

# 4TPM207U - Système de Gestion de Base de Données (Coloration informatique)

- Équipe enseignante
  - ✓ Nicolas Bonichon
  - ✓ Alain Griffault
  - ✓ Vincent Lepetit
  - ✓ Bruno Pinaud (responsable)
  - ✓ Maria Predari
- Tout est sur la plateforme Moodle sciences et technologie (moodle1).
- Le cours apparaît déjà dans votre liste de cours.
- Pas de courrier électronique. Utilisez la plateforme Moodle. Attention, tous les inscrits recevront le message.
- Pour chaque partie du cours, un forum dédié permet de poser des questions.

La note de contrôle continu (coef 0.6) est composé d'un DS prévu en semaine 10 (coef 0.3) et d'une évaluation du projet qui sera réalisé en TD machines (coef 0.3 -- modalités à définir).



[Posez vos questions sur l'UE ici.](#)



[Support de cours](#) Document PDF Déposé le 24 janv. 17, 15:36

Support des deux premiers cours.



[Posez vos questions sur le cours de présentation ici.](#)

# *Généralités sur l'UE*

- **En quelques mots**

- ✓ C2i  $\Rightarrow$  Savoir utiliser les sites et outils web modernes (Amazon, Facebook, ...) correctement
- ✓ Cette UE : comprendre les grands principes de fonctionnement de tels sites sur un choix restreint de technologies (Python notamment).
- ✓ Faire une petite application web dynamique à partir de données réelles.

- **Évaluations**

- ✓ Contrôles continus (0,6) : DS en amphi en S10 (0,3) et un projet noté sur machines (0,3)
- ✓ Examen terminal (0,4) en amphi qui reprendra les parties théoriques et pratiques
- ✓ Correction des devoirs et examen par l'enseignant de chaque groupe.

# *Application web*

# *Application web : Success stories*

- **YouTube**

- Fondé en 2005 par trois anciens de PayPal en une nuit
- Revendu à Google 1,65 milliards de dollars en 2006

- **Wikipedia**

- Créé en 2001
- Modèle participatif
- En 2014 recense plus de 10 millions d'articles en 200 langues

# *Application web*

## **Définition générale**

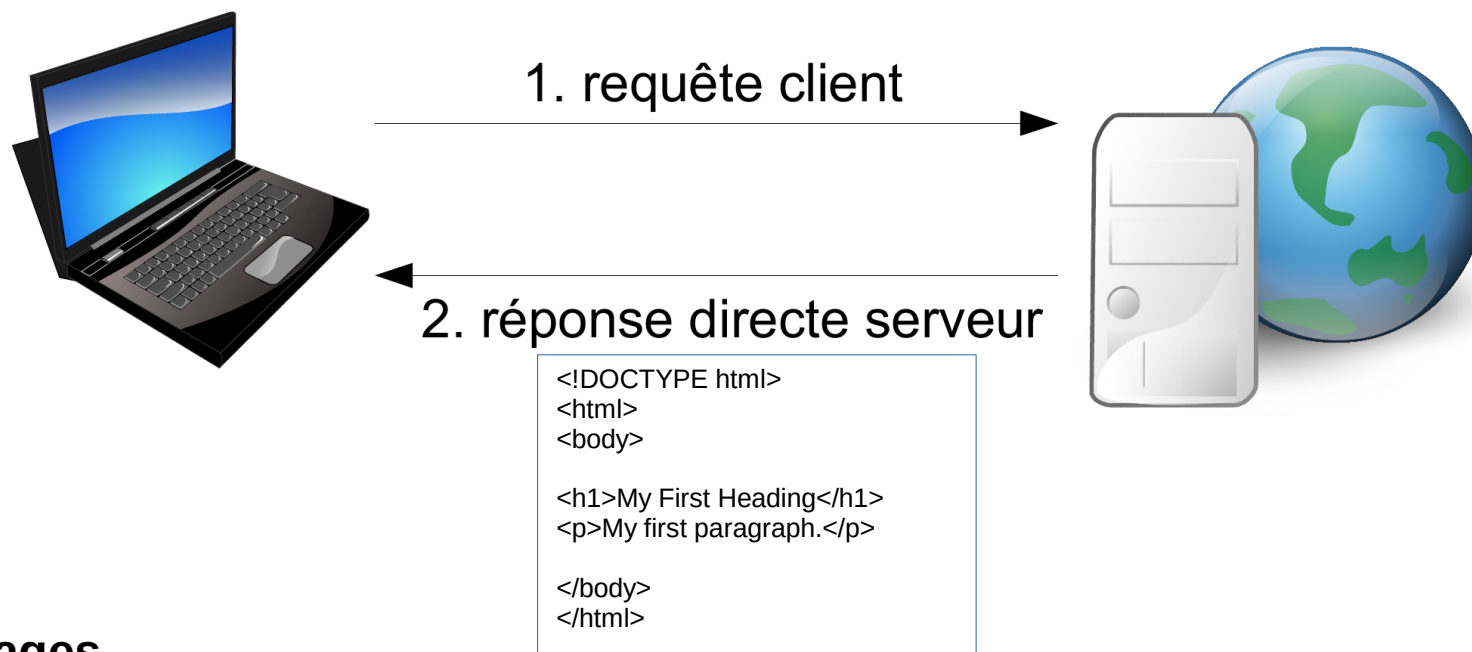
- Application client-serveur qui s'exécute sur le Web en général via un navigateur web
- Ensemble de codes sources hétérogènes : HTML, code côté client, code côté serveur
- Avantages : déploiement facile, utilise peu de ressources côté client
- Inconvénients : interfaces pauvres, nécessite une infrastructure serveur solide, ne fonctionne pas sans réseau ou serveur

# *Les styles d'applications Web*

**Pour ce cours, on retient 3 types**

1. Les applications statiques (site web basique)
2. Les applications dynamiques côté serveur
3. Les applications dynamiques côté serveur et client

# Applications statiques



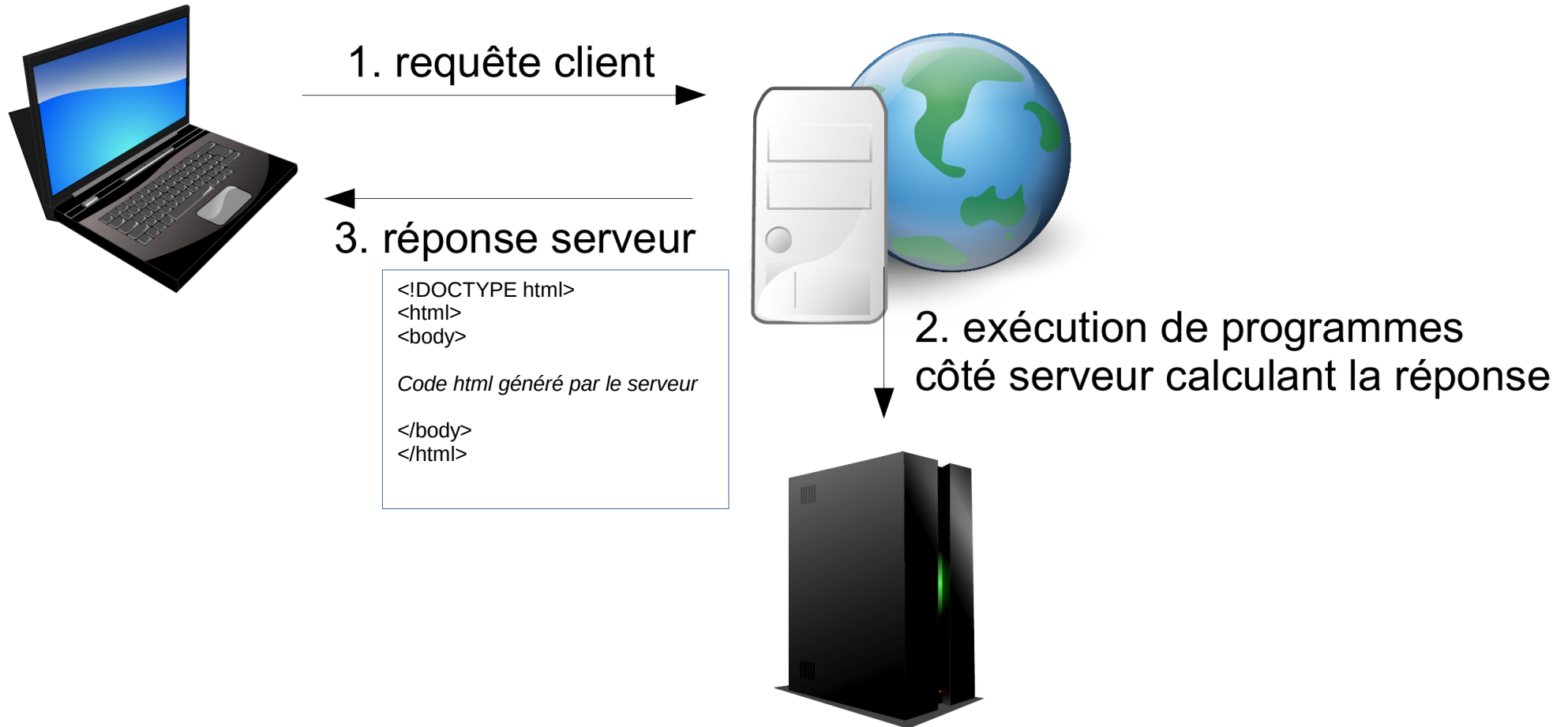
## Avantages

- Facile à mettre en œuvre (uniquement du HTML (et CSS))
- Maintenance minimale

## Inconvénients

- Interface sommaire
- Contenu fastidieux à produire et maintenir
- Prochainement : TD machines sur ce type d'application (base du HTML)

# Applications dynamiques serveur





# *Applications dynamiques serveur*

## **Avantages**

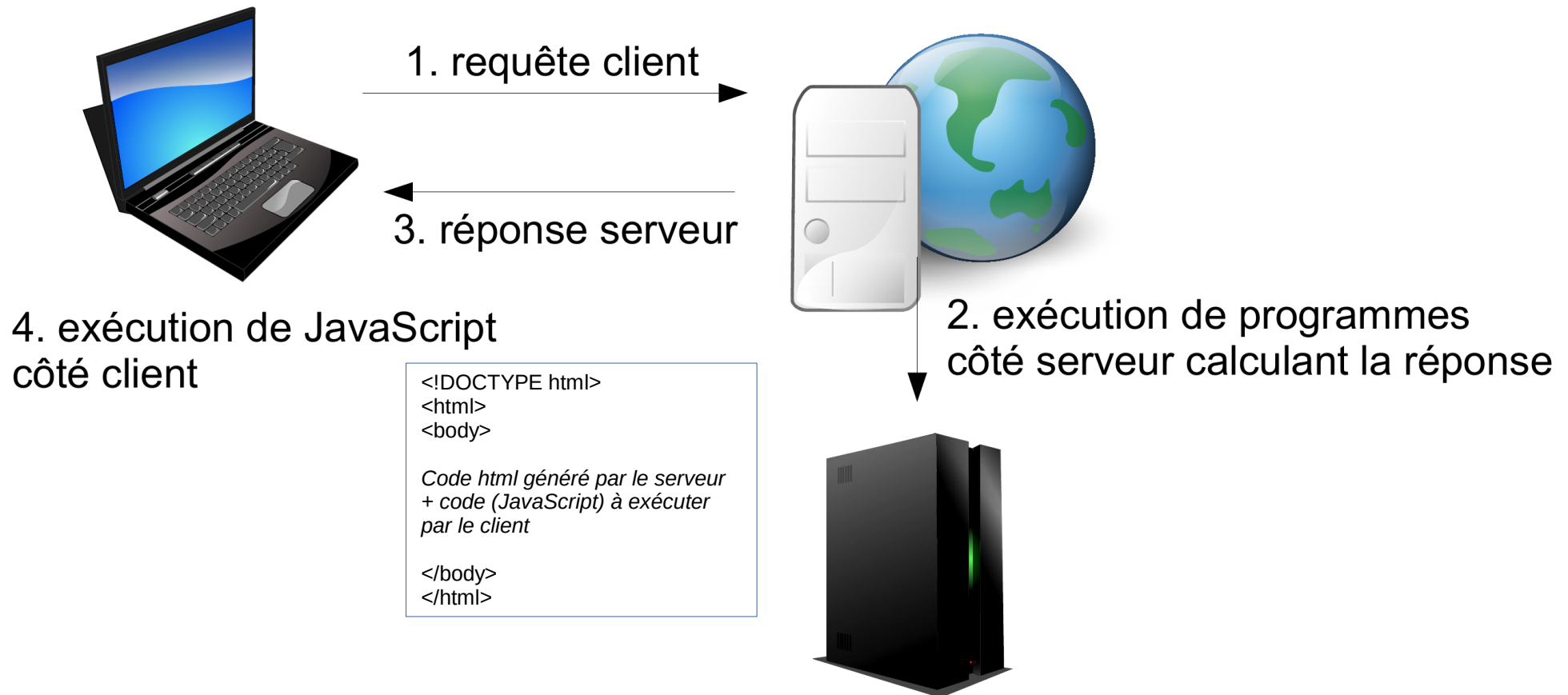
- Séparation données - présentation
- Production de contenu beaucoup plus rapide

## **Inconvénients**

- Interface sommaire
- Charge serveur élevée
- Nécessite de connaître un langage de programmation supplémentaire (par ex. PHP)

**Prochainement** : TD machines sur ce type d'application (Python côté serveur)

# Application dynamique client/serveur



# *Application dynamique client/serveur*

## **Avantages**

- Séparation données - présentation
- Production de contenu beaucoup plus rapide
- Interface plus riche que précédemment

## **Inconvénients**

- Charge serveur élevée
- Charge client élevée
- Nécessite de connaître deux langages de programmation (un côté serveur et un autre côté client, souvent javascript) supplémentaires

# *Application web : synthèse*

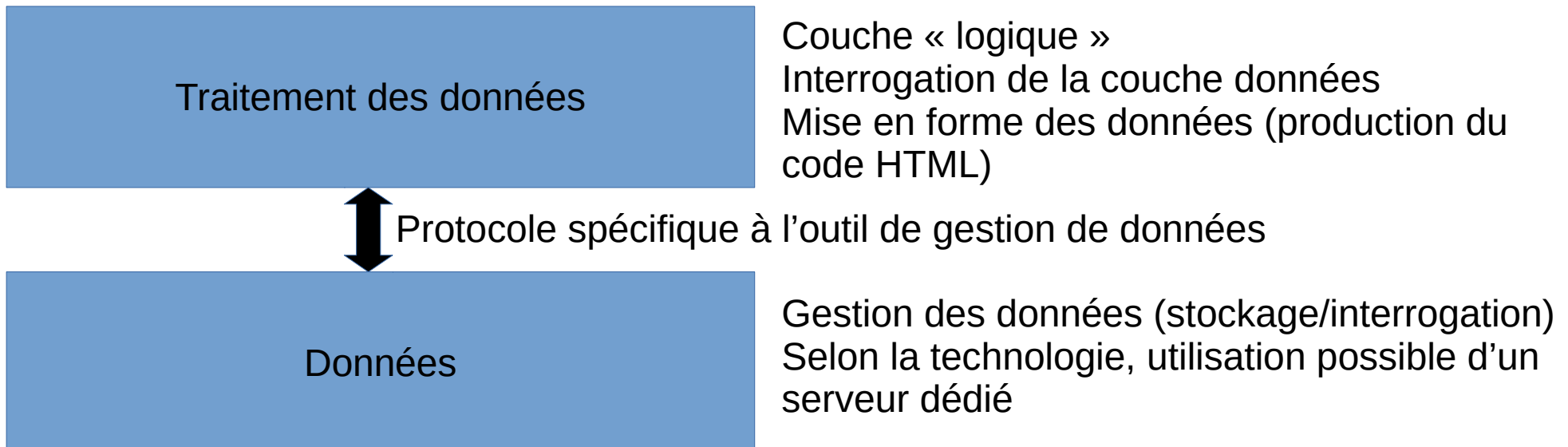
**Application dynamique côté serveur (ou architecture 3-tiers)**

Données

Gestion des données (stockage/interrogation)  
Selon la technologie, utilisation possible d'un serveur dédié

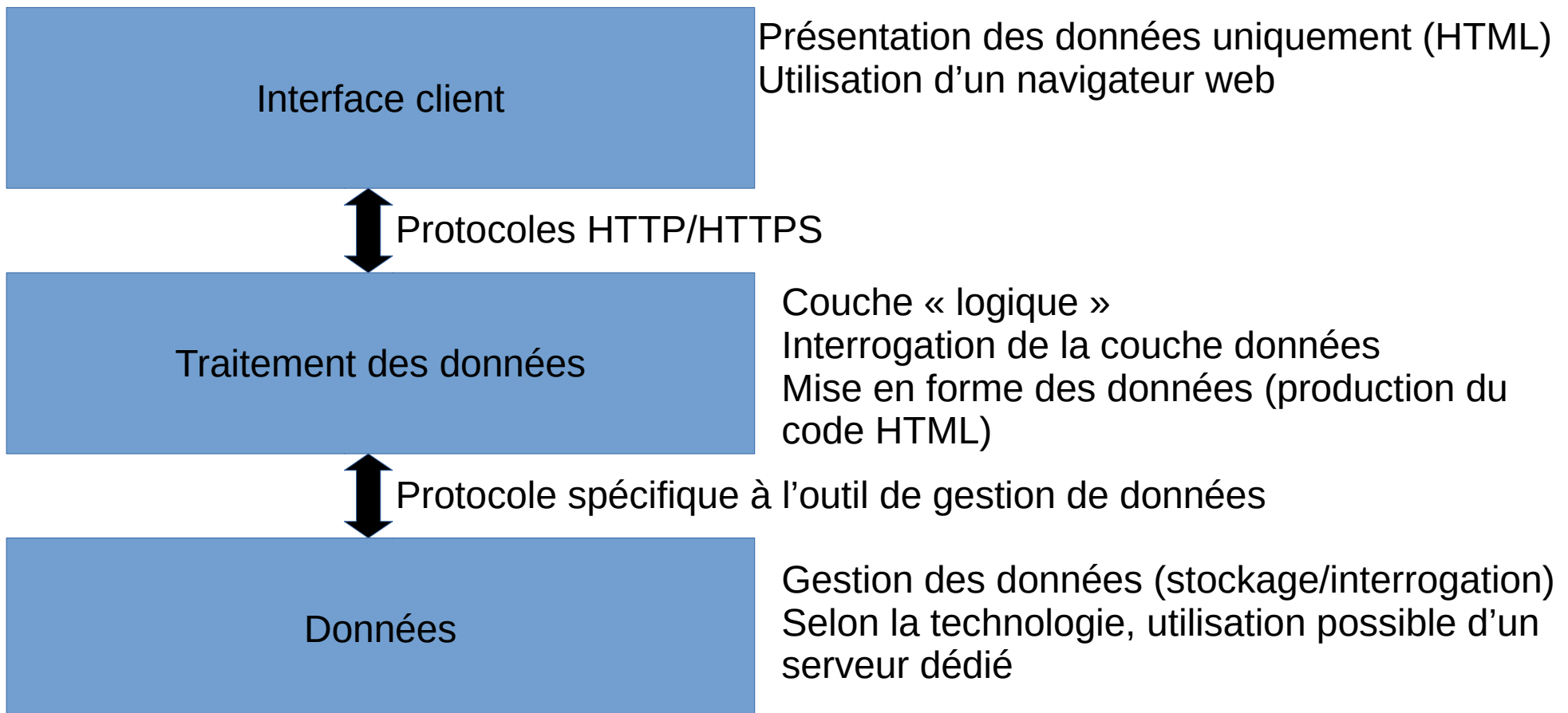
# *Application web : synthèse*

## **Application dynamique côté serveur (ou architecture 3-tiers)**



# Application web : synthèse

## Application dynamique côté serveur (ou architecture 3-tiers)



# *La couche données*

- Les très répandues **Bases de Données Relationnelles**
- Les bases de données « NoSQL »

# *Base de données relationnelles*

- Plus qu'un simple ensemble de données non-indépendantes
- « [...] Ensemble de données modélisant les objets d'une partie du monde réel et servant de support à une application informatique » G. Gardarin

## **Propriété importante**

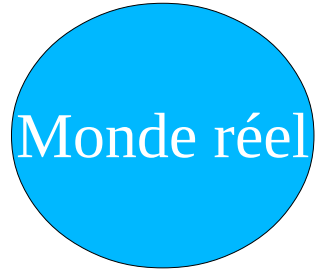
interrogeable par le contenu selon n'importe quel critère

Exemple : quels sont les numéros de Radioactive Man à moins de 10\$ ?



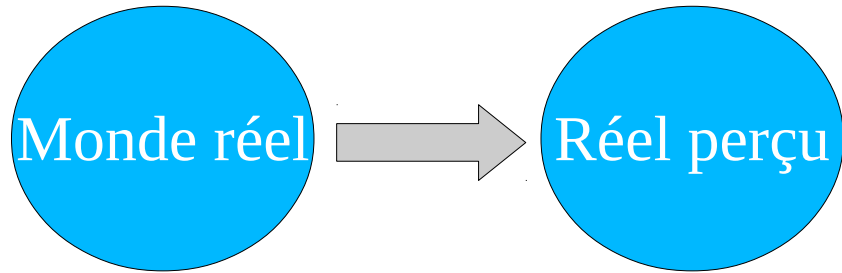
# *Introduction*

## *Processus de construction d'une base de données*



# *Introduction*

## *Processus de construction d'une base de données*



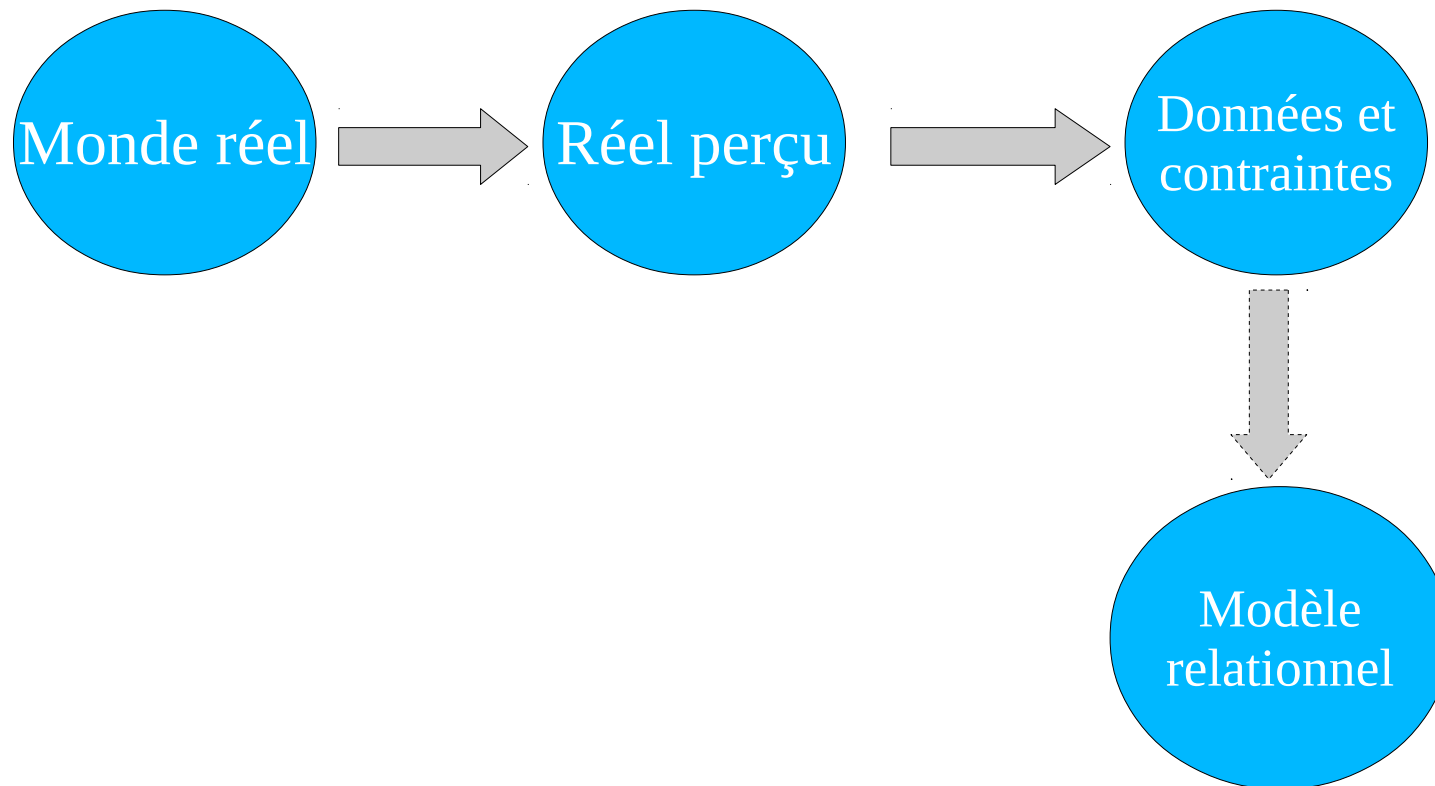
# *Introduction*

## *Processus de construction d'une base de données*



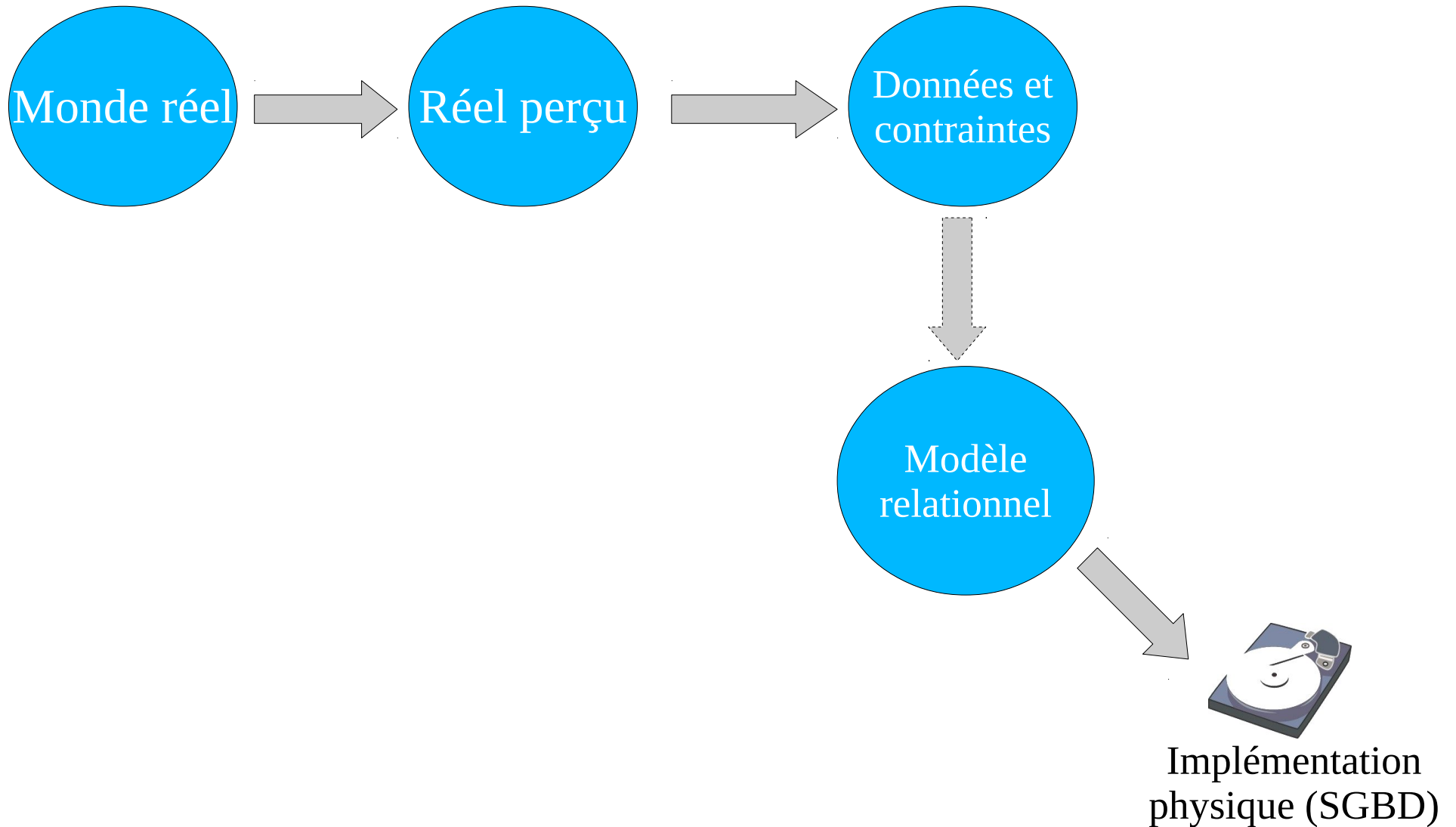
# *Introduction*

## *Processus de construction d'une base de données*



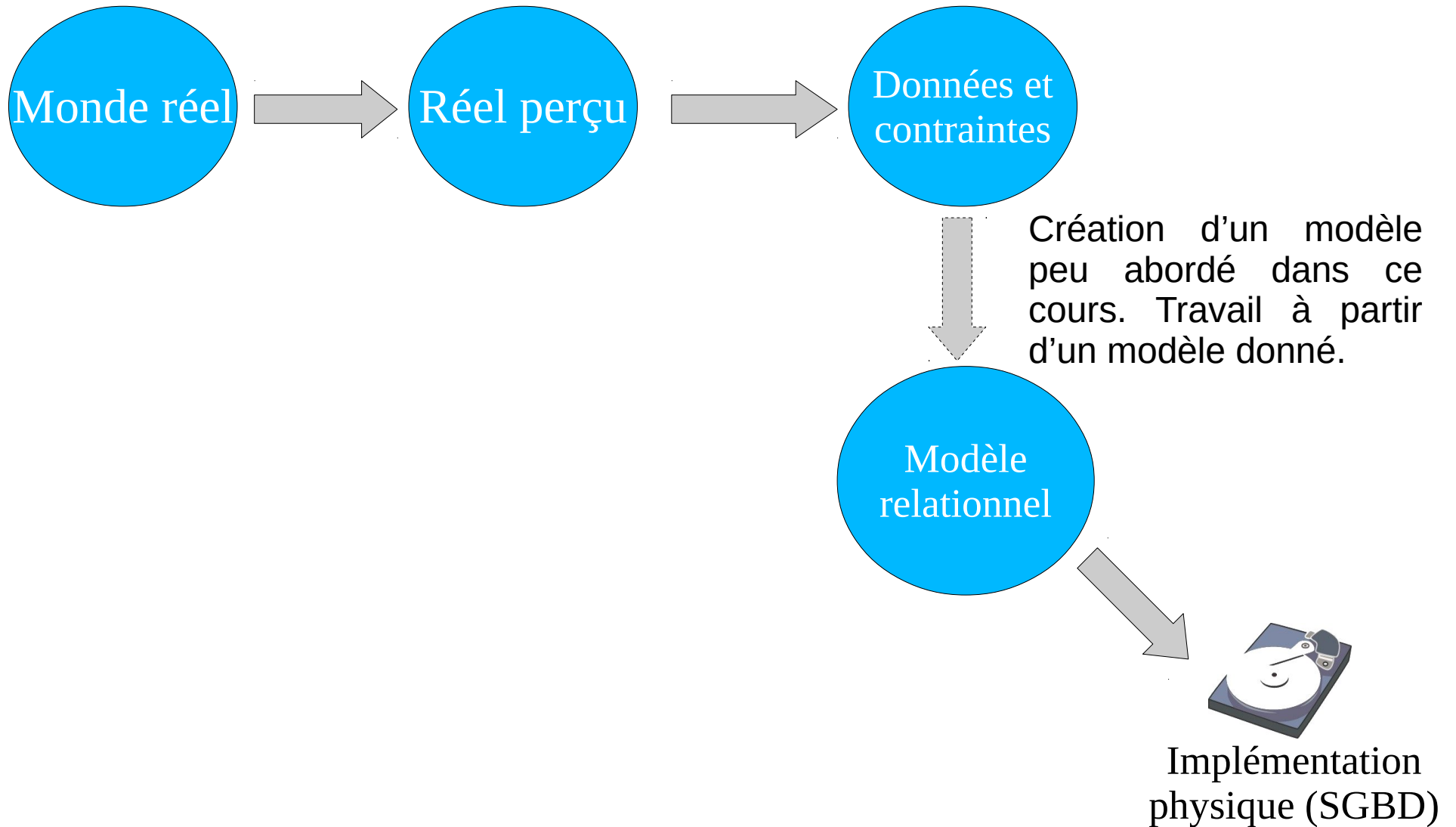
# *Introduction*

## *Processus de construction d'une base de données*



# Introduction

## Processus de construction d'une base de données



## *Définition : SGBD*

- **SGBD** : Système de Gestion de Bases de Données (parfois SGBDR)
- **DBMS** : DataBase Management System (parfois RDBMS)
- Outil informatique pour gérer des BD.

# Définition : SGBD Relationnel

- **SGBD** : Système de Gestion de Bases de Données (parfois SGBDR)
- **DBMS** : DataBase Management System (parfois RDBMS)
- Outil informatique pour gérer des BD.
- **Doit absolument posséder plusieurs fonctionnalités**
  - ✓ Sauvegarde (persistance) des données
  - ✓ Interrogation (SQL) des données
  - ✓ Recherche et mise en forme des données stockées
  - ✓ Partage des données entre les différents utilisateurs
  - ✓ Gestion de la concurrence d'accès
  - ✓ Sécurité des données (gestion des incidents)
  - ✓ Optimisation des opérations dans un souci constant de performance



# *Définition : SGBD Relationnel*

Les principales opérations (LDD et LMD, SQL-92) :

- Création d'une relation (create table)
- Suppression d'une relation (drop table)
- Modification d'une relation (alter table)

# Définition : SGBD Relationnel

Les principales opérations (LDD et LMD, SQL-92) :

- Création d'une relation (*create table*)
- Suppression d'une relation (*drop table*)
- Modification d'une relation (*alter table*)
  
- Interrogation (*select*)
- Insertion (*insert*)
- Mise à jour (*update*)
- Suppression (*delete*)

# ***BD : Vocabulaires et définitions***

# *Concepts de base : domaine*

## **Définition**

- Ensemble de valeurs atomiques (non décomposable)
- Équivalent au typage en programmation

## **Exemples**

entier, réel, caractère

booléen =  $\{0,1\}$

numéro INSEE (bien qu'il soit composé de plusieurs champs)

# Concepts de base : produit cartésien

## Définition

Le produit cartésien d'un ensemble de domaine  $D_1, D_2, \dots, D_n$  noté  $D_1 \times D_2 \times \dots \times D_n$  est l'ensemble des n-uplets (tuples)  $\langle v_1, v_2, \dots, v_n \rangle$  tels que  $v_i \in D_i$ .

## Exemple

$$D_1 = \{0, 1\}$$

$$D_2 = \{\text{bleu, blanc, rouge}\}$$

$$D_2 \times D_1$$

Bleu	0
Blanc	0
Rouge	0
Bleu	1
Blanc	1
Rouge	1

# Concepts de base : relation

## Définition

- Sous ensemble  $r$  du produit cartésien d'un ensemble de domaines
- Caractérisée par un nom

## Exemple

$D_1 = \{\text{beignets, bonbons, salade}\}$

$D_2 = \{\text{Lisa, Bart, Homer}\}$

$D_2 \times D_1$

Lisa	beignets
Lisa	bonbons
Lisa	salade
Bart	beignets
Bart	bonbons
Bart	salade
Homer	beignets
Homer	bonbons
Homer	salade

## RELATION

Aliments préférés

Lisa	Salade
Bart	Bonbons
Homer	Beignets

# Concepts de base : relation

## Définition

- Sous ensemble  $r$  du produit cartésien d'un ensemble de domaines
- Caractérisée par un nom

## Exemple

$D_1 = \{\text{beignets, bonbons, salade}\}$

$D_2 = \{\text{Lisa, Bart, Homer}\}$

Aliments préférés

Lisa	Salade
Bart	Bonbons
Homer	Beignets

colonne  $\subset$  domaine

# Concepts de base : relation

## Définition

- Sous ensemble  $r$  du produit cartésien d'un ensemble de domaines
- Caractérisée par un nom

## Exemple

$D_1 = \{\text{beignets, bonbons, salade}\}$

$D_2 = \{\text{Lisa, Bart, Homer}\}$

Aliments préférés

Lisa	Salade
Bart	Bonbons
Homer	Beignets

Cardinalité  
=3

$n$ -uplet (ou tuple en anglais)



# Concepts de base : attribut

## Définition

- Nom donné à une colonne
- Composé d'un identifiant et d'un domaine
- Nombre d'attributs d'une relation = degré de la relation (ou arité)

## Exemple

Aliments préférés

Personne	Aliment
Lisa	Salade
Bart	Bonbons
Homer	Beignets

Degré = 2

# Concepts de base : schéma de relation

## Définition

- Un schéma de relation **R** est défini par un ensemble d'attributs **U** et un ensemble de contraintes.
- On le note couramment **R(U)**.
- Le schéma **R(U)** (attribut et contraintes) décrit l'**intention** de la relation.
- La relation (tableau avec les valeurs) définit une **extension**.
- Une relation **r** est une instance finie d'un schéma de relation, notée **r:R(U)**.

## Exemples de schéma de relation

Aliments préférés (nom, type, origine, bio)

RégimeAlimentaire (aliment, calories)

# Concepts de base : base de données

## Définition

- Un **schéma de base de données** est un ensemble de schémas de relations **liés** par des **dépendances référentielles (un type de contraintes)** : attributs communs ou plus généralement des dépendances d'inclusion.
- Une **base de données** est alors un ensemble de relations (extensions) associé au schéma de base de données et vérifiant toutes ses **contraintes**.

# Un mauvais exemple

**Objectif** : Modélisation d'une compagnie aérienne.

On veut savoir quel est l'avion utilisé pour chaque ligne et sa capacité.

Création d'une **relation universelle** avec tous les attributs

## Transports

Vol	Avion	Capacité
IT5033	Airbus A330	335
AF2401	Airbus A330	335
AF2409	Boeing 727	150
IT5133	Airbus A330	335
IT5035	Canadair	90
AF2802	Airbus A330	335

# *Un mauvais exemple*

## **Transports**

Vol	Avion	Capacité
IT5033	Airbus A330	335
AF2401	Airbus A330	335
AF2409	Boeing 727	150
IT5133	Airbus A330	335
IT5035	Canadair	90
AF2802	Airbus A330	335

Anomalies de la relation  
**Transports**

# *Un mauvais exemple*

## **Transports**

Vol	Avion	Capacité
IT5033	Airbus A330	335
AF2401	Airbus A330	335
AF2409	Boeing 727	150
IT5133	Airbus A330	335
IT5035	Canadair	90
AF2802	Airbus A330	335

Anomalies de la relation  
**Transports**

1. **Espace de stockage** : on apprend 4 fois que l'A330 a une capacité de 335 passagers

# *Un mauvais exemple*

## Transports

Vol	Avion	Capacité
IT5033	Airbus A330	335
AF2401	Airbus A330	335
AF2409	Boeing 727	150
IT5133	Airbus A330	335
IT5035	Canadair	90
AF2802	Airbus A330	335

Anomalies de la relation  
**Transports**

1. **Espace de stockage** : on apprend 4 fois que l'A330 a une capacité de 335 passagers
2. **Problèmes de mises à jour** : changement de la capacité des avions ?  
Changement de type d'avion sur un vol ?

# *Un mauvais exemple*

## Transports

Vol	Avion	Capacité
IT5033	Airbus A330	335
AF2401	Airbus A330	335
AF2409	Boeing 727	150
IT5133	Airbus A330	335
IT5035	Canadair	90
AF2802	Airbus A330	335

Anomalies de la relation  
**Transports**

1. **Espace de stockage** : on apprend 4 fois que l'A330 a une capacité de 335 passagers
2. **Problèmes de mises à jour** : changement de la capacité des avions ?  
Changement de type d'avion sur un vol ?
3. **Problème d'insertion** : incohérence en cas d'introduction d'un A330 avec une capacité de 300. Introduction d'un nouveau type d'avion sans le faire voler ?



# Un mauvais exemple

## Transports

Vol	Avion	Capacité
IT5033	Airbus A330	335
AF2401	Airbus A330	335
AF2409	Boeing 727	150
IT5133	Airbus A330	335
IT5035	Canadair	90
AF2802	Airbus A330	335

Anomalies de la relation  
**Transports**

1. **Espace de stockage** : on apprend 4 fois que l'A330 a une capacité de 335 passagers
2. **Problèmes de mises à jour** : changement de la capacité des avions ?  
Changement de type d'avion sur un vol ?
3. **Problème d'insertion** : incohérence en cas d'introduction d'un A330 avec une capacité de 300. Introduction d'un nouveau type d'avion sans le faire voler ?
4. **Problème de suppression** : suppression de IT5035, supprime le type Canadair.


# *Le modèle relationnel*

- Défini par E. F. Codd (IBM Research) dans « A Relational Model of Data for Large Shared Data Banks », CACM 13, No. 6, June 1970)
- Indépendant de la représentation physique des données
- Assise mathématique forte (algèbre relationnelle, formes normales)

## **Objectifs généraux**

- ✓ Éliminer les comportements anormaux des relations lors des mises à jours
- ✓ Éliminer les données redondantes
- ✓ Meilleure compréhension des relations sémantiques entre les données

# Conception d'un schéma relationnel

**But** : Éliminer les anomalies de la relation universelle pour faciliter la manipulation des relations  **Normaliser les relations.**

**Méthode** : **Décomposer** la relation universelle en sous-relations qui ne souffrent pas des anomalies précédentes.

**Problématique** : obtenir un nouveau schéma de base de données qui :

- ✓ conserve toutes les données,
- ✓ conserve un minimum de contraintes d'intégrité mais respecte le réel perçu,
- ✓ élimine toutes les anomalies.

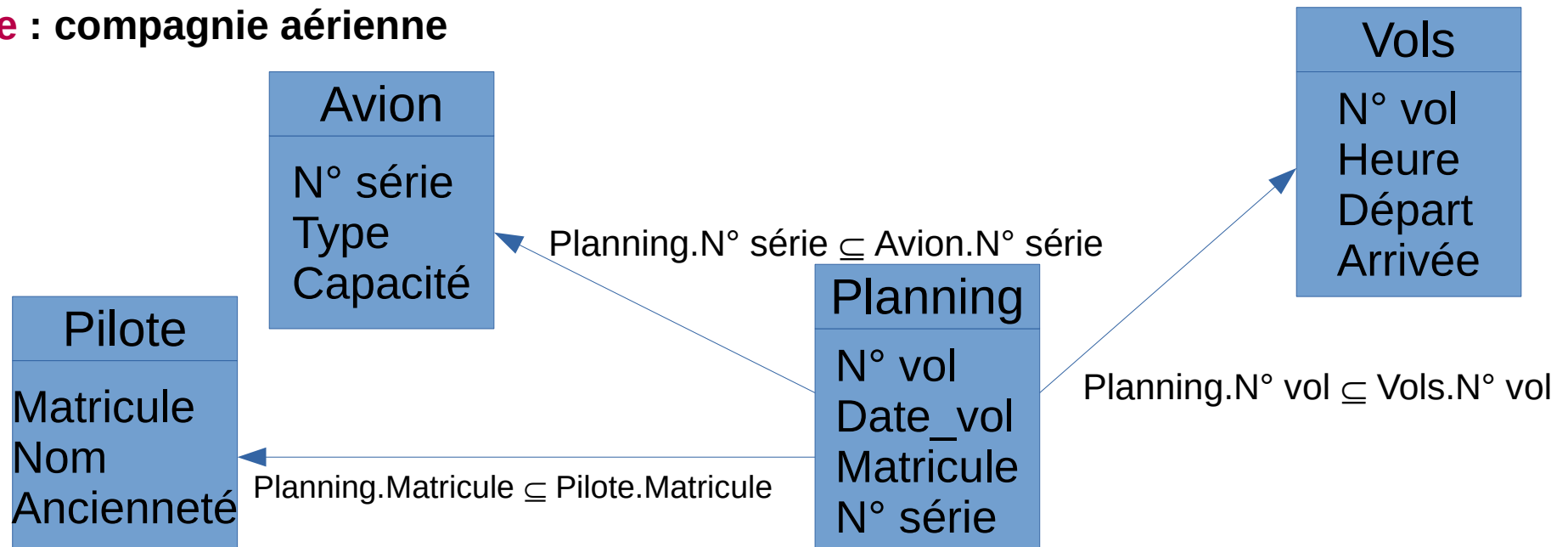
- Partie théorique qui sera abordée (probablement rapidement) en fin d'UE.
- Pour l'UE : le schéma sera donné. Il faudra le comprendre, savoir l'importer dans une base de données et l'interroger.

## Concepts de base : base de données

### Définition (rappel)

- Un **schéma de base de données** est un ensemble de schémas de relations **liés** par des **dépendances référentielles (un type de contraintes)** : attributs communs ou plus généralement des dépendances d'inclusion.
- Une **base de données** est alors un ensemble de relations (extensions) associé au schéma de base de données et vérifiant toutes ses **contraintes**.

### Exemple : compagnie aérienne



## Concepts de base : base de données

**Avion**

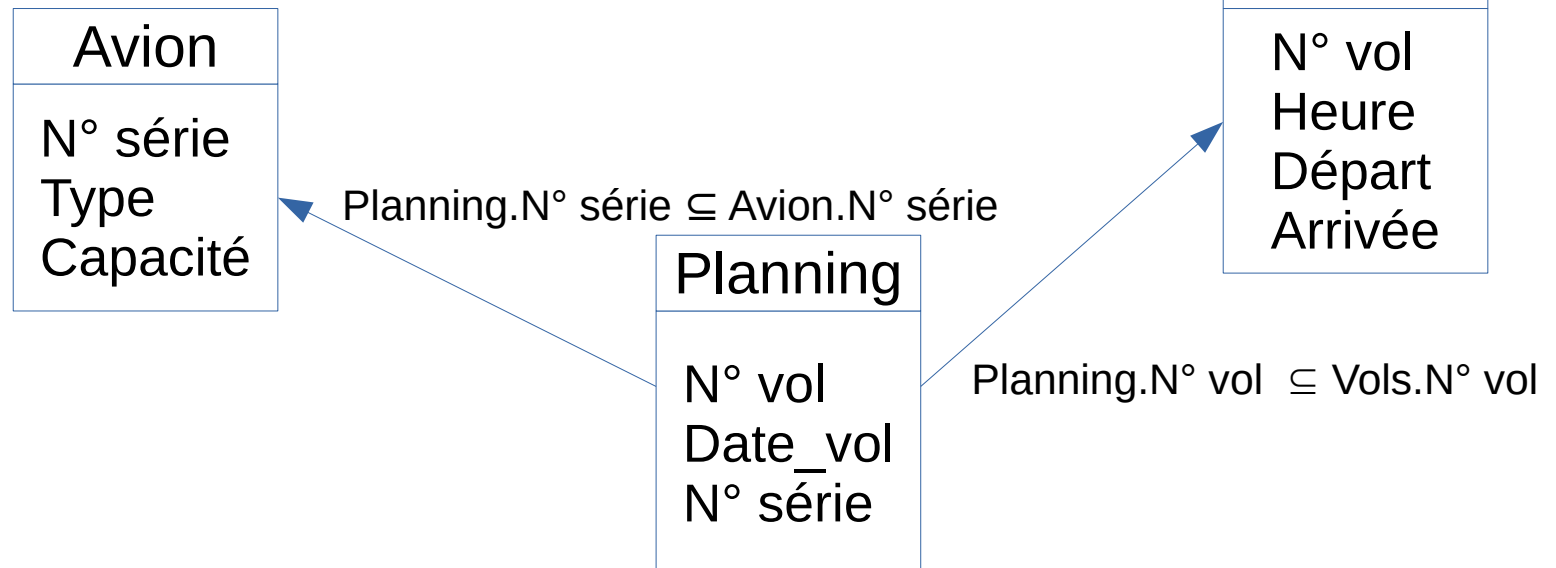
No Série	Type	Capacité
2001	Airbus A330	335
2002	Airbus A330	335
301	Boeing 727	150
94	Canadair	90

**Planning**

No Vol	Date_vol	No série
IT5033	...	2001
AF2401		2002
AF2409		2002
IT5133		301
IT5035		301
AF2802		2002

**Vol**

No Vol	Heure Départ	Heure Arrivée
IT5033	...	
AF2401		
AF2409		
IT5133		
IT5035		
AF2802		



# Contraintes d'intégrité

## Définition

- Tout schéma de base de données doit être conçu pour imposer le respect d'un maximum de contraintes, appelées **contraintes d'intégrité**, du réel perçu.
- Les **contraintes d'intégrité** sont des expressions logiques qui **doivent être satisfaites à tout instant** par une instance de base de données.

# Contraintes d'intégrité

## Définition

- Tout schéma de base de données doit être conçu pour imposer le respect d'un maximum de contraintes, appelées **contraintes d'intégrité**, du réel perçu.
- Les **contraintes d'intégrité** sont des expressions logiques qui **doivent être satisfaites à tout instant** par une instance de base de données.
- Plusieurs types (par ordre croissant de complexité) :
  - ✓ *Contraintes sur les attributs* : domaines, valeurs nulles, ... ;
  - ✓ *Contraintes sur les  $n$ -uplets* : la valeur d'un attribut peut dépendre d'un ou plusieurs autres attributs du même  $n$ -uplet (dates d'emprunt et de retour pour une bibliothèque), ... ;
  - ✓ *Contraintes sur les relations* : clés, cardinalité, ... ;
  - ✓ *Contraintes sur la base de données* : clés étrangères, ... ;
  - ✓ *Contraintes sur l'évolution temporelle de la base de données* : évolution chronologique (diplômes, état-civil), ...

# Contraintes d'intégrité : notion de clé

- Une relation est un ensemble de  $n$ -uplets. Par définition, un ensemble n'a pas d'élément en double, donc chaque  $n$ -uplet d'une relation est unique.
- Pour identifier les  $n$ -uplets de façon unique sans en donner toutes les valeurs et respecter leur unicité une clé est nécessaire.

## Définition

- Groupe d'attributs minimum qui détermine un  $n$ -uplet de façon unique.
- Plus formellement :  $X$  clé de  $R(U)$  avec  $X \subseteq U$  ssi  $\forall r: R(U), \forall t_1, t_2 \in r,$

$$t_1[X] = t_2[X] \Rightarrow t_1 = t_2$$

et

$$\nexists Y \subset X \Rightarrow t_1[Y] = t_2[Y] \Rightarrow t_1 = t_2$$



# *Contraintes d'intégrité : notion de clé*

## **Propriétés**

- Toute relation possède au moins 1 clé :

# *Contraintes d'intégrité : notion de clé*

## **Propriétés**

- Toute relation possède au moins 1 clé : l'ensemble de ses attributs

# Contraintes d'intégrité : notion de clé

## Propriétés

- Toute relation possède au moins 1 clé : l'ensemble de ses attributs
- Si une relation possède plusieurs **clés candidates**, on choisit une clé qui sera privilégiée : **la clé primaire**. Aucun des attributs d'une clé primaire n'admet de valeur nulle (vide).

# Contraintes d'intégrité : notion de clé

## Propriétés

- Toute relation possède au moins 1 clé : l'ensemble de ses attributs
- Si une relation possède plusieurs **clés candidates**, on choisit une clé qui sera privilégiée : **la clé primaire**. Aucun des attributs d'une clé primaire n'admet de valeur nulle (vide).
- **Défini sur le schéma de relation et pas sur les extensions.** Une clé est définie sur l'ensemble des valeurs possibles et pas seulement sur les valeurs présentes dans la base.

## Conventions d'écriture

- On souligne la clé primaire.
- Les clés candidates sont soulignées en pointillés (souvent omis).

## Exemple

Pilote (matricule, n° INSEE, nom, ancienneté)

Vol (n° vol, heure, départ, arrivée)

Avion (n° série, type, capacité)

Planning (n°vol, date vol, matricule, n° série)

# Contraintes d'intégrité : notion de clé

## Exemple illustratif

Soit la relation R(A,B,C,D,E) ci-dessous.

A	B	C	D	E
A1	B1	C1	D1	E3
A1	B1	C2	D1	E1
A1	B1	C5	D1	E1
A1	B2	C5	D4	E1
A2	B1	C5	D1	E1
A2	B1	C4	D1	E1
A2	B1	C5	D3	E2
A2	B2	C5	D4	E1
A3	B3	C5	D1	E2

Est-ce que (A, C, D) est une clé candidate de R ?

# Contraintes d'intégrité : notion de clé

## Exemple illustratif

Soit la relation R(A,B,C,D,E) ci-dessous.

A	B	C	D	E
A1	B1	C1	D1	E3
A1	B1	C2	D1	E1
A1	B1	C5	D1	E1
A1	B2	C5	D4	E1
A2	B1	C5	D1	E1
A2	B1	C4	D1	E1
A2	B1	C5	D3	E2
A2	B2	C5	D4	E1
A3	B3	C5	D1	E2

Est-ce que (A, C, D) est une clé candidate de R ?

Deux étapes :

# Contraintes d'intégrité : notion de clé

## Exemple illustratif

Soit la relation  $R(A,B,C,D,E)$  ci-dessous.

A	B	C	D	E
A1	B1	C1	D1	E3
A1	B1	C2	D1	E1
A1	B1	C5	D1	E1
A1	B2	C5	D4	E1
A2	B1	C5	D1	E1
A2	B1	C4	D1	E1
A2	B1	C5	D3	E2
A2	B2	C5	D4	E1
A3	B3	C5	D1	E2

Est-ce que  $(A, C, D)$  est une clé candidate de  $R$  ?

Deux étapes :

1. Vérification que  $ACD$  est clé

→ Valeur unique de  $ACD$  pour chaque  $n$ -uplet

# Contraintes d'intégrité : notion de clé

## Exemple illustratif

Soit la relation R(A,B,C,D,E) ci-dessous.

A	B	C	D	E
A1	B1	C1	D1	E3
A1	B1	C2	D1	E1
A1	B1	C5	D1	E1
A1	B2	C5	D4	E1
A2	B1	C5	D1	E1
A2	B1	C4	D1	E1
A2	B1	C5	D3	E2
A2	B2	C5	D4	E1
A3	B3	C5	D1	E2

Est-ce que (A, C, D) est une clé candidate de R ?

Deux étapes :

1. Vérification que ACD est clé

→ Valeur unique de ACD pour chaque  $n$ -uplet

2. Vérification que la clé est minimale

→ Tous les attributs de la clé sont nécessaires



# *Les bases NoSQL*

# *Les bases NoSQL*

- **Alternatives modernes aux bases de données relationnelles.**
- **NoSQL**  $\Rightarrow$  *Not Only SQL*.
- **NoSQL**  $\rightarrow$  Open-source, distributed, non-relational databases

## **Historique**

Le terme « NoSQL » dans sa signification actuelle vient du nom d'un séminaire (pendant une conférence sur Hadoop) qui a eu lieu en 2009 sur les nouveaux types de bases de données qui n'utilisent pas SQL. Il fallait un nom qui fasse un bon hashtag sur Twitter : court, facile à mémoriser, et non populaire sur Google pour espérer pouvoir remonter dans les premiers résultats.

# *Les bases NoSQL*

- **Limites des bases de données relationnelles**
  - Modèle non adapté au fonctionnement moderne sur un réseau comme Internet (Internet n'existait pas en 1970)
  - Si Facebook fonctionnait grâce à des bases de données relationnelles à chaque mise à jour d'un mur, tout le système serait bloqué le temps que l'ensemble des serveurs répartis sur la planète soit mis à jour...
  - Modèle relationnelle peu performant et mal adapté aux données « big data » (nombre très important de colonnes et de lignes, nombreuses données manquantes, besoin de stocker des données complexes).

# Les systèmes NoSQL

- *Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable (<http://nosql-database.org/>).*
- Pas un Système de Gestion de Bases de Données mais plus un système de stockage de données.
- Pas de modèle pré-défini, pas de contrainte stricte comme en relationnel, tout doit rester simple pour aller vite. L'ajout d'un serveur doit améliorer les performances.

# Les systèmes NoSQL

Les systèmes NoSQL peuvent être résumés ainsi :

## **Basically Available, Soft state, Eventually consistent**

- **Eventual consistency** : la cohérence du système est atteinte au bout d'un certain temps (au lieu de en permanence en relationnel).
- **Soft state** : l'état du système peut donc changer même sans action de l'utilisateur pour atteindre un état global cohérent. Il faut donc que le système soit capable de fonctionner dans des états transitoires.
- **Basically available** : le système reste donc globalement opérationnel même si parfois il peut rendre un service dégradé.

# *Les systèmes NoSQL : choix*

## **4 grands types de systèmes**

- Orienté clé-valeurs
- Orienté documents
- Orienté colonnes
- Orienté graphes

# *Les systèmes NoSQL : clé-valeurs*

- Rapidement : stockage simple et direct d'un tableau associatif (hash-table, dictionnaire Python). Possible en relationnel avec une table à deux attributs dont les accès se font uniquement par la clé primaire.
- Système le plus simple : lecture, écriture et suppression. Ne fait que stocker des données en rapport avec une clé.
- Implique pas de relation entre les données. Le système ne connaît pas les données ou ce qu'elles représentent (sémantique associée).
- Interrogation uniquement par la clé. On récupère une donnée (blob notamment) associée à une clé.

# *Les systèmes NoSQL : Documents*

- Basé sur un système clé-valeur
- Mais la valeur est une structure que le système connaît et peut donc parcourir pour n'en renvoyer qu'une partie.
- Pour la plupart des systèmes pas d'opérations croisées entres plusieurs documents.



# Les systèmes NoSQL : colonnes

- « Opposé » d'une BD relationnelle : en relationnel, le modèle est orienté  $n$ -uplets dans des tables, le nombre de colonnes est fixé. Avec un modèle orienté colonnes, on ne stocke que les colonnes qui ont des valeurs non nulles regroupées dans des familles de colonnes.
- Les colonnes sont stockées sous une forme clé-valeur. On peut ajouter une colonne dans un enregistrement aussi facilement que l'ajout d'un  $n$ -uplet en relationnel.

	A	B	C	D	E
1	foo	bar	hello		
2		Tom			
3			java	scala	cobol

Organisation d'une table dans  
une BDD relationnelle

1	A foo	B bar	C hello
2	B Tom		
3	C java	D scala	E cobol

Organisation d'une table dans  
une BDD orientée colonnes

<http://blog.xebia.fr/2010/05/04/nosql-europe-bases-de-donnees-orientees-colonnes-et-cassandra/>

# *Les systèmes NoSQL : Graphes*

- On ne stocke plus un simple ensemble de données mais des relations.
- Stockage d'entités et de relations entre les entités. On peut ajouter des propriétés aux relations et aux entités.
- Cohérence forte des données. Pas d'arête pendante.
- Difficile de monter en charge (ajout de serveurs). Le stockage d'un graphe sur plusieurs serveurs est un problème difficile (problème de performance même pour un simple calcul de chemin si le graphe est réparti sur plusieurs serveurs).