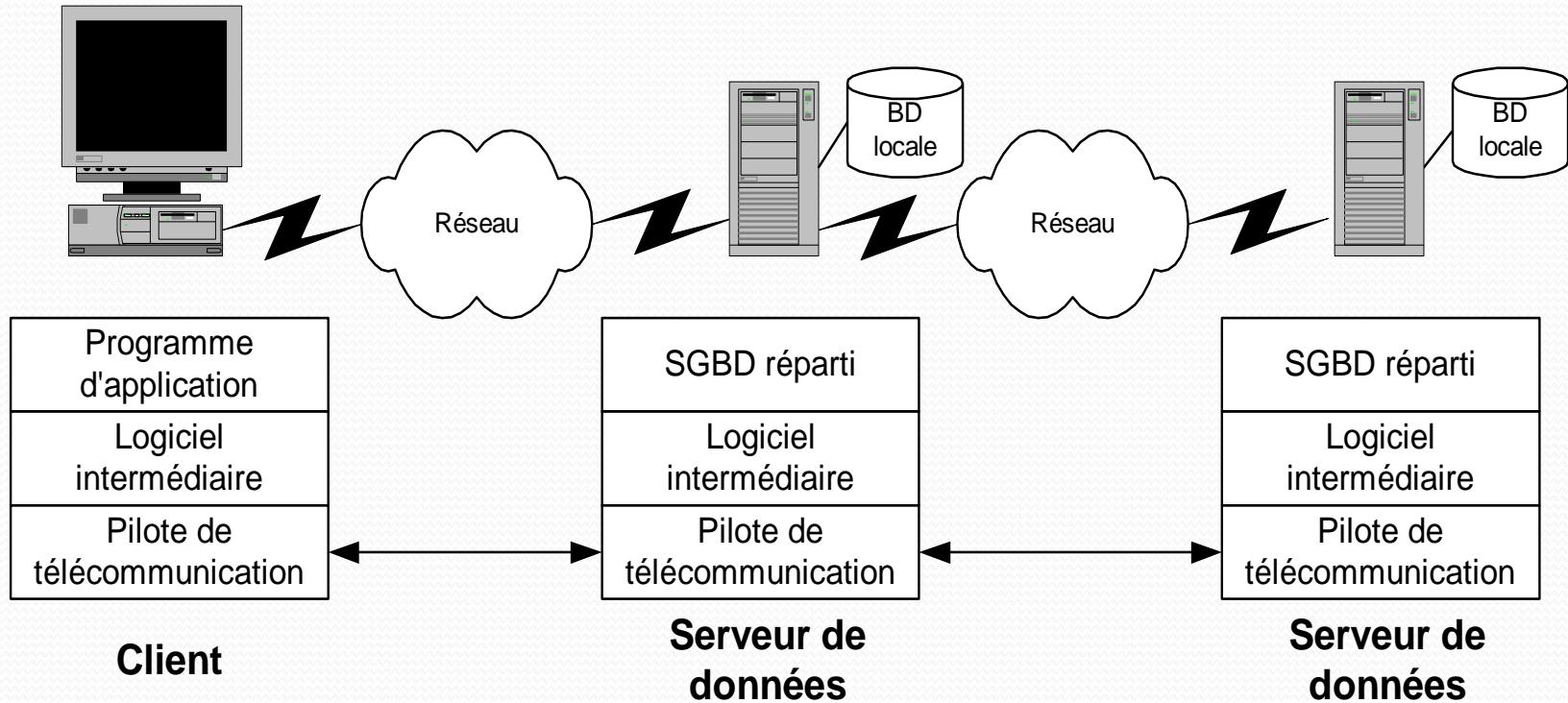
Architecture client / serveur



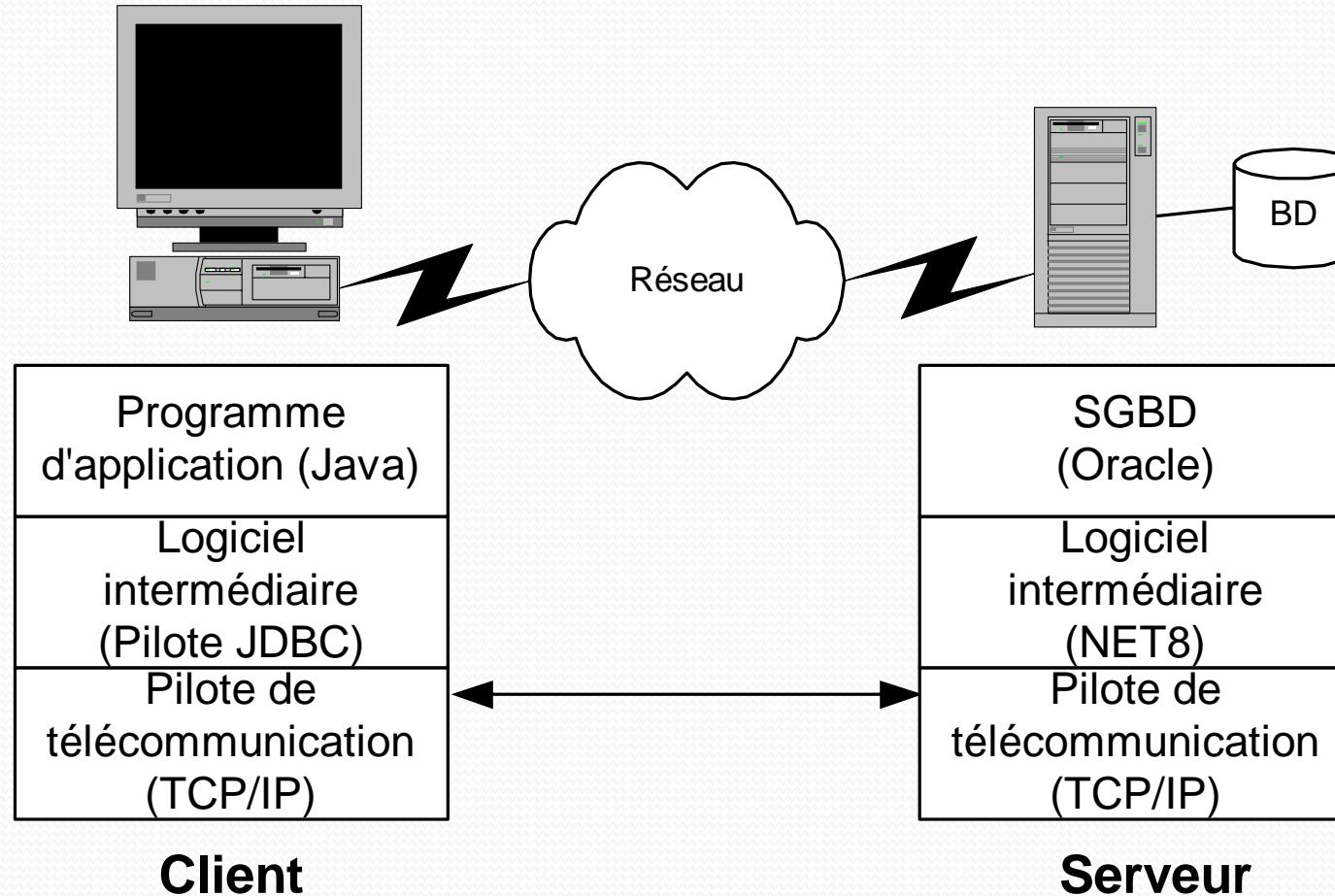
Architecture 3 tiers



Base de données distribuées



Architecture Java



Définition des données

Algèbre relationnelle

- C'est l'ensemble des opérations possible sur les relations.
On distingue :
 - La **projection** : on ne sélectionne qu'un ou plusieurs attributs d'une relation en ignorant les autres.
 - La **sélection** : on sélectionne tout ou partie des tuples en fonction de critères de sélection qui portent sur les valeurs des attributs.
 - La **jointure** : on fabrique une nouvelle relation à partir de 2 ou plusieurs en prenant comme pivot (ou élément commun) un ou plusieurs attributs.
- Cette algèbre est facilement possible avec la syntaxe SQL (Strutured Query Language) :

SELECT attributs FROM relations WHERE criteres

Tables, lignes et colonnes

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

Tables, lignes et colonnes

schéma

ligne

colonne obligatoire

colonne facultative

données

		CLIENT	LOCALITE	(CAT)	COMPTE
NCLI	NOM	ADRESSE			
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

Projection

Personnes

Nom	Prénom	E-Mail	Laboratoire
Charnay	Daniel	charnay@in2p3.fr	Centre de Calcul
Macchi	Pierre-Etienne	macchi@in2p3.fr	Centre de Calcul

On projette la table *Personnes* sur les colonnes nom, prénom

SELECT nom,prénom FROM *Personnes*

Nom	Prénom
Charnay	Daniel
Macchi	Pierre-Etienne

Sélection

Personnes

Nom	Prénom	E-Mail	Laboratoire
Charnay	Daniel	charnay@in2p3.fr	Centre de Calcul
Macchi	Pierre-Etienne	macchi@in2p3.fr	Centre de Calcul
Boutherin	Bernard	Boutherin@isn.in2p3.fr	ISN

On ne sélectionne que les tuples dont l'attribut laboratoire est égal à ISN

SELECT * FROM *Personnes* WHERE Laboratoire="ISN"

Nom	Prénom	E-Mail	Laboratoire
Boutherin	Bernard	Boutherin@isn.in2p3.fr	ISN

Jointure

Personnes

Nom	Prénom	E-Mail	Labo
Charnay	Daniel	charnay@in2p3.fr	CC
Macchi	Pierre-Etienne	macchi@in2p3.fr	CC
Boutherin	Bernard	Boutherin@isn.in2p3.fr	ISN

On joint les
2 tables
grâce à la
colonne
Labo.

Unité

Ville	Labo
Villeurbanne	CC
Villeurbanne	CC
Grenoble	ISN

SELECT Personnes.Nom, Personnes.Prénom, Unité.Ville FROM
Personnes, Unité WHERE Personnes.Labo=Unité.Labo



Nom	Prénom	Ville
Charnay	Daniel	Villeurbanne
Macchi	Pierre-Etienne	Villeurbanne
Boutherin	Bernard	Grenoble

Identifiants et clés étrangères

Un **identifiant** (ou clé primaire) 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 sert à référencer une ligne de la table T.

Identifiants et clés étrangères

identifiant

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

clé étrangère

dans le schéma
dans les données

Schéma et contenu

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	

le schéma

les données

Schéma et contenu

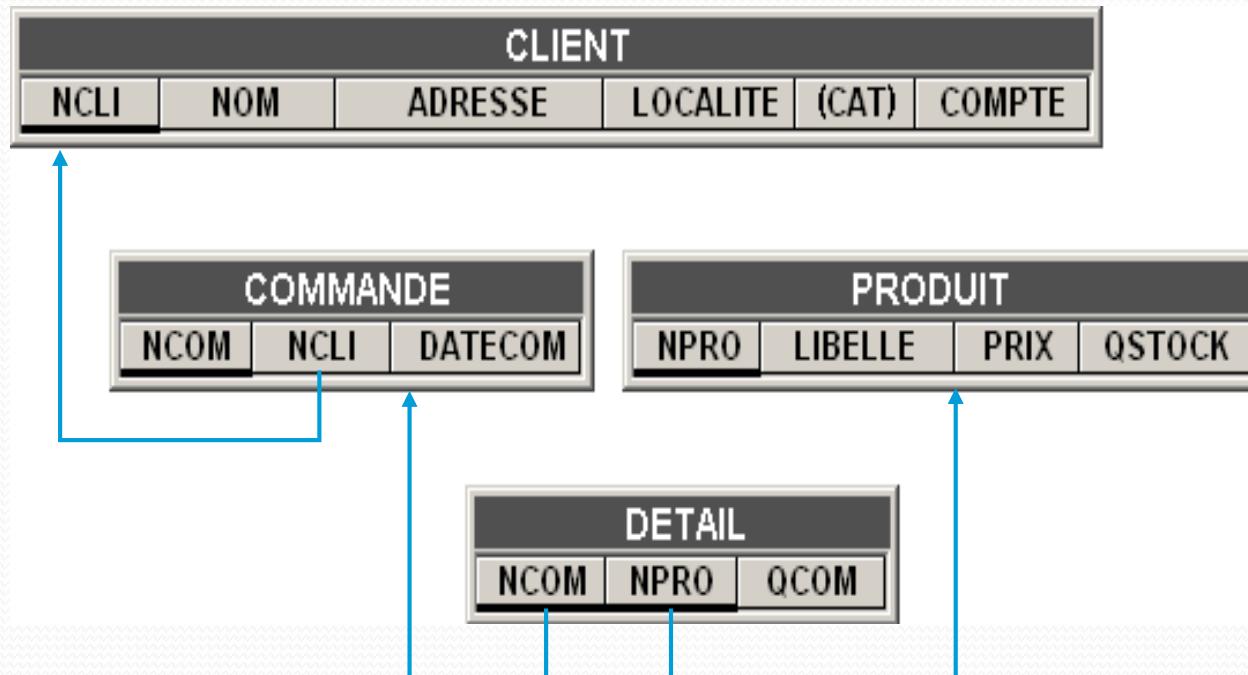
Le **schéma d'une table** définit sa structure. Il spécifie notamment :

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

Exemple de bases de données



Exemple de bases de données

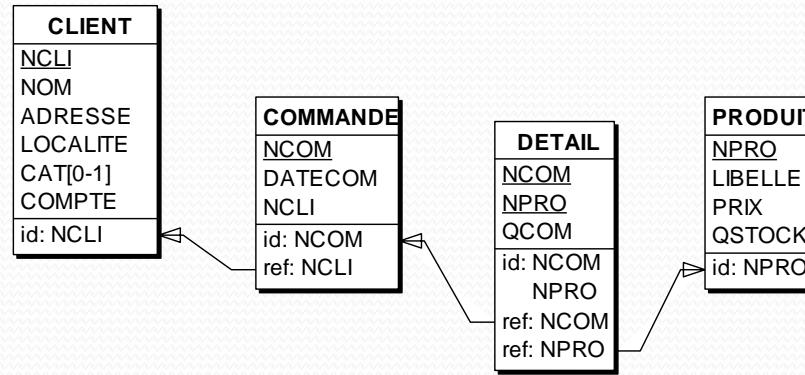
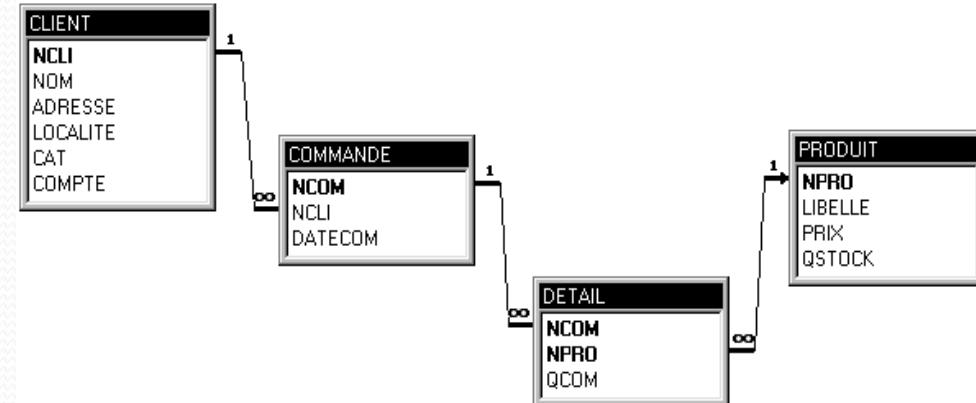
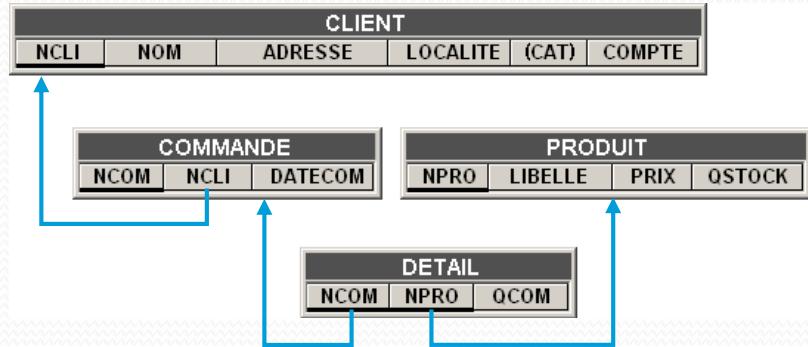
CLIENT					
NCLI	NOM	ADRESSE	LOCALITE	(CAT)	COMPTE
B062	GOFFIN	72, r. de la Gare	Namur	B2	-3200
B112	HANSENNE	23, r. Dumont	Poitiers	C1	1250
B332	MONTI	112, r. Neuve	Genève	B2	0
B512	GILLET	14, r. de l'Eté	Toulouse	B1	-8700
C003	AVRON	8, r. de la Cure	Toulouse	B1	-1700
C123	MERCIER	25, r. Lemaître	Namur	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

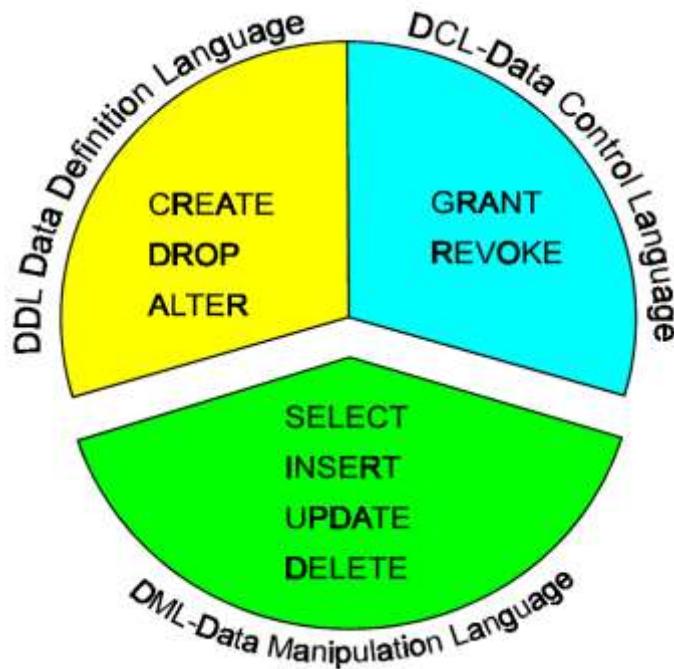
COMMANDE		
NCOM	NCLI	DATECOM
30178	K111	21/12/2008
30179	C400	22/12/2008
30182	S127	23/12/2008
30184	C400	23/12/2008
30185	F011	2/01/2009
30186	C400	2/01/2009
30188	B512	3/01/2009

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

Variantes de schéma



Classification des requêtes SQL



- Description des données : codification structuration, grâce à un Langage de Description de Données (LDD)
- Manipulation et restitution des données (insertion, mise à jour, interrogation)
 - Mise en œuvre à l'aide d'un Langage de Manipulation de Données (LMD)
 - S.Q.L. (Structures Query Language) : Langage standard
- Contrôle (partage, intégrité, confidentialité, sécurité)

Le « moment » d'utilisation du LDD

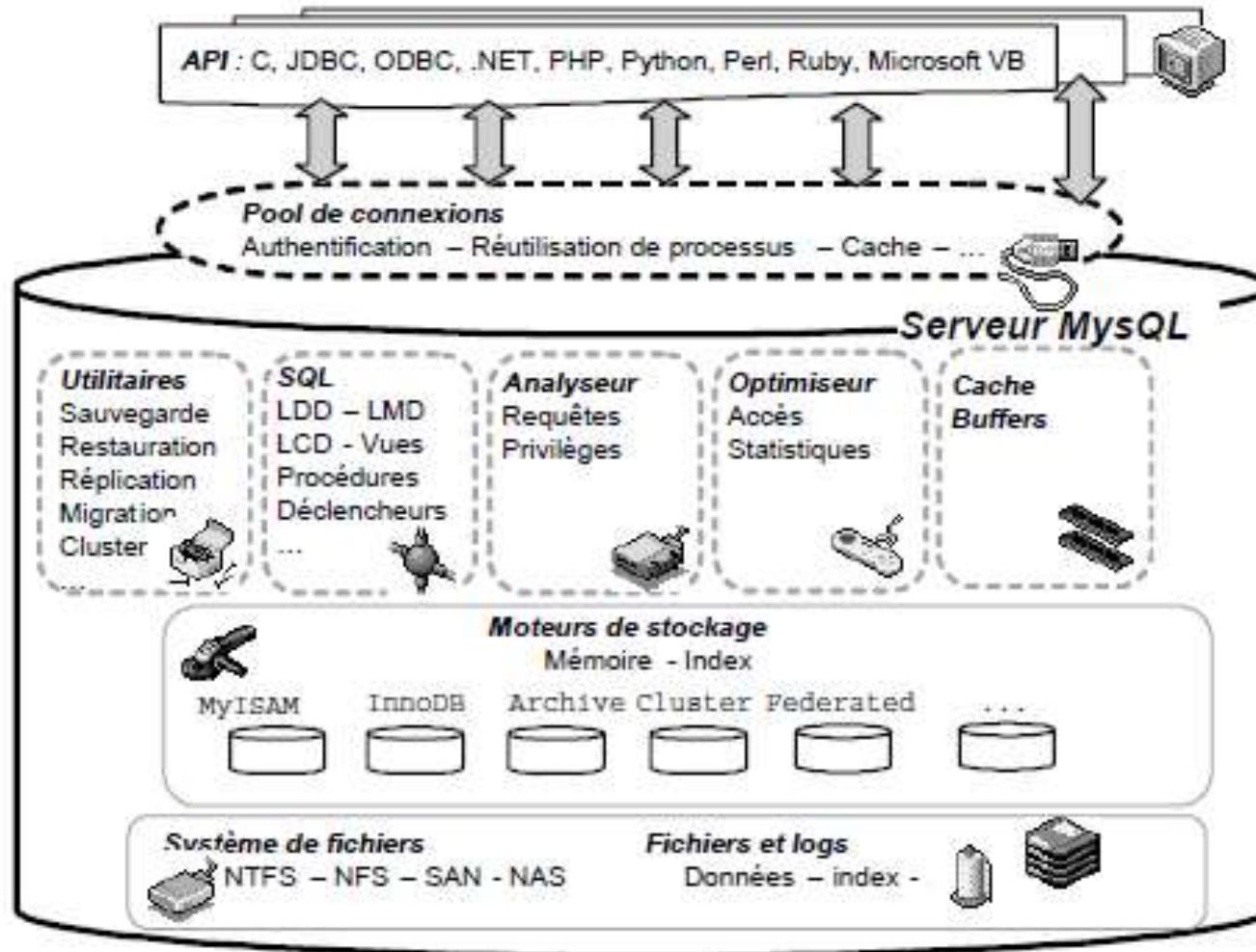
- Le LDD s'utilise au moment de la constitution de la base de données
- Les instructions DROP s'utilisent principalement :
 - Pour supprimer une table obsolète
 - Avant de recréer une nouvelle version d'une table
- Les instructions ALTER s'utilisent :
 - Pour mettre à jour la structure d'un objet sans perte de données (mise à jour progressive).

Principaux ordres du LMD

- INSERT
Insertion de données dans les objets
- UPDATE
Mise à jour de données dans les objets
- DELETE
Suppression de données dans les objets
- SELECT
sélection de données contenues dans les objets

Ces quatre opérations forment un modèle **ESSENTIEL** de la gestion de données informatiques : le CRUDE

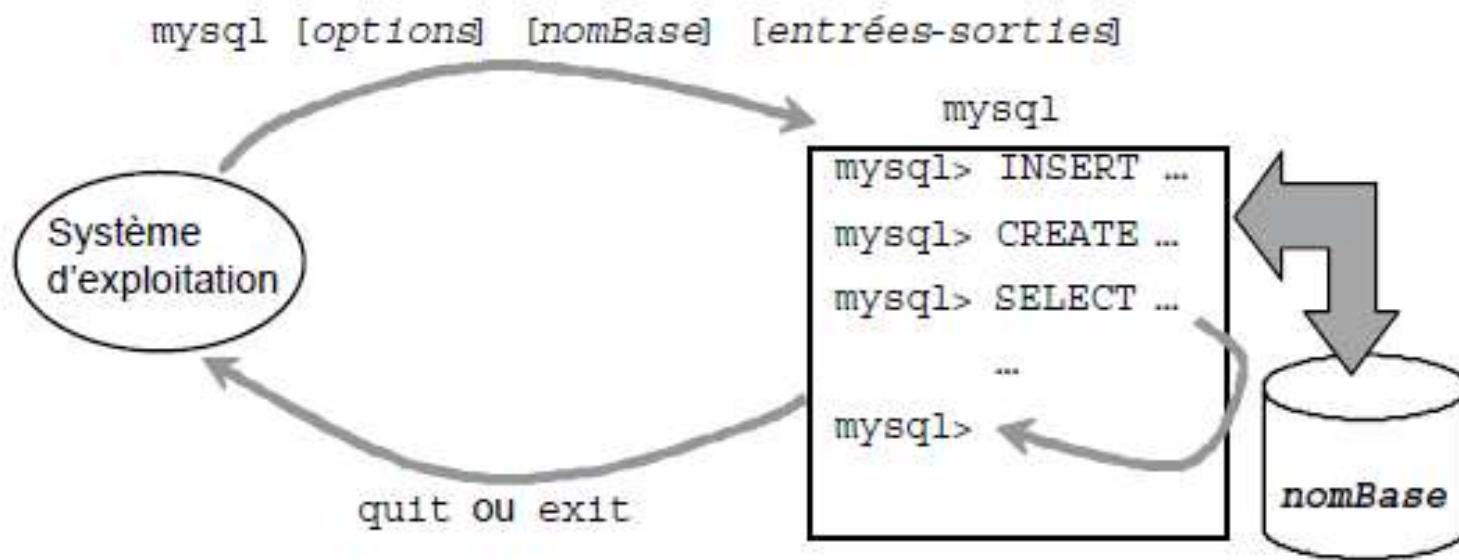
Offre MySQL



MySQL Products

- MySQL community server
 - > Free
- MySQL Enterprise
 - > Commercial
 - > Enterprise features - monitoring
- MySQL Cluster
 - > Provides fault tolerance
- MySQL embedded database
 - > Embedded in small devices
- MySQL Workbench
 - > GUI tool

Interface de commande en ligne



Création d'une table

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] [nomBase.]nomTable
( colonne1 type1
    [NOT NULL | NULL] [DEFAULT valeur1] [COMMENT 'chaine1']
[, colonne2 type2
    [NOT NULL | NULL] [DEFAULT valeur2] [COMMENT 'chaine2'] ]
[CONSTRAINT nomContrainte1 typeContrainte1] ...)
[ENGINE= InnoDB | MyISAM | ....];
```

Instruction SQL	Commentaires
<pre>CREATE TABLE bdsoutou.Compagnie (comp CHAR(4), nrue INTEGER(3), rue CHAR(20), ville CHAR(15) DEFAULT 'Paris' COMMENT 'Par defaut : Paris', nomComp CHAR(15) NOT NULL);</pre>	<p>La table contient cinq colonnes (quatre chaînes de caractères et un numérique de trois chiffres). La colonne ville est commentée.</p> <p>La table inclut en plus deux contraintes :</p> <ul style="list-style-type: none">• DEFAULT qui fixe <i>Paris</i> comme valeur par défaut de la colonne ville ;• NOT NULL qui impose une valeur non nulle dans la colonne nomComp.

Identificateurs

- Les noms de bases, relations, attributs, index et alias sont constitués de caractères alphanumériques plus _ et \$.
- Un nom comporte au maximum 64 caractères.
- Le point "." est un caractère réservé utilisé comme séparateur entre le nom d'une base, celui d'une relation et celui d'un attribut :

Select base1.table2.attribut3 from base1.table2

Différentes écritures SQL

Sans commentaire

```
CREATE TABLE Test(colonne  
DECIMAL(38,8));
```

```
CREATE TABLE  
Test  
(colonne  
DECIMAL(38,8)  
);
```

Avec commentaires

```
CREATE TABLE  
      -- nom de la table  
Test( #début de la description  
      COLONNE DECIMAL(38,8)  
      )  
      -- fin, ne pas oublier le point-virgule.  
;  
CREATE TABLE Test (/* une plus grande description  
des colonnes */  
COLONNE /* type : */ DECIMAL(38,8));
```

Clé primaire

```
CREATE TABLE Personne (
    id SMALLINT UNSIGNED PRIMARY KEY
        AUTO_INCREMENT,
    nom VARCHAR(40),
    prenom VARCHAR(40),
    adresse VARCHAR(40),
    laboratoire VARCHAR(30) )
```

- Cet identifiant numérique unique auto-incrémental, s'appelle une « *clé primaire* ».
- La numérotation des clés primaires, débute à 1 et pas à 0.

Clé primaire

- Notre clé primaire peut être associée simultanément à plusieurs attributs mais selon une syntaxe différente.
- Si au lieu de créer un identifiant numérique unique, on souhaite simplement interdire d'avoir des doublon sur le couple (*nom,prénom*) et d'en interdire la nullité, on va créer une clé primaire sur ce couple.
- La connaissance des seuls nom et prénom suffit à identifier sans ambiguïté un et un seul enregistrement.

- **Mauvaise syntaxe :**
- CREATE TABLE Personne (
 - *nom* VARCHAR(40) PRIMARY KEY,
 - *prénom* VARCHAR(40) PRIMARY KEY,
 - email VARCHAR(40) ,
 - laboratoire VARCHAR(30))
- **Bonne syntaxe :**
- CREATE TABLE Personne (
 - *nom* VARCHAR(40),
 - *prénom* VARCHAR(40),
 - email VARCHAR(40) ,
 - laboratoire VARCHAR(30) ,
 - PRIMARY KEY (*nom,prénom*)
-)

Attribut non nul

- Considérons que l'on souhaite que certains attributs aient obligatoirement une valeur. On utilisera l'option **NOT NULL**.
- Dans ce cas, si malgré tout, aucune valeur n'est fournie, la valeur par défaut – si elle est déclarée à la création de la relation – sera automatiquement affectée à cet attribut dans l'enregistrement.
- Au contraire, on utilisera l'option **NULL** si on autorise l'absence de valeur.

Valeur par défaut

- Pour donner une valeur par défaut à un attribut, on utilise l'option **DEFAULT**.
- Lors de l'ajout d'un enregistrement cette valeur sera affectée à l'attribut si aucune valeur n'est donnée.
- Exemple :
- ‘*téléphone*’ **DECIMAL(10,0) DEFAULT ‘0123456789’**
- Les attributs de type chaîne de caractères de la famille TEXT et BLOB ne peuvent pas avoir de valeur par défaut.

Attribut sans doublon

- Pour interdire l'apparition de doublon pour un attribut, on utilise l'option **UNIQUE**.
- Syntaxe :
 - **UNIQUE [nomde la contrainte](liste des attributs)**
- Pour interdire les doublons sur l'attribut *nom* mais les interdire aussi sur '*prénom*', tout en les laissant indépendants :
 - **UNIQUE(*nom*)**
 - **UNIQUE(*prénom*)**

<i>nom</i>	<i>prénom</i>
Dupond	Marc
Dupont	Pierre
Martin	Marc

enregistrement interdit
car 'Marc' est un doublon
dans la colonne 'prénom'

Attribut sans doublon

- Pour interdire tout doublon à un ensemble d'attributs (tuple), on passe en paramètre à **UNIQUE** la liste des attributs concernés.
- Pour interdit tout doublon du couple (*nom*, ‘*prénom*’) :
- **UNIQUE(*nom,prénom*)**

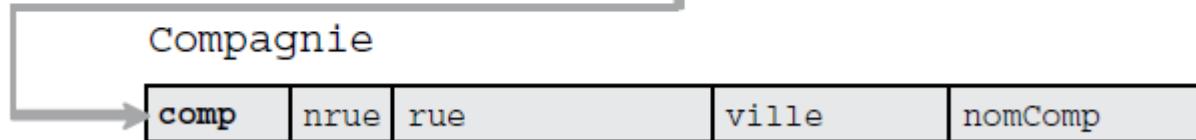
<i>nom</i>	<i>prénom</i>
Dupond	Marc
Dupont	Pierre
Martin	Marc
Martin	Pierre
Martin	Marc

enregistrement interdit car le couple ('Martin', 'Marc') est un doublon du couple (nom,prénom)



Exemple

brevet	nom	nbHVol	compa



Tables

```
CREATE TABLE Compagnie
(comp CHAR(4), nrule INTEGER(3),
rue CHAR(20), ville CHAR(15) DEFAULT 'Paris',
COMMENT 'Par defaut : Paris',
nomComp CHAR(15) NOT NULL,
CONSTRAINT pk_Compagnie PRIMARY KEY(comp));
```

```
CREATE TABLE Pilote
(brevet CHAR(6), nom CHAR(15) NOT NULL,
nbHVol DECIMAL(7,2), compa CHAR(4),
CONSTRAINT pk_Pilote PRIMARY KEY(brevet),
CONSTRAINT ck_nbHVol
    CHECK(nbHVol BETWEEN 0 AND 20000),
CONSTRAINT un_nom UNIQUE (nom),
CONSTRAINT fk_Pil_compa_Comp
    FOREIGN KEY (compa)
        REFERENCES Compagnie(comp));
```

Contraintes

Deux contraintes en ligne
et une contrainte nommée
de clé primaire.

Une contrainte en ligne
et quatre contraintes nommées :

- Clé primaire
- NOT NULL
- CHECK (nombre d'heures de vol
compris entre 0 et 20 000)
- UNIQUE (homonymes interdits)
- Clé étrangère

Types de données

Les types de colonnes

- caractères (CHAR, VARCHAR, TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT) ;
- valeurs numériques (TINYINT, SMALLINT, MEDIUMINT, INT, INTEGER, BIGINT, FLOAT, DOUBLE, REAL, DECIMAL, NUMERIC, et BIT) ;
- date/heure (DATE, DATETIME, TIME, YEAR, TIMESTAMP) ;
- données binaires (BLOB, TINYBLOB, MEDIUMBLOB, LONGBLOB) ;
- énumérations (ENUM, SET).

Le type caractère

Type	Description	Commentaire pour une colonne
CHAR (n) [BINARY ASCII UNICODE]	Chaîne fixe de n octets ou caractères.	Taille fixe (maximum de 255 caractères).
VARCHAR (n) [BINARY]	Chaîne variable de n caractères ou octets.	Taille variable (maximum de 65 535 caractères).
BINARY (n)	Chaîne fixe de n octets.	Taille fixe (maximum de 255 octets).
VARBINARY (n)	Chaîne variable de n octets.	Taille variable (maximum de 255 octets).
TINYTEXT (n)	Flot de n octets.	Taille fixe (maximum de 255 octets).
TEXT (n)	Flot de n octets.	Taille fixe (maximum de 65 535 octets).
MEDIUMTEXT (n)	Flot de n octets.	Taille fixe (maximum de 16 mégaoctets).
LONGTEXT (n)	Flot de n octets.	Taille fixe (maximum de 4,29 gigaoctets).

Le type numérique

Type	Description
BIT [(n)]	Ensemble de n bits. Taille de 1 à 64 (par défaut 1).
TINYINT [(n)] [UNSIGNED] [ZEROFILL]	Entier (sur un octet) de -128 à 127 signé, 0 à 255 non signé.
BOOL et BOOLEAN	Synonymes de TINYINT(1), la valeur zéro est considérée comme fausse. Le non-zéro est considéré comme vrai. Dans les prochaines versions, le type boolean, comme le préconise la norme SQL, sera réellement pris en charge.
SMALLINT [(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 2 octets) de -32 768 à 32 767 signé, 0 à 65 535 non signé.
MEDIUMINT [(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 3 octets) de -8 388 608 à 8 388 607 signé, 0 à 16 777 215 non signé.
INTEGER [(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 4 octets) de -2 147 483 648 à 2 147 483 647 signé, 0 à 4 294 967 295 non signé.
BIGINT [(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 8 octets) de -9 223 372 036 854 775 808 à 9 223 372 036 854 775 807 signé, 0 à 18 446 744 073 709 551 815 non signé.
FLOAT [(n[,p])] [UNSIGNED] [ZEROFILL]	Flottant (de 4 à 8 octets) p désigne la précision simple (jusqu'à 7 décimales) de $-3.4 \cdot 10^{-38}$ à $-1.1 \cdot 10^{-38}$, 0, signé, et de $1.1 \cdot 10^{-38}$ à $3.4 \cdot 10^{-38}$ non signé.
DOUBLE [(n[,p])] [UNSIGNED] [ZEROFILL]	Flottant (sur 8 octets) p désigne la précision double (jusqu'à 15 décimales) de $-1.7 \cdot 10^{-308}$ à $-2.2 \cdot 10^{-308}$, 0, signé, et de $2.2 \cdot 10^{-308}$ à $1.7 \cdot 10^{-308}$ non signé.
DECIMAL [(n[,p])] [UNSIGNED] [ZEROFILL]	Décimal à virgule fixe, p désigne la précision (nombre de chiffres après la virgule, maximum 30). Par défaut n vaut 10, p vaut 0.

Le type de données dates et heures

Type	Description	Commentaire pour une colonne
DATE	Dates du 1 ^{er} janvier de l'an 1000 au 31 décembre 9999 après J.-C.	Sur 3 octets. L'affichage est au format 'YYYY-MM-DD'.
DATETIME	Dates et heures (de 0 h de la première date à 23 h 59 minutes 59 secondes de la dernière date).	Sur 8 octets. L'affichage est au format 'YYYY-MM-DD HH:MM:SS'.
YEAR [(2 4)]	Sur 4 positions : de 1901 à 2155 (incluant 0000). Sur 2 positions : de 70 à 69 (désignant 1970 à 2069).	Sur 1 octet ; l'année est considérée sur 2 ou 4 positions (4 par défaut). Le format d'affichage est 'YYYY'.
TIME	Heures de -838 h 59 minutes 59 secondes à 838 h 59 minutes 59 secondes.	L'heure au format 'HHH:MM:SS' sur 3 octets.
TIMESTAMP	Instants du 1 ^{er} Janvier 1970 0 h 0 minute 0 seconde à l'année 2037.	Estampille sur 4 octets (au format 'YYYY-MM-DD HH:MM:SS') ; mise à jour à chaque modification sur la table.

Le type binaire et énumération

Type	Description	Commentaire pour une colonne
TINYBLOB (<i>n</i>)		Taille fixe (maximum de 255 octets).
BLOB (<i>n</i>)		Taille fixe (maximum de 65 535 octets).
MEDIUMBLOB (<i>n</i>)	Flot de <i>n</i> octets.	Taille fixe (maximum de 16 mégaoctets).
LONGBLOB (<i>n</i>)		Taille fixe (maximum de 4,29 gigaoctets).

Type	Description
ENUM ('valeur1', 'valeur2', ...)	Liste de 65 535 valeurs au maximum.
SET ('valeur1', 'valeur2', ...)	Ensemble de référence (maximum de 64 valeurs).

Structure des tables

Résultat						Commentaires																																			
mysql> DESCRIBE Pilote;						Les clés primaires sont NOT NULL (désignées par PRI dans la colonne Key).																																			
<table border="1"> <thead> <tr> <th>Field</th><th>Type</th><th>Null</th><th>Key</th><th>Default</th><th>Extra</th></tr> </thead> <tbody> <tr> <td>brevet</td><td>char(6)</td><td>NO</td><td>PRI</td><td></td><td></td></tr> <tr> <td>nom</td><td>char(15)</td><td>YES</td><td>UNI</td><td>NULL</td><td></td></tr> <tr> <td>nbHVol</td><td>double(7,2)</td><td>YES</td><td></td><td>NULL</td><td></td></tr> <tr> <td>compa</td><td>char(4)</td><td>YES</td><td>MUL</td><td>NULL</td><td></td></tr> </tbody> </table>						Field	Type	Null	Key	Default	Extra	brevet	char(6)	NO	PRI			nom	char(15)	YES	UNI	NULL		nbHVol	double(7,2)	YES		NULL		compa	char(4)	YES	MUL	NULL							
Field	Type	Null	Key	Default	Extra																																				
brevet	char(6)	NO	PRI																																						
nom	char(15)	YES	UNI	NULL																																					
nbHVol	double(7,2)	YES		NULL																																					
compa	char(4)	YES	MUL	NULL																																					
						Les unicités sont désignées par UNI dans la colonne Key.																																			
						Les occurrences multiples possibles sont désignées par MUL dans la colonne Key.																																			
mysql> DESCRIBE Compagnie;						Les contraintes NOT NULL nommées (définies via les contraintes CHECK) n'apparaissent pas.																																			
<table border="1"> <thead> <tr> <th>Field</th><th>Type</th><th>Null</th><th>Key</th><th>Default</th><th>Extra</th></tr> </thead> <tbody> <tr> <td>comp</td><td>char(4)</td><td>NO</td><td>PRI</td><td></td><td></td></tr> <tr> <td>nrue</td><td>int(3)</td><td>YES</td><td></td><td>NULL</td><td></td></tr> <tr> <td>rue</td><td>char(20)</td><td>YES</td><td></td><td>NULL</td><td></td></tr> <tr> <td>ville</td><td>char(15)</td><td>YES</td><td></td><td>Paris</td><td></td></tr> <tr> <td>nomComp</td><td>char(15)</td><td>NO</td><td></td><td></td><td></td></tr> </tbody> </table>						Field	Type	Null	Key	Default	Extra	comp	char(4)	NO	PRI			nrue	int(3)	YES		NULL		rue	char(20)	YES		NULL		ville	char(15)	YES		Paris		nomComp	char(15)	NO			
Field	Type	Null	Key	Default	Extra																																				
comp	char(4)	NO	PRI																																						
nrue	int(3)	YES		NULL																																					
rue	char(20)	YES		NULL																																					
ville	char(15)	YES		Paris																																					
nomComp	char(15)	NO																																							
						La colonne Extra indique notamment les séquences (AUTO_INCREMENT).																																			

Effacer une table

- La commande **DROP TABLE** prend en paramètre le nom de la table à supprimer. Toutes les données qu'elle contient sont supprimées et sa définition aussi.
- Syntaxe :
 - **DROP TABLE *relation***
- Exemple :
 - **DROP TABLE *Personnes***
- Si un beau jour on s'aperçoit qu'une relation a été mal définie au départ, plutôt que de la supprimer et de la reconstruire bien comme il faut, on peut la modifier très simplement. Cela évite de perdre les données qu'elle contient.

Exemple de session avec MySQL

Exemple d'une session avec MySQL

- Création de la base de données

```
mysql> CREATE DATABASE mydb;  
Query OK, 1 row affected (0.01 sec)
```

- Utilisation de la base de données

```
mysql> USE mydb;  
Database changed
```

Effacer une base de données

```
mysql> DROP DATABASE temp_db;  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> DROP DATABASE IF EXISTS temp_db;  
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
mysql> SHOW DATABASES;
```

Database
information_schema
mydb
mysql
test

```
+-----+  
| Database      |  
+-----+  
| information_schema |  
| mydb           |  
| mysql          |  
| test           |  
+-----+  
4 rows in set (0.00 sec)
```

Créer une table

```
mysql> CREATE TABLE person (
-> person_id SMALLINT UNSIGNED NOT NULL,
-> first_name VARCHAR(45) NOT NULL,
-> last_name VARCHAR(45) NOT NULL,
-> PRIMARY KEY (person_id)
-> ) ENGINE=InnoDB;
```

Query OK, 0 rows affected (0.14 sec)

```
mysql> SHOW TABLES;
+-----+
| Tables_in_mydb |
+-----+
| person        |
+-----+
1 row in set (0.00 sec)
```

Modifier le nom d'une table

```
mysql> ALTER TABLE person rename to person1;  
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> SHOW TABLES;  
+-----+  
| Tables_in_mydb |  
+-----+  
| person1 |  
+-----+  
1 row in set (0.00 sec)
```

```
mysql> RENAME TABLE person1 TO whatever;  
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> SHOW TABLES;  
+-----+  
| Tables_in_mydb |  
+-----+  
| whatever |  
+-----+
```

Modifier le nom d'un champ

```
mysql> ALTER TABLE person CHANGE last_name surname varchar(30);
Query OK, 0 rows affected (0.62 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> DESCRIBE person;
```

Field	Type	Null	Key	Default	Extra
person_id	smallint(5) unsigned	NO	PRI	NULL	
first_name	varchar(45)	NO		NULL	
surname	varchar(30)	YES		NULL	

3 rows in set (0.01 sec)

Ajouter ou effacer un champ

```
mysql> ALTER TABLE person ADD age smallint(3) unsigned not null;  
Query OK, 0 rows affected (0.42 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> DESCRIBE person;
```

Field	Type	Null	Key	Default	Extra
person_id	smallint(5) unsigned	NO	PRI	NULL	
first_name	varchar(45)	NO		NULL	
surname	varchar(30)	YES		NULL	
age	smallint(3) unsigned	NO		NULL	

```
4 rows in set (0.01 sec)
```

```
mysql> ALTER TABLE person DROP first_name;  
Query OK, 0 rows affected (0.25 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

Effacer des tables

```
mysql> SHOW TABLES;
```

Tables_in_temp_db
temp_table

1 row in set (0.00 sec)

```
mysql> DROP TABLE temp_table;
```

Query OK, 0 rows affected (0.06 sec)

```
mysql> DROP TABLE IF EXISTS temp_table;
```

Query OK, 0 rows affected, 1 warning (0.12 sec)

```
mysql> SHOW TABLES;
```

Empty set (0.00 sec)

Travailler avec des tables de plusieurs bases de données

```
mysql> SELECT * FROM temp_db.temp_table;
```

temp_id	temp_whatever
1	life is good
2	life is even better

2 rows in set (0.00 sec)

```
mysql> SELECT * FROM mydb.student;
```

student_id	first_name	last_name	age	grade
1	yuna	kim	19	4
2	kelly	jones	22	5

2 rows in set (0.00 sec)

Insérer un enregistrement

```
mysql> INSERT INTO person (person_id, first_name, last_name, age)
-> VALUES (1, 'sang', 'shin', 88);
Query OK, 1 row affected (0.10 sec)
```

```
mysql> SELECT * FROM person;
+-----+-----+-----+
| person_id | first_name | last_name | age |
+-----+-----+-----+
|      1 | sang      | shin     | 88   |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Insérer plusieurs enregistrements

```
mysql> INSERT INTO person (person_id, first_name, last_name, age)
-> VALUES
-> (2, 'kelly', 'jones', 22),
-> (3, 'jack', 'kennedy', 56),
-> (4, 'paul', 'kennedy', 34),
-> (5, 'daniel', 'song', 24),
-> (6, 'nichole', 'scott', 9);
```

Query OK, 3 rows affected (0.05 sec)

Records: 3 Duplicates: 0 Warnings: 0

- Effacer plusieurs enregistrements

```
mysql> DELETE FROM person WHERE age < 10;  
Query OK, 1 row affected (0.07 sec)
```

Mettre à jour des enregistrements

```
mysql> UPDATE person SET age = 88  
      -> WHERE age = 99 OR first_name = 'paul';  
Query OK, 1 row affected, 2 warnings (0.04 sec)  
Rows matched: 1  Changed: 1  Warnings: 2
```

```
mysql> SELECT * FROM person;  
+-----+-----+-----+---+  
| person_id | first_name | last_name | age |  
+-----+-----+-----+---+  
| 1 | sang | shin | 88 |  
| 2 | kelly | jones | 22 |  
| 3 | jack | kennedy | 56 |  
| 4 | paul | kennedy | 88 |  
+-----+-----+-----+---+  
4 rows in set (0.00 sec)
```

Retrouver la valeur de certains champs

```
mysql> SELECT last_name, age FROM person;
```

last_name	age
shin	88
jones	22
kennedy	56
kennedy	34
song	24

```
5 rows in set (0.00 sec)
```

Rechercher des informations avec la clause WHERE

```
mysql> SELECT first_name, age FROM person  
-> WHERE age > 50;
```

first_name	age
sang	88
jack	56

2 rows in set (0.00 sec)

```
mysql> SELECT first_name, last_name, age FROM person  
-> WHERE age < 50 AND first_name LIKE '%niel';
```

first_name	last_name	age
daniel	song	24

Retrouver des enregistrements triés

```
mysql> SELECT last_name, age FROM person  
-> ORDER BY age ASC;
```

last_name	age
jones	22
song	24
kennedy	34
kennedy	56
shin	88

5 rows in set (0.00 sec)

```
mysql> SELECT * FROM person  
-> ORDER BY age DESC;
```

person_id	first_name	last_name	age
1	sang	shin	88
3	jack	kennedy	56
4	paul	kennedy	34
5	daniel	song	24
2	kelly	jones	22

Retrouver un nombre d'enregistrements limité

```
mysql> SELECT * from person  
-> ORDER BY age DESC  
-> LIMIT 3;
```

person_id	first_name	last_name	age
1	sang	shin	88
3	jack	kennedy	56
4	paul	kennedy	34

3 rows in set (0.00 sec)

Faire des opérations arithmétiques

```
mysql> SELECT 3 + 6;
```

```
+-----+
| 3 + 6 |
+-----+
|    9 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT 45 * (1+2);
```

```
+-----+
| 45 * (1+2) |
+-----+
|      135 |
+-----+
1 row in set (0.00 sec)
```

COUNT, AVG, SUM

```
mysql> SELECT COUNT(age) FROM person;
+-----+
| COUNT(age) |
+-----+
|      5 |
+-----+
1 row in set (0.04 sec)
```

```
mysql> SELECT AVG(age) from person;
+-----+
| AVG(age) |
+-----+
| 44.8000 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT SUM(age) FROM person;
+-----+
| SUM(age) |
+-----+
|    224 |
+-----+
1 row in set (0.00 sec)
```

MIN, MAX

```
mysql> SELECT MIN(age) FROM person;
```

MIN(age)
22

```
1 row in set (0.00 sec)
```

```
mysql> SELECT MAX(age) FROM person;
```

MAX(age)
88

```
1 row in set (0.00 sec)
```

Exécuter un script SQL

```
mysql> SOURCE c:/tmp/student.sql  
Query OK, 0 rows affected (0.10 sec)
```