Administration des SGBD

Modélisation BigData (documents & multi-colonnes)

M1 Informatique

Damien Ploix

Modélisation BigData

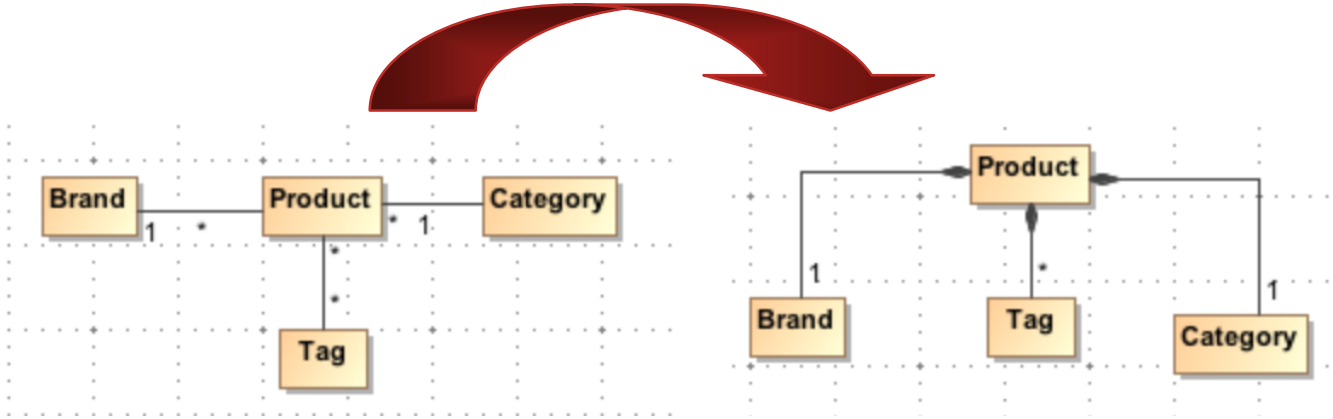
- Rappel : familles de bases « BigData »
  - Documents (JSON) : MongoDB (Oracle / MySQL / SQLServer / PostgreSQL)
  - Multicolonnes (tableau) : Cassandra, HBase
  - Clé/Valeur (token) : Redis
  - Recherche/indexation : Elastisearch / Splunk / Solr
  - Graphes (vectoriel) : Neo4j (Oracle / SQLServer)

- Points commun :
  - Elles posent une « structure » permettant d'intégrer des données sans en décrire/figer le schéma.
  - Ces « structures » de base sont adaptées par rapport aux traitements réalisés

Modélisation BigData

- ***Document, Famille de colonnes :***
  - Les données sont agrégées au sein ***d'agrégat***
  - L'agrégation obéit en général au ***modèle de consultation***
  - Toutes les entités censées être ***restituées ensemble*** sont agrégées au sein d'une ***même structure***

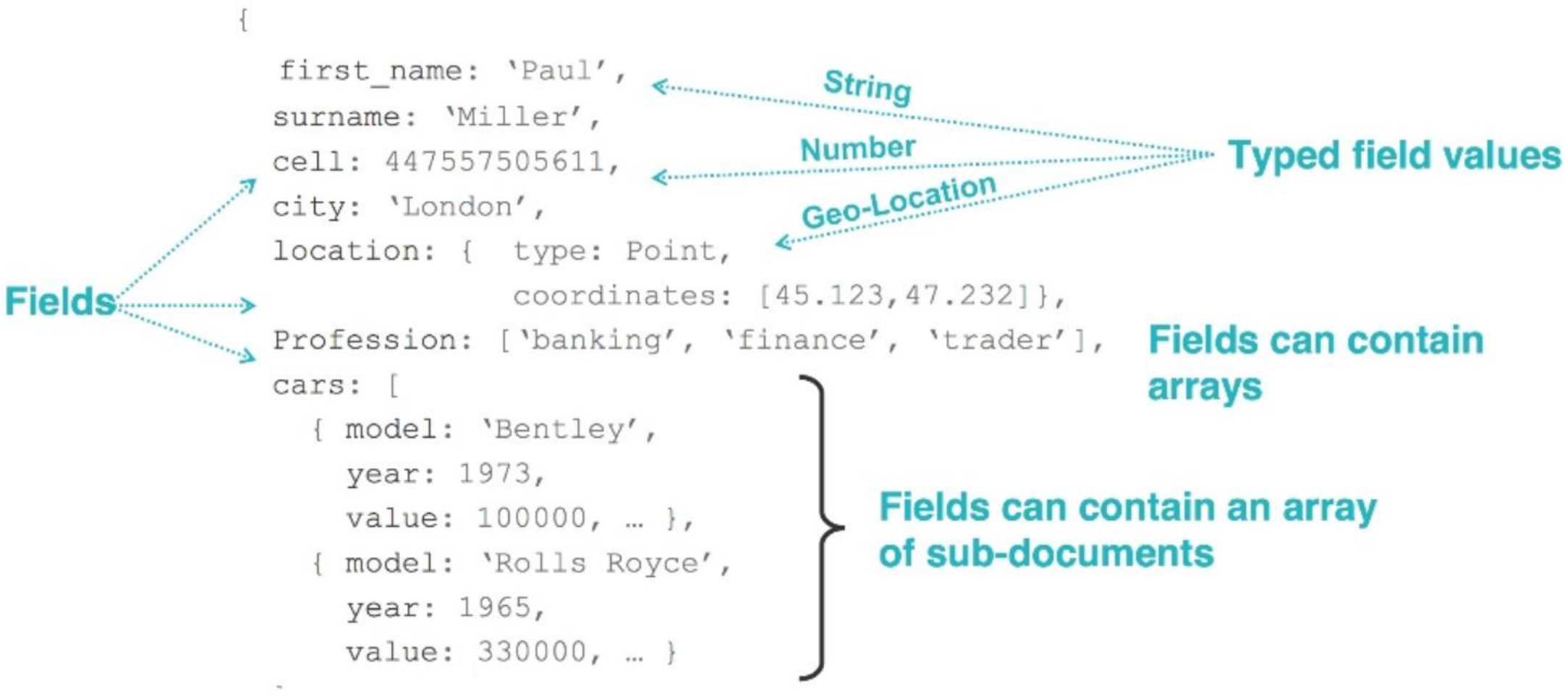
Objectif : gestion des produits



NoSQL / Principes de modélisation

- Conséquence du modèle d'agrégat : limite du modèle Relationnel
  - Dans le modèle relationnel il est impossible de distinguer les relations d'agrégation des relations de liaison (car les requêtes ne sont pas connues à priori)
  - Les moteurs RDBMS ne sont pas conçus pour utiliser cette information d'agrégation et optimiser le modèle de stockage ou distribuer les données sur plusieurs serveurs car ils effectuent une résolution dynamique des liaisons.
  - Solution/optimisation via la dénormalisation (cf cours précédent) pour limiter les liaisons/jointures (exemple type : modélisation décisionnelle, tables préjointes, rapports pré-calculés, vues matérialisées, ...)

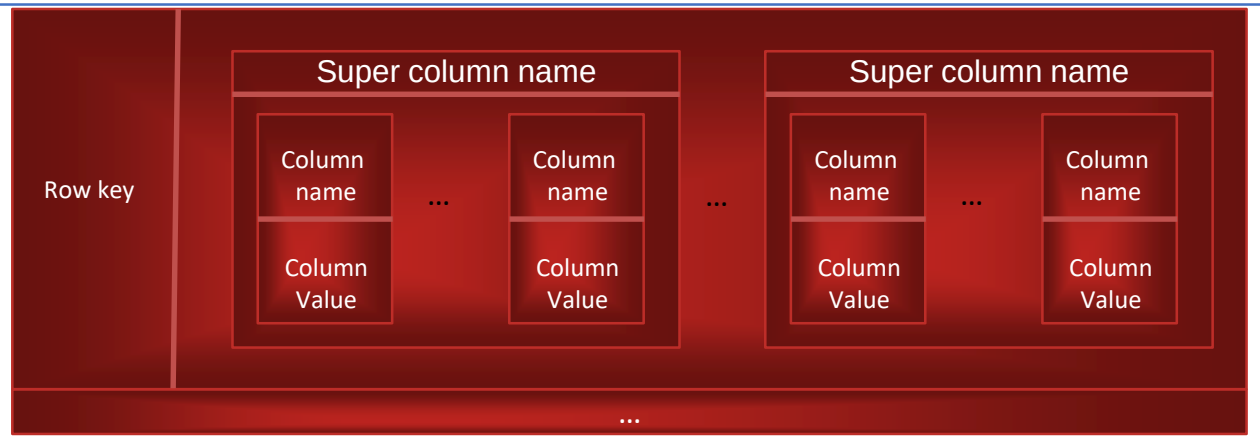
Structure de données « documents » (ex MongoDB)



Source : Daniel Coupal

Structure de données tableau (exemple Hbase)

Famille de colonnes et super colonnes



User table Column family for book ratings by userid for bookids

Key	data:fname	...	rating:bookid1	rating:bookid2
userid1			5	4

Book table Column family for ratings for bookid by userid

Key	data:title	...	rating:userid1	rating:userid2
bookid1			5	4

"1234…2121"	ProductInfo			SellerInfo		
	Title	…	Description	Name	…	Location
	Product1	…	l	ToyStore	…	Paris

```
Keyspace.get("products",  "1234…2121", "ProductInfo", "Title")
```

Structure de données HBase

- Une table est un ensemble de ligne (row)
- Une ligne est un ensemble de familles de colonnes (mais limitées)
  - Une famille est comparable à un espace de nomage (famille:colonne)
- Une famille de colonne est un ensemble de colonne
  - *pas limitées* en nombre : possibilité d'en avoir des M
  - Une colonne est une paire clé/valeur
- Les valeurs sont dans les cellules famille:colonne
  - Versionnée (3 versions par défaut) via un timestamp ou par un versionnage explicite
- ***Valeur = table+ligne+famille+colonne+timestamp***

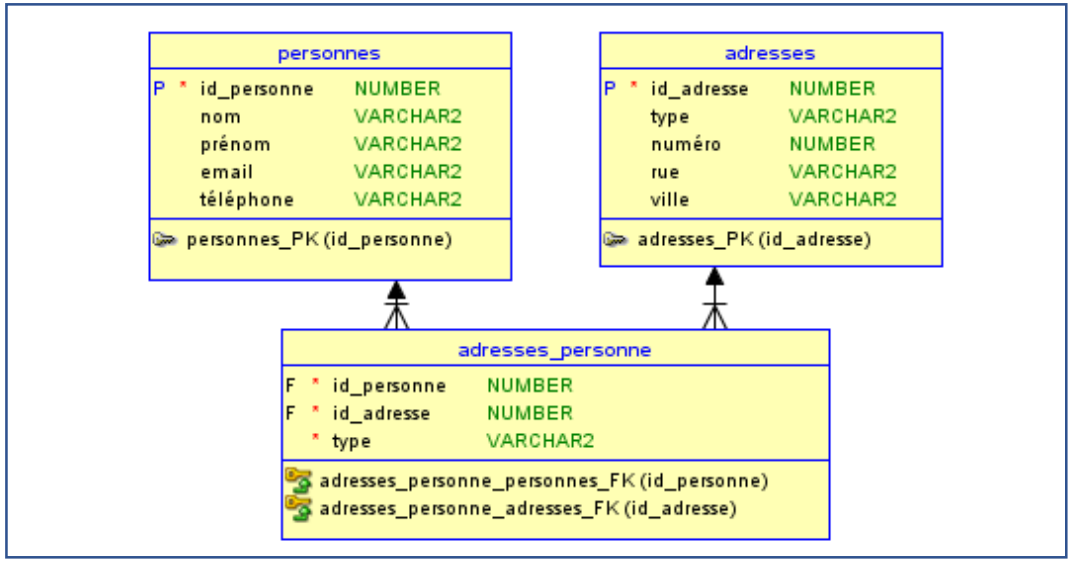
ID Ligne	timestamp	employe				departement	
empno		ename	job	sal	comm	dname	loc
1	T1	toto	analyste	5000	100	marketing	paris
1	T2		manager	5500			
2	T10	titi	manager	6000	150	marketing	paris
2	T12				200		
2	T20					ventes	lyon

Modélisation comparée document (MongoDB), tableau mutli-colonnes (Hbase), relationnel (Oracle)

Objectif : gérer les information sur une personnes (identité, adresses) :

Le prénom et le nom : Damien, Ploix
L'adresse email, le téléphone portable : damien.ploix@univ-evry.fr, privé
Adresse d'habitation (numéro, rue, ville) : 28, privé, Montreuil
Adresse de travail (numéro, rue, ville) : 35, privé, La Défense

```
{
 "nom": "Ploix",
 "prenom": "Damien",
 "email": "damien.ploix@univ-evry.fr",
 "téléphone": "privé",
 "adresses": [
  {
   "type": "habitation",
   "numero": 28,
   "rue": "privé",
   "ville": "Montreuil"
  },
  {
   "type": "travail",
   "numero": 35,
   "rue": "privé",
   "ville": "La Défense"
  }
 ]
}
```



famille de colonnes	**identité**				**adresses**							
colonnes	**nom**	**prénom**	**email**	**téléphone**	**type1**	**numéro1**	**rue1**	**ville1**	**type2**	**numéro2**	**rue2**	**ville2**
valeur	ploix	damien	damien.ploix@univ-evry.fr	privé	habitation	28	privé	Montreuil	travail	35	privé	La Défense

Migration de données d'une base dans une autre

- Intégration de données en 3 étapes :
  - Extraction des données de la base source
  - Transformation pour les mettre dans le format correct
  - Load pour les charger dans la base cible

- Outils utilisés :
  - Spécifiques ETL (Talend, SQLServeur, …)
  - Scripts étudiés en classe bash shell, NodeJS (via le TD5).

Exemple : d'Oracle à MongoDB

- Extraction :
  - Requête(s) SQL correspondant aux données
  - Transformation en format JSON
  - Chargement dans le(s) base(s)/collection(s) MongoDB cible

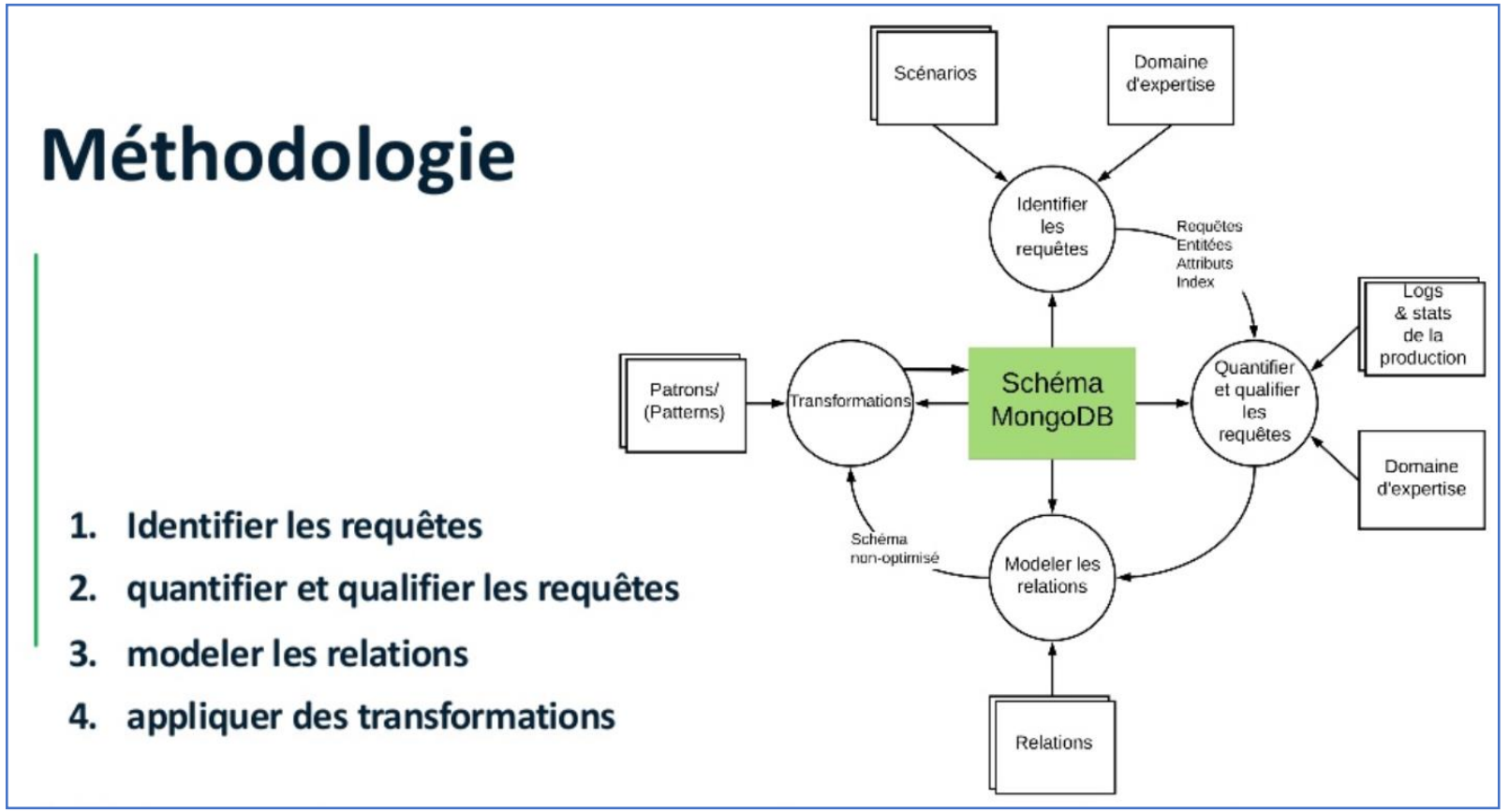
- Objectif :
  - Gérer les adresses des personnes dans MongoDB depuis les données présentes dans la base Oracle source.

Démonstration par l'exemple ☺ !

- Système de réservation et de gestion de salles de réunion à la défense.
- Il est demandé d'étudier le stockage des informations via une base NoSQL de type MongoDB.

Modélisation NoSQL (document / tableau multi-colonnes)

	SQL (modèle : entité/relation)	NoSQL (modèle : tables/documents)
Objectif	Utilisation multiples (générale/générique)	Utilisation spécifique
Répond à	*Quelles réponses avons-nous ?*	*Quelles questions avons-nous ?*
Étape 1	Modélisation des données	Écriture des requêtes
Étape 2	Écriture de l'application	Ajout d'indexes
Étape 3	Écriture des requêtes	Modélisation des données
Étape 4	Ajout d'indexes	Écriture de l'application



Source : Daniel Coupal

Modélisation NoSQL : du scénario aux requêtes

- Les étapes du scénario sont :
  - Réservation d'une salle
  - Contrôle des salles disponibles
- Les estimation de volumétrie sont :
  - Nombre de salles : 1000
  - Nombre d'utilisateurs : 100K
  - Nombre de réservation : 10 / salle / jour

Modélisation NoSQL : requêtes

Requête	Type	description
Réservation d'une salle	Écriture	un utilisateur réserve une salle sur un créneau donné pour un nombre de participant fixe
Recherche de salle	lecture	Récupère le nombre de salles disponibles

Quantifier / Qualifier les requêtes

Requête	Quantification	Qualification
Réservation d'une salle / j	1000 * 10	Écriture critique
contrôle des salles disponibles	1000	Lecture critique

Espace de stockage nécessaire :

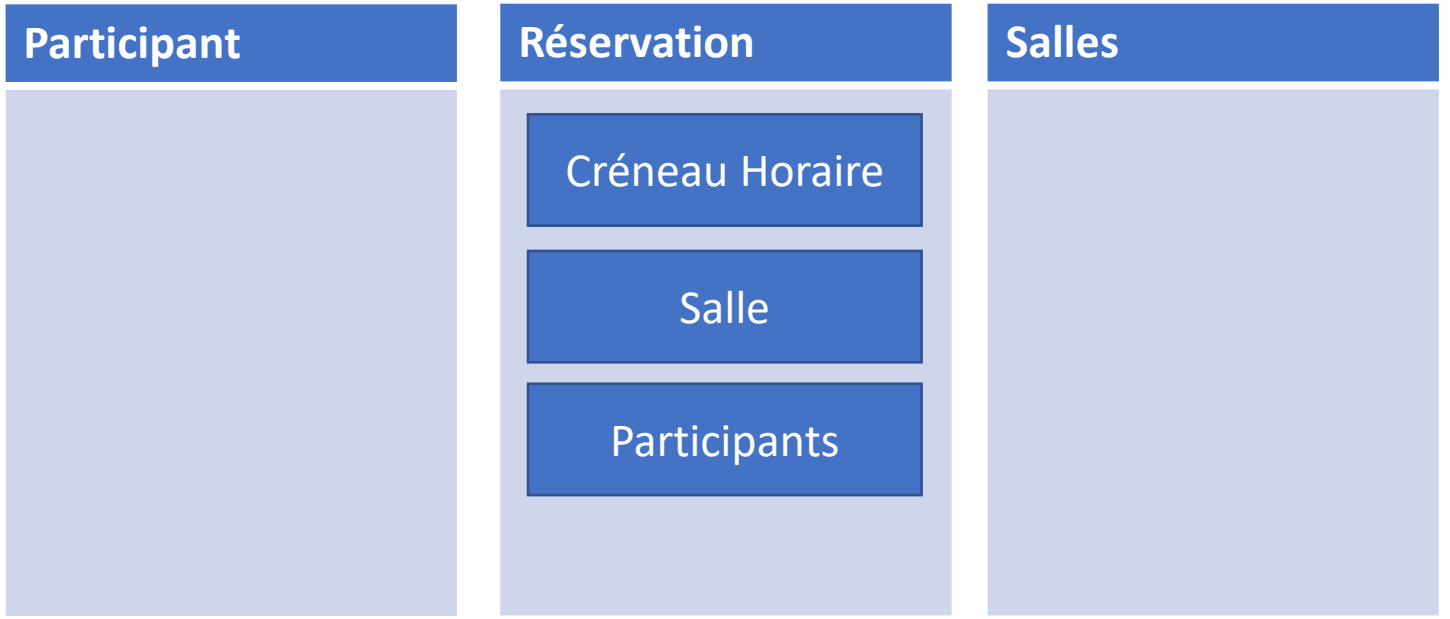
- Réservation : 1000 * 10 * 365 : 3,65M
- Abonnés : 100K

Modélisation des documents

Type de relation (cardinalité)	1 – 1	1 – N	N – N
Document imbriqué dans le parent	• Lecture unique • Pas de jointure	• Lecture unique • Pas de jointure	• Lecture unique • Pas de jointure • Duplication d'information
Document référencé dans le parent	• Lectures multiples • Lectures plus petites	• Lectures multiples • Lectures plus petites	• Lectures multiples • Lectures plus petites • Pas de duplication de l'information

Entités pour la réservation de salles

- Salle (bâtiment, …)
- Créneau horaire
- Participant (estAbonné, …)
- Réservation



Optimisation du schéma

- Les principales optimisations (patterns de conception) adressent les problématiques suivantes :
  - Évolution de la définition du schéma de données
    - Les documents intègrent un numéro de version de schéma
  - Optimisation du coût de dérivation / reporting / pré-calcul
    - Les documents stockent le résultat de calcul sur les informations
  - Optimisation de l'accès à une partie des informations
    - Pour une série d'informations incluse, les documents distinguent le sous ensemble accédé fréquemment (les 100 dernières) par rapport aux autres (les plus anciennes)
  - Regroupement / séparation d'informations
    - Regroupement dans un « méta » document de N documents unitaires d'une série
  - Références externes
    - Établissement de lien entre documents
  - Hiérarchie
    - Présence de lien hiérarchique (parent / enfant) et parcours de graphes

Modèle optimisé

- Réservation de salles : Optimisation du modèle :
  - Intégration d'un numéro de version de schéma dans les documents,
  - Regroupement des réservations dans un document Réservations,
  - Regroupement de la série des réservations dans une sous série « récentes » et une sous série « passée » et définition d'un attribut définissant la date de bascule,
  - Maintient à jour de propriétés précalculées dans le document des réservations,
  - Références externes entre document et participant,

Participant

_id=number
Schéma_version=1,
Nom=string,
Prenom=string,
estAbonné=boolean,
NumeroAbonne=number

Salles

_id=number
Schéma_version=1,
Adresse,
…

UneReservation

Numero=number
Abonne=référence_abonne,
Participants=[références_personnes],
Salle=référence_salle,
creneau={début=date,fin=date}

Réservation

_id=number
Schéma_version=1,
Max_récentes=nombre (N),
Salles_disponibles=[xN {reference_salle, créneau}]
Recentes=[xN documents « UneReservation »],
Anciennes=[xM documents « UneReservation »]

Administration MongoDB

Administration MongoDB : Types de configuration

- Les types de configuration sont :
  - Standalone
  - Haute disponibilité via des « replica set » :
    - Failover automatique
    - Redondance des données
  - Scalabilité horizontale « shrading » :
    - Distribution des données entre instances

      ➔ chaque instance va être responsable de gérer une/un ensemble de collection(s)

Administration MongoDB : stockage

- Stockage des données dans des fichiers gérés par MongoDB dans le répertoire indiqué dans le fichier de configuration de l'instance.
- Fonctionnement du stockage :
  - Limitation des accès concurrent au niveau des documents

    ➔ si un document intègre trop de données, il peut entraîner des attentes
  - Génération de « snapshot » :
    - Présentent une version cohérente des documents en mémoire à un point temporel (60s) ou de quantité de modifications (2GB de log)
  - Génération de write-ahead log (journaux) :
    - Enregistre les modifications entre deux snapshots
    - Peut être désactivé
      - Entraine la perte des modifications entre deux snapshot pour les instance standalone,
  - Compression :
    - Par défaut snappy pour les documents et prefix pour les indexes, permet zlib, …

Administration MongoDB : in-memory

- Le travail d'une instance en mode « in-memory » entraîne :
  - Pas de stockage sur disque mais en mémoire
  - Pas de journalisation
  - Pas de garantie de durabilité
  - La limitation des accès concurrent au niveau des documents

- Mais peut être intégré à un déploiement sécurisé via des replica set :
  - Les instances « in-memory » seront positionnées en « principales »,
  - Les instances « stockage sur disque » seront positionnées en back

  ➔ spécialisation de l'instance de stockage sur l'écriture des données,

  ➔ resynchronisation automatique en cas de redémarrage d'une instance « in-memory »

- Ou à un déploiement réparti via des shard :
  - Les instances « in-memory » vont gérer les documents non persistants (liés à du calcul),
  - Les instances « stockage sur disque » vont gérer les documents persistants.

Administration MongoDB : Configuration logicielle

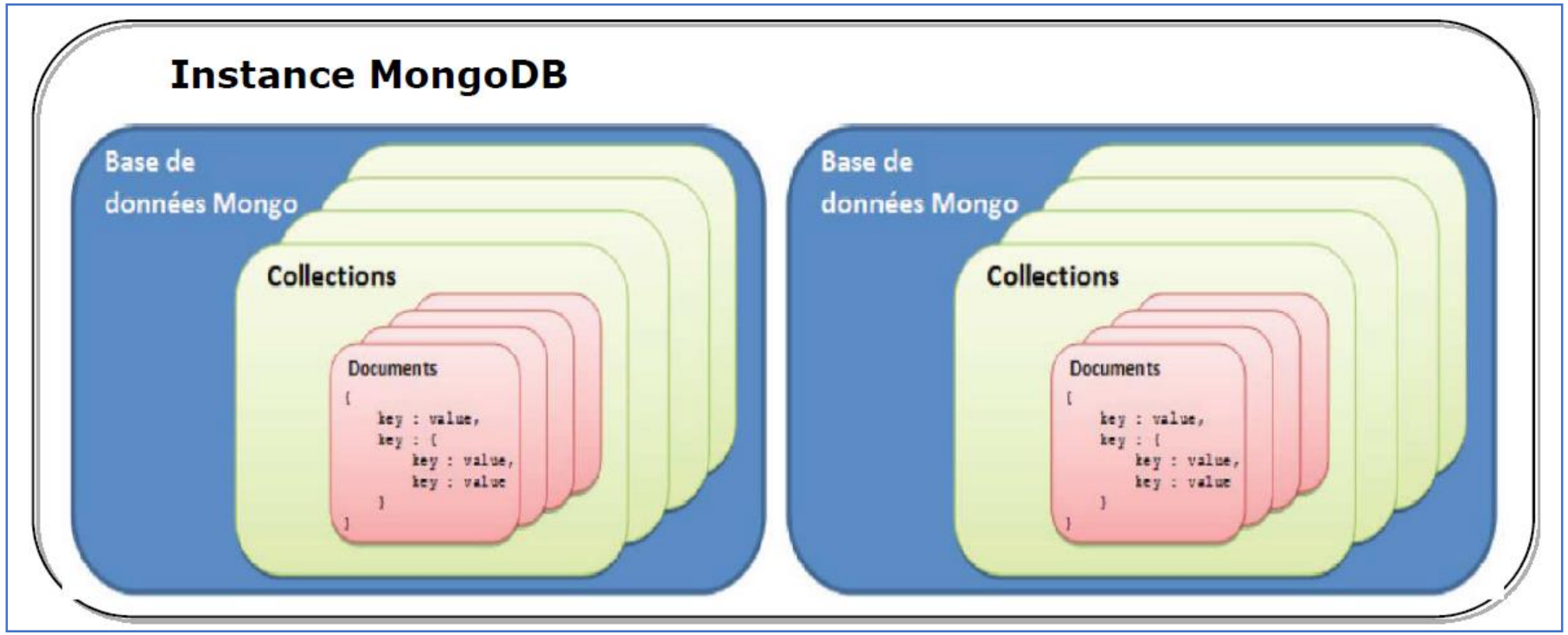
- Les distribution de MongoDB existent pour les principales version de Linux (et également pour Windows).
- Pour l'installation sous Oracle Linux (serveur Oracle utilisé pour le projet), la distribution à utiliser est celle de redhat
  - https://docs.mongodb.com/manual/tutorial/install-mongodb-on-red-hat/
- Le principal élément de configuration est le fichier mongod.conf
  - Placé par défaut dans /etc
  - Les sections (les plus) essentielles :
    - storage : définition du mode stockage, du répertoire de données, de la journalisation
    - net : port d'écoute de l'instance et adresses IP pouvant accéder à l'instance (bindIP)
  - https://docs.mongodb.com/manual/reference/configuration-options/

Administration MongoDB : configuration de la sécurité

- Sécurisation « utilisateur »
  - Par défaut, aucune sécurisation n'est en place et les connections ne demandent pas de nom d'utilisateur ni ne restreignent les droits (admin par défaut).
  - Activation via la propriété « security.authorization=enable » dans mongod.conf
- Permet l'utilisation d'utilisateurs pour chaque base de données
  - db.createUser, db.dropUser, db.changeUserPassword, …
- Et la gestion des droits pour les utilisateurs :
  - db.grantRolesToUser, db.revokeRolesToUser
  - Les droits « built-in » sont :
    - read : permet la lecture de documents dans la db,
    - readWrite : permet en + l'écriture de documents et la création d'indexes dans la db,
    - dbAdmin : permet la réalisation d'activité d'administration de la db,
    - userAdmin : permet la gestion des utilisateurs et des droits
    - dbOwner : readWrite+dbAdmin+userAdmin

Administration MongoDB : configuration logicielle

- Architecture logique MongoDB :



Mise en œuvre

- Architecture logique MongoDB :

  - Les « bases de données » et les « collections » définissent des « espace de nommage » pour l'accès et l'indexation des documents/données.
  - Les collections sont comparables aux tables, elles permettent les références externes entre documents.

On continu avec l'exemple !

```
$ mongo
> use reservationSalles;                    # créé une/se positionne sur base de donnée pour la réservation de salles
> db.createCollection('salles');            # créé la collection des salles
> db.createCollection('personnes');         # créé la collection des personnes
> db.createCollection('reservations');      # créé la collection des reservations
> db.personnes.insert({ nom:"ploix", prenom:"damien", estAbonne:true, numeroAbonne:1}); # premier abonné !
> db.personnes.find( { } );
{ "_id" : ObjectId("5dd15aa26ff79a34fb912909"), "nom" : "ploix", "prenom" : "damien", "estAbonne" : true, "numeroAbonne" : 1 }
```

Pour conclure

DBA

Outillage du DBA

- Le Rôle de l'outillage est de réduire :
  - Le temps des opérations par
    - Leur automatisation,
    - Leur simplification,
  - Les efforts liés à l'administration :
    - Les systèmes administrés sont de plus en plus complexes,
  - Les erreurs humaines…
- Problème de l'outillage :
  - Très lié au DBMS utilisé… et
  - Les organisations implémentent(aient) *leur* outils/procédures d'administration,

Outillage du DBA

- Angle d'attaque :
  - Identifier les besoins et choisir ensuite…
  - Ajuster les options proposées aux besoins et compléter par des développements/scripts spécifiques…
  - Critères de sélection…
- **Modélisation et design**
  - Génération du modèle physique adapté au DBMS cible,
  - Système expert de vérification et de conseil
  - Cross-référence entre les modèles logiques et physique qui permettent les dénormalisation,
  - Interface avec les environnements de dev.
- Les outils sont des outils de CASE…

Outillage du DBA

- **Interrogation et analyse**
  - Génération de scripts SQL corrects pour tous les objets de la base,
  - Analyse des « DROP » afin d'en mesurer l'impact,
  - Permet le parcours interactifs de tous les objets de la base.
- **Gestion du changement**
  - Outils de gestion du changement
  - Comparaison de bases de données
  - Outils de migration
  - Vérification de l'intégrité du référentiel
  - Audit
  - Interrogation / analyse du catalogue
  - Sécurité

Outillage du DBA

- **Gestion de la performance**
  - Observation du système
  - Observation de la base de donnée
  - Observation de l'application
  - Observation de l'espace

- **Gestion des entrepôts de données**
  - Phase de chargement :
    - Réplication : ETL du commerce ou scripts (?)
    - Propagation : EAI / Bus inter-applicatif si besoin
  - Phase d'analyse
    - optimiseur de requête (MicroStratégie, Buissness Object, …)
    - orientés calcul (SAS, …).
  - Phase d'optimisation / surveillance :
    - C'est une base comme les autres

Règles d'Or du DBA

- Analyser, simplifier, se concentrer,
  - Du fait du nombres d'éléments en jeu…
- Ne pas paniquer !
  - Il y aura des problèmes, quoi que l'on fasse…
- Analyser, tester, tester, tester, puis appliquer !
  - Rien à ajouter ?
- Comprendre le métier et pas uniquement la technologie,
  - Permet de comprendre les contraintes *réelles !*
- Ne pas devenir un Hermite,
  - Être accessible malgré la tendance et la réputation…
- Utiliser toutes les ressources à sa disponibilité,
  - Les problèmes ont déjà été rencontrés par d'autres…
- Rester à la page,
  - Même les bases de données évoluent…