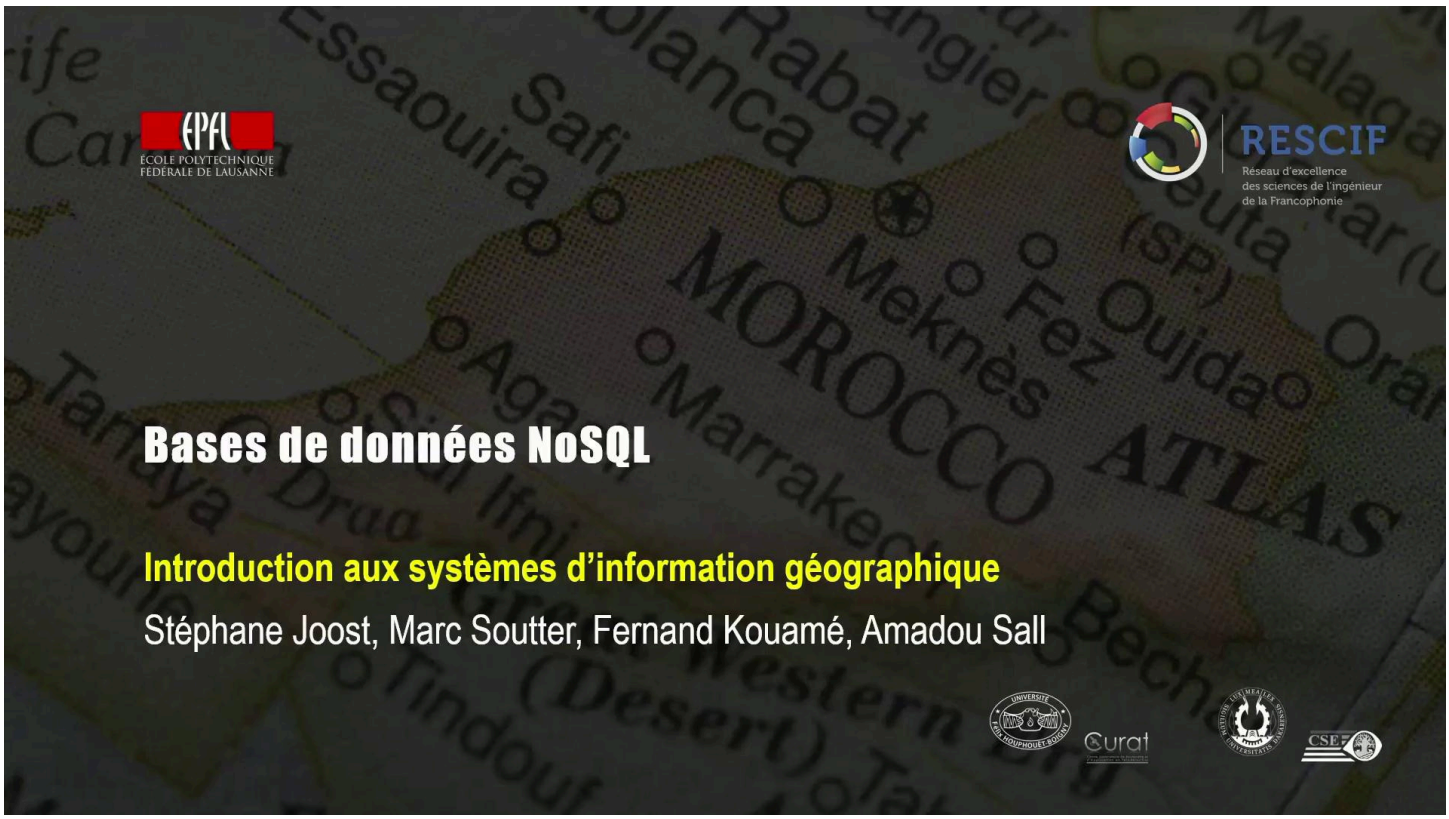
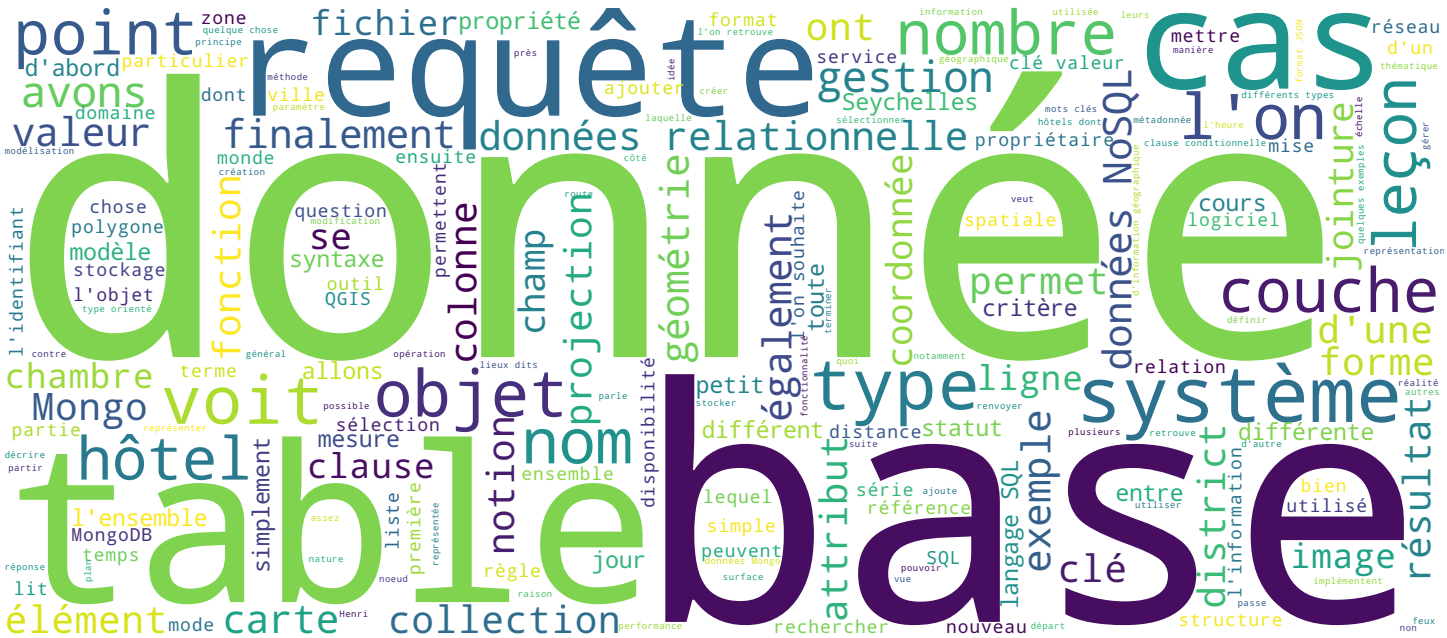
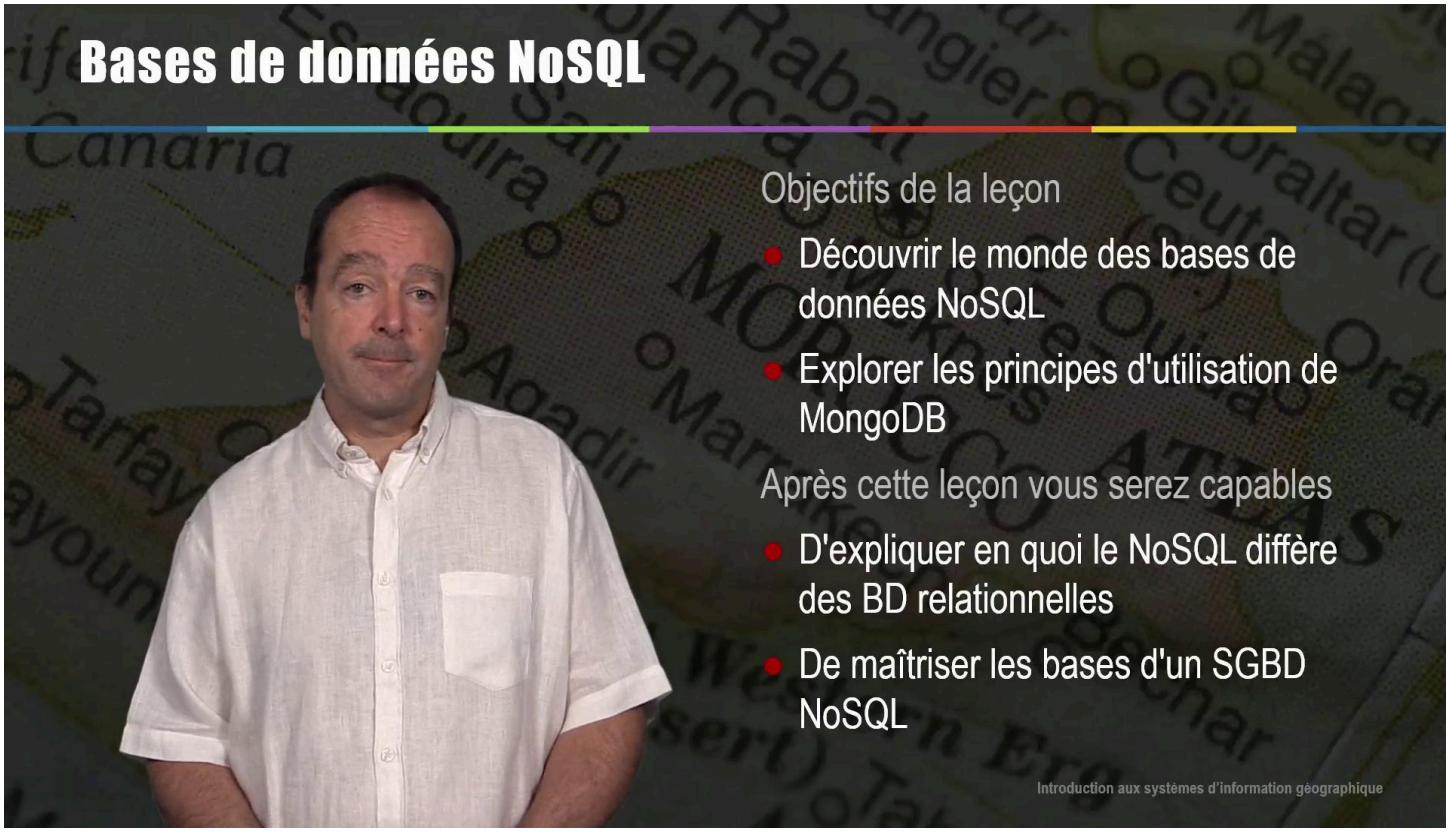




Search MOOC

Video

EPFL

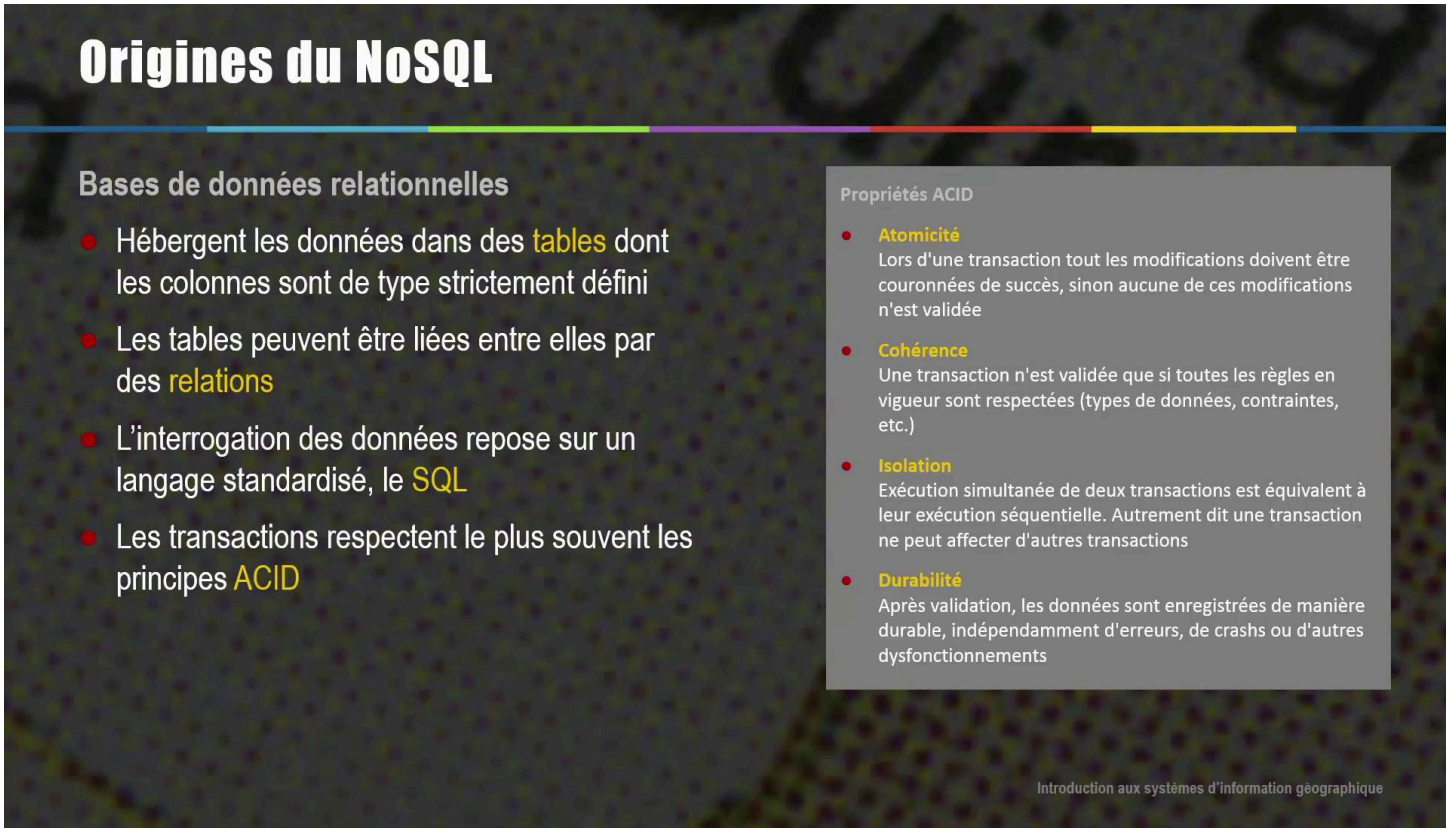


Cette leçon va donc porter sur les bases de données NoSQL qui sont une vaste famille regroupant un grand nombre de types de base de données différentes et qui se distinguent des bases de données relationnelles classiques que nous avons vu jusqu'à présent par le fait qu'à l'origine elles n'implémentaient pas le langage SQL. Donc la logique de stockage de l'information est un peu différente et les règles d'utilisation également. Nous aborderons toutes ces questions au cours de la leçon. L'objectif de cette leçon est donc de compléter un petit peu ce que nous avons appris des bases de données relationnelles en ouvrant le champ à l'exploration des bases de données NoSQL et peut-être plus particulièrement à l'une d'entre elles qui est la base de données Mongo. Au terme de cette leçon vous devriez être en mesure d'expliquer en quoi le monde NoSQL se différencie du monde des bases de données relationnelles et vous devriez être capable d'utiliser un petit peu une de ces bases de données.

Notes

Summary

0m 22s

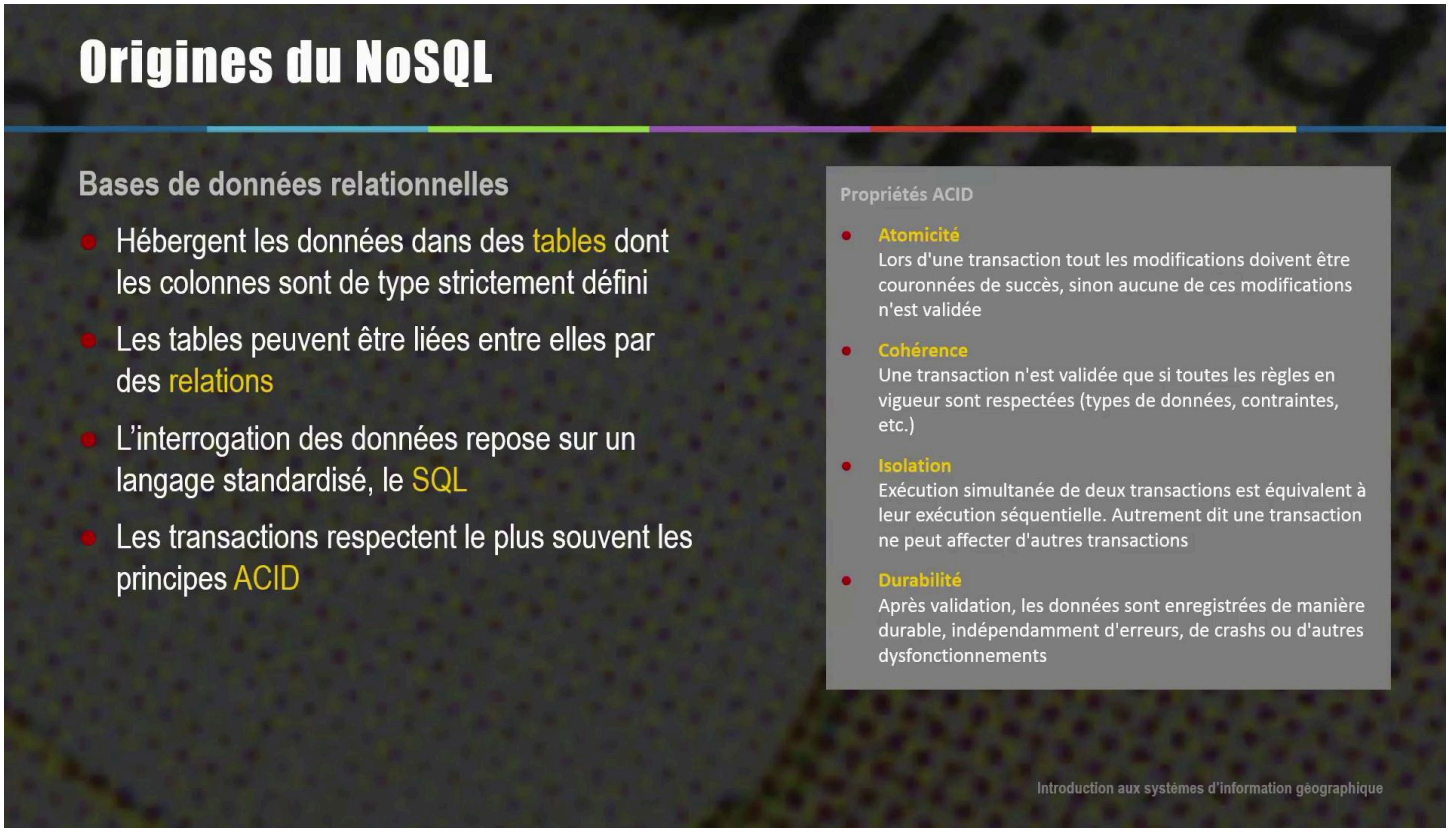


Dans cette leçon nous parlerons tout d'abord des origines du NoSQL puis nous passerons en revue les différents types de bases de données NoSQL et les systèmes de gestion de bases de données qui leur sont associés avant de concentrer notre attention plus particulièrement sur MongoDB qui est un type de base de données NoSQL. Et pour terminer, nous verrons comment les fonctions CRUD donc Create, Read, Update, Delete, sont implémentées dans un système comme MongoDB. Nous avons vu dans la leçon portant sur les bases de données relationnelles que ce type de base de données stocke les données dans des tables dont les colonnes ont des types bien définis, que ces tables peuvent être liées entre elles par des relations, que l'interrogation des données repose sur un langage standardisé, le SQL, et finalement que les transactions effectuées dans ces tables répondent en général aux principes ACID à savoir atomicité, cohérence, isolation et durabilité. Atomicité signifiant que toutes les modifications d'une transaction doivent être effectuées avec succès, faute de quoi l'ensemble de ces modifications doit être invalidé.

Notes

Summary

1m 26s



Cohérence, qu'une transaction n'est validée que si toutes les règles en vigueur sont respectées sur le type de données, sur les contraintes, les primaires, les valeurs nulles, possibles ou pas, etc. Isolation signifie que l'exécution simultanée de deux transactions est équivalente à leur exécution séquentielle donc une transaction n'affecte pas l'autre transaction et la durabilité signifie qu'après validation les données sont enregistrées de manière durable, indépendamment d'erreurs, de crash ou d'autres dysfonctionnements qui peuvent survenir.

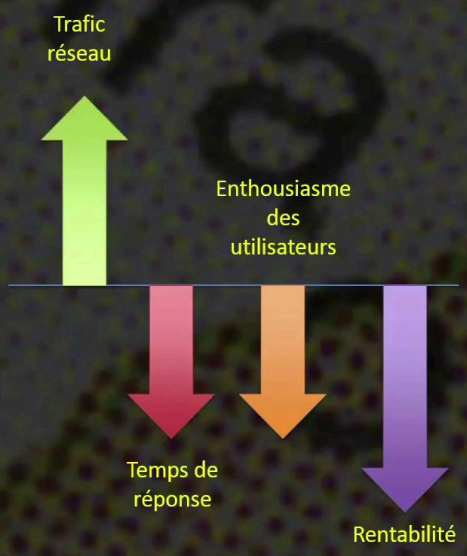
Notes

Summary

2m 53s

Origines du NoSQL

- Les services Web contemporains (type Google, Amazon, Facebook, LinkedIn, etc.) gèrent de très gros volumes de données et doivent répondre à de brusques afflux de trafic
- Nécessité de maintenir les fonctionnalités et les performances en cas de forte demande (scalabilité ou évolutivité)



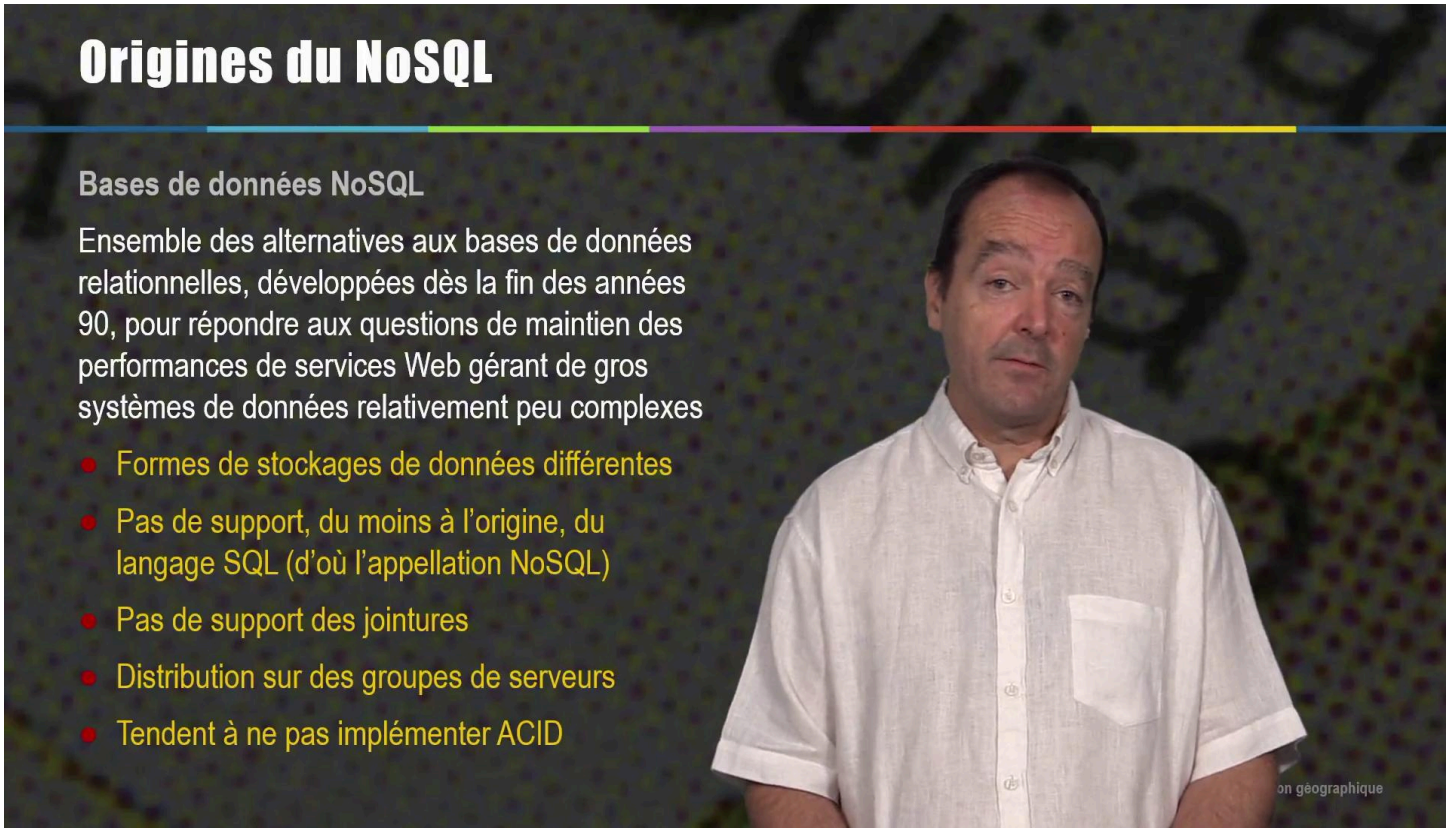
Introduction aux systèmes d'information géographique

La seconde génération web a vu apparaître des services qui gèrent de très gros volumes de données comme Google, Amazon, Facebook, LinkedIn, etc et qui doivent faire face à de brusques afflux de traffic ponctuels. Afflux de traffic signifie généralement diminution du temps de réponse, diminution de l'enthousiasme des utilisateurs et perte de rentabilité, d'où la nécessité de maintenir les fonctionnalités et les performances en cas de forte demande, caractéristique qu'on appelle la scalabilité ou l'évolutivité.

Notes

Summary

3m 24s



Ce maintien des fonctionnalités et des performances peut être obtenu par accroissement de puissance et on parle alors de scalabilité verticale ou par démultiplication des noeuds et distribution des services sur les noeuds du réseau et on parle alors de scalabilité horizontale. Un meilleur ajustement des logiciels aux types de données gérées permet également d'accélérer le traitement des données et de maintenir la performance.

Notes

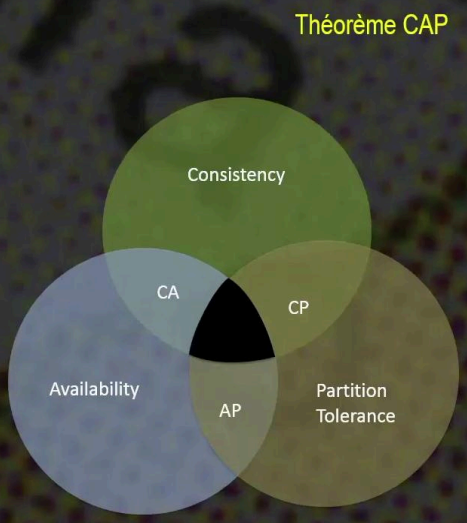
Summary

3m 59s

Origines du NoSQL

Un système distribué, soit une collection de nœuds interconnectés partageant des données, ne peut jamais satisfaire simultanément que deux des trois critères suivants (théorème CAP):

- Cohérence (C). Tous les nœuds du système voient exactement les mêmes données au même moment
- Disponibilité (A). Garantie que toutes les requêtes reçoivent une réponse quand à leur succès ou échec
- Tolérance au Partitionnement (P). Le système reste fonctionnel en dépit d'un partitionnement aléatoire dû à des ruptures de communication



Introduction aux systèmes d'information géographique

On voit donc que les bases de données NoSQL regroupent un ensemble d'alternatives aux bases de données relationnelles qui ont été développées dès la fin des années 90 pour répondre au besoin de maintien des performances de certain gros acteurs du web qui gèrent de très grosses bases de données, de complexité faible à modérée. Les principales caractéristiques des bases de données NoSQL se trouvent tout d'abord dans la forme de stockage des données qui diffère pas mal de ce à quoi on est habitué dans le monde des bases de données relationnelles dans le fait de ne pas fournir, en tout cas à l'origine, d'implémentation du langage SQL, dans le fait de ne pas apporter de support aux jointures, donc on ne peut pas faire de jointure dans les bases de données NoSQL, de fonctionner en système distribué, c'est-à-dire de distribuer les données sur plusieurs serveurs et finalement une tendance à ne pas implémenter les principes ACID. Le principe de la scalabilité horizontale, et donc de vouloir distribuer le service ou les données sur les nombreuses machines constituant un réseau, donc sur un ensemble de noeuds, conduit tout droit à ce théorème dit "théorème CAP" qui stipule que seul 2 des 3 critères de cohérence de disponibilité et de tolérance au partitionnement peuvent être satisfaits simultanément.

Notes

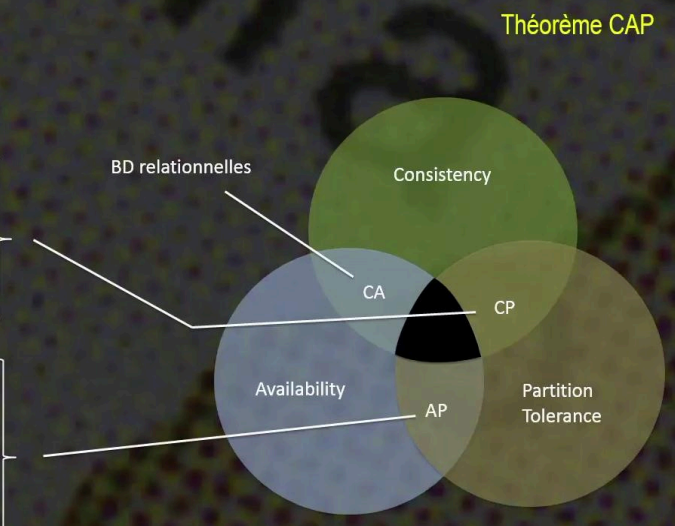
Summary

4m 25s

Origines du NoSQL

Les bases de données distribuées subissent forcément des partitionnements, si bien qu'il faut choisir de privilégier la cohérence ou la disponibilité

- Dans le premier cas (CP), cela signifie que l'on attend une synchronisation des nœuds, au prix éventuel d'un timeout ou d'une erreur (disponibilité pas respectée)
- Dans le second cas (AP), cela signifie que l'on renvoie la dernière version disponible au niveau du nœud, et que l'on accepte que tous les nœuds peuvent ne pas être «à jour» et donc renvoyer éventuellement des valeurs différentes



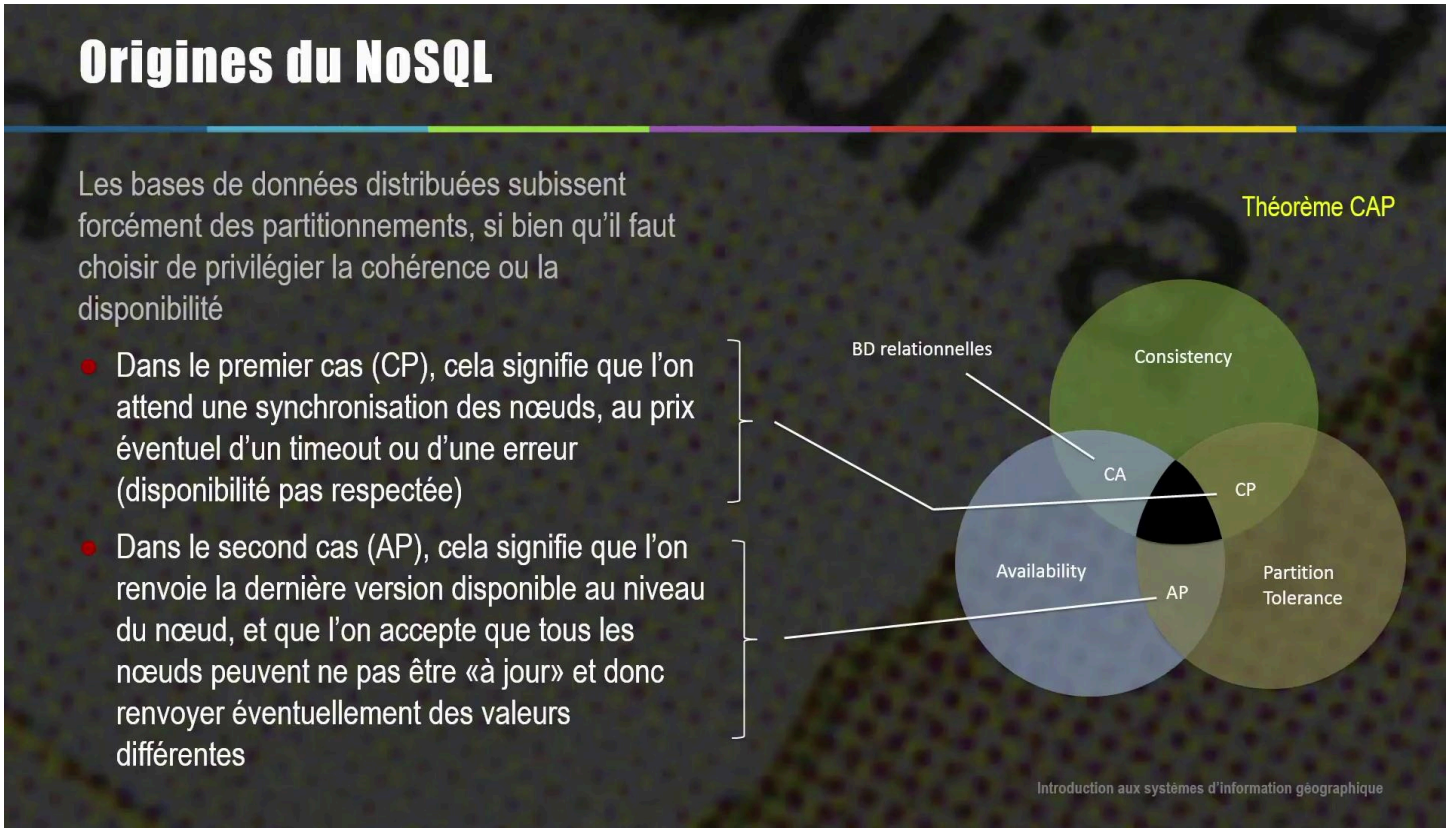
Introduction aux systèmes d'information géographique

Cohérence signifiant que tous les noeuds d'un système voient exactement les mêmes données au même moment. Disponibilité que chaque fois que l'on envoie une requête vers l'un des noeuds du système on est certain de recevoir une réponse soit en terme de succès ou d'échec de la requête mais on reçoit une réponse. La notion de tolérance au partitionnement qui exprime l'idée que dans un système distribué il arrive fréquemment que les différents noeuds puissent être isolés les uns des autres soit parce qu'un noeud ne fonctionne plus, il a été victime d'un crash, une coupure de courant, des ruptures de communication, la communication ne passe plus, etc. Dans le cas des systèmes NoSQL où l'on essaye de répartir la charge et l'information sur les nombreux noeuds d'un réseau, le partitionnement est une réalité, si bien qu'en fait il faut choisir de privilégier la cohérence ou la disponibilité. Dans le premier cas les systèmes que l'on appelle "CP", donc plutôt cohérence partitionnement, cela signifie que l'on attend une synchronisation des noeuds avant d'assurer la réponse avec le risque d'avoir de temps en temps des timeout ou des erreurs, des messages d'erreur, et donc le critère de disponibilité n'est pas respecté.

Notes

Summary

6m 06s

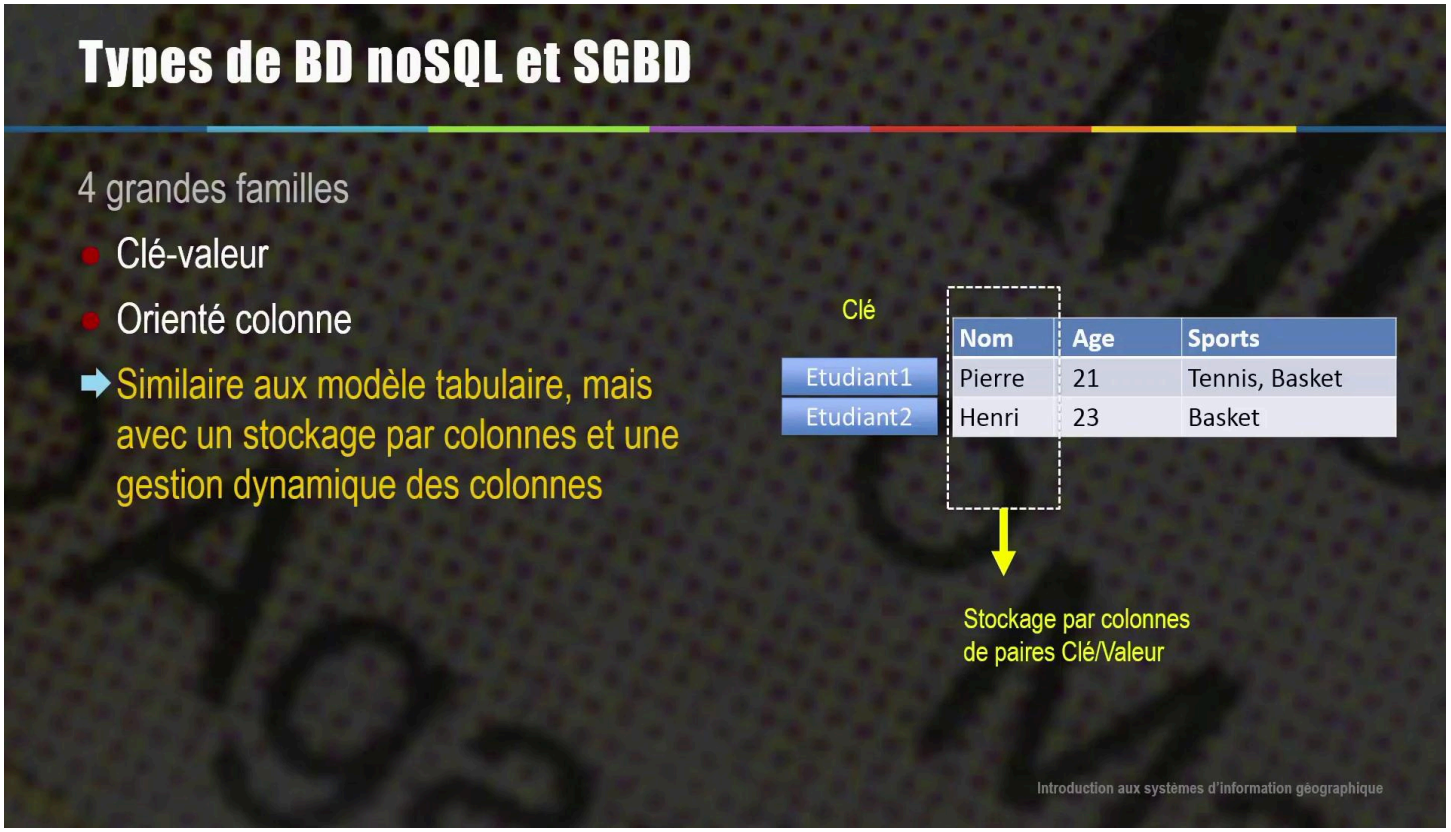


Ou alors on va vers des solutions AP, donc où on privilégie la disponibilité. Dans ce cas-là le noeud que l'on interroge va renvoyer la dernière version disponible d'une donnée au niveau d'un noeud, ce qui veut dire que l'on accepte que tous les noeuds puissent ne pas toujours être à jour et des fois renvoyer des valeurs différentes. Après, toute la question est de savoir quelle est la durée de mise à jour tolérable. On voit que les bases de données relationnelles, qui de par leur nature même, de par l'existence de relations entre les tables, se prêtent mal à une distribution des données sur plusieurs machines différentes, se trouvent en général dans le registre CA, donc cohérence-disponibilité, mais ne se prêtent pas du tout à une scalabilité horizontale par répartition sur un réseau.

Notes

Summary

7m 31s



Si l'on regarde d'un peu plus près le monde des bases de données NoSQL on s'aperçoit qu'on peut distinguer quatre grandes familles de bases de données NoSQL qui se distinguent par la manière dont elles stockent l'information. La première de ces familles est constituée par les bases de données de type clé-valeur qui sont donc basées sur des dictionnaires permettant d'accéder à la valeur d'un objet par l'intermédiaire d'une clé qui doit être unique. Comme exemple, on peut voir ici que la clé peut être représentée par un prénom, Pierre ou Henri, qui donne accès à une série de valeurs, le type l'activité, l'âge, le type d'activité sportive, etc. Les principaux systèmes de gestion de bases de données qui implémentent ce type clé-valeur sont Redis et Voldemort qui a été développé par LinkedIn pour gérer ses bases de données. Le second type de base de données NoSQL est le type orienté colonne qui s'apparente un petit peu au modèle relationnel puisque les données peuvent être représentées par des tables. Simplement, le stockage est organisé par colonnes sur une base clé-valeur où l'on associe pour chaque colonne la clé avec la valeur de cette clé dans cette colonne.

Notes

Summary

8m 39s

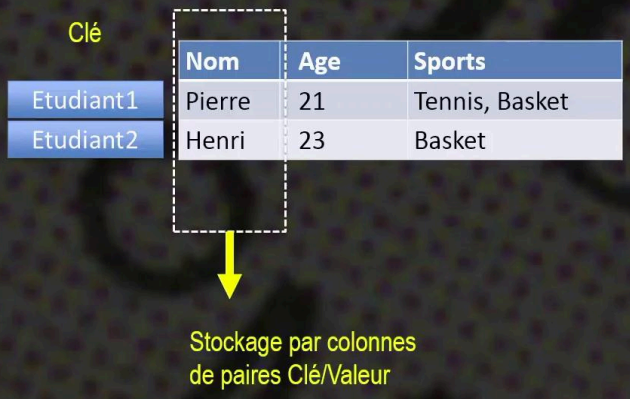
Types de BD noSQL et SGBD

4 grandes familles

- Clé-valeur
- Orienté colonne

➡ Similaire aux modèle tabulaire, mais avec un stockage par colonnes et une gestion dynamique des colonnes

➡ SGBD
Cassandra, Hbase, BigTable, Vertica



L'avantage de ce système est de permettre une gestion très dynamique d'un nombre très élevé de colonnes. Les principaux systèmes de gestion de base de données qui implémentent ce modèle sont Cassandra, Hbase, BigTable et Vertica.

Notes

Summary

9m 56s

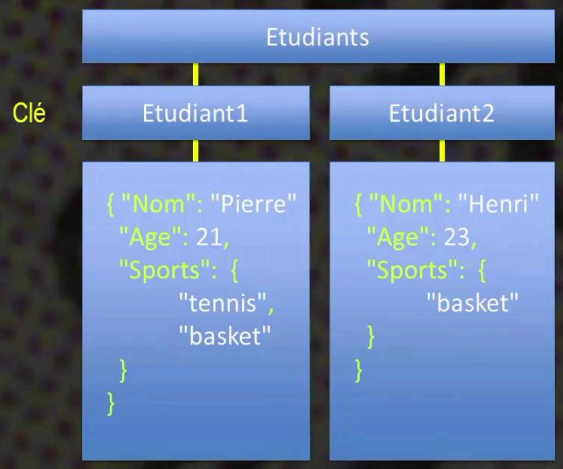
Types de BD noSQL et SGBD

4 grandes familles

- Clé-valeur
- Orienté colonne
- Orienté document

➡ Collections de documents de type JSON ou XML selon un paradigme clé-valeur

➡ SGBD
CouchDB, SimpleDB, MongoDB

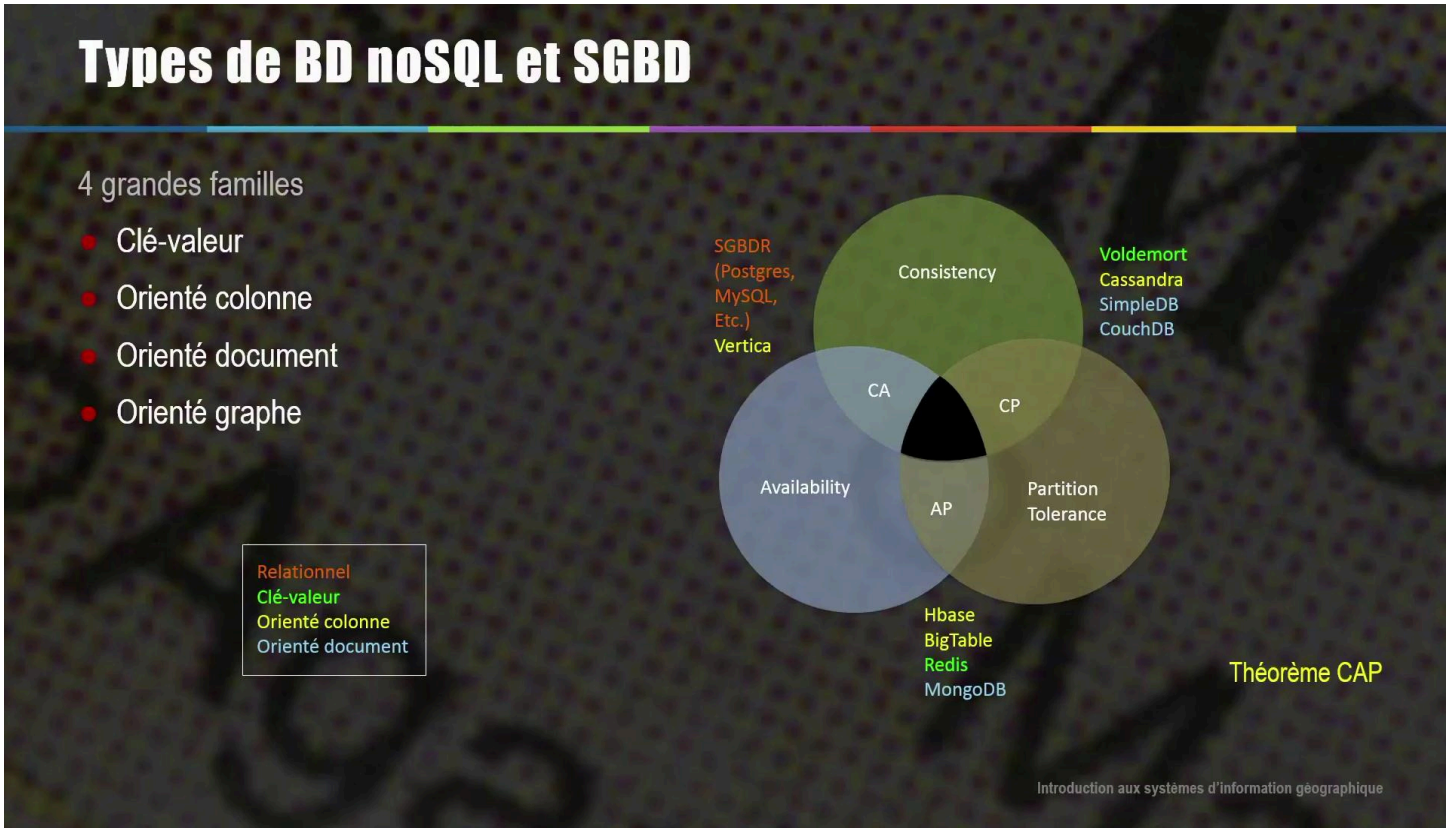


Le troisième type de base de données NoSQL que l'on appelle "orienté document" repose également sur un paradigme clé-valeur en associant à la clé des documents qui sont de type JSON ou XML. Donc on a ici toujours le même exemple de Pierre et de Henri avec étudiant 1 et étudiant 2 qui sont les clés et derrière une syntaxe JSON qui donne les attributs de l'objet. Les systèmes de gestion de bases de données qui implémentent ce modèle sont principalement CouchDB, SimpleDB et MongoDB.

Notes

Summary

10m 14s

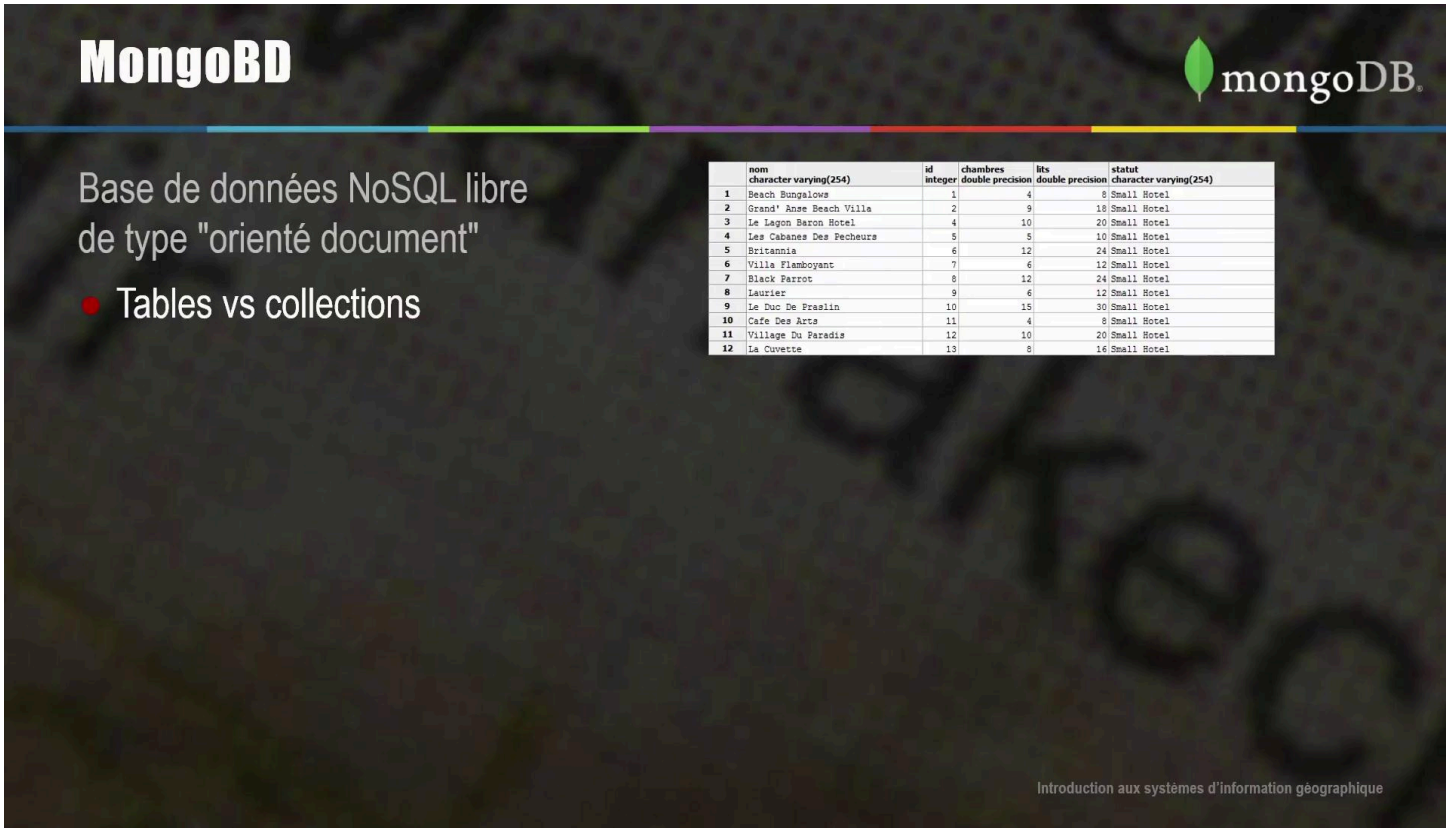


Et finalement, les bases de données NoSQL de type "orienté graphe" qui se basent sur la théorie des graphes, sur les notions de noeud, de relation et de propriété pour stocker l'information, ici relative aux étudiants Pierre et Henri. Le principal, et seul quasiment, système de gestion de bases de données, le seul système un peu abouti, est Neo4J. Dans cette figure qui reprend les idées du théorème CAP on a représenté les différents types de système de gestion de bases de données en fonction de leurs caractéristiques, de leur positionnement dans ce diagramme en représentant en rouge-orangé les systèmes de gestion de bases de données relationnelles, notamment Postgres, MySQL, etc. Les systèmes en vert, de type clé-valeur, comme Voldemort et Redis. Les systèmes orientés colonne en jaune, avec Cassandra du côté CP, HBase et BigTable du côté AP. Et finalement en bleu les systèmes SGBD orientés document où l'on voit que MongoDB est un système qui est du côté de la disponibilité alors que ses alter-egos, SimpleBD et CouchDB, sont plutôt du côté de la cohérence.

Notes

Summary

10m 52s



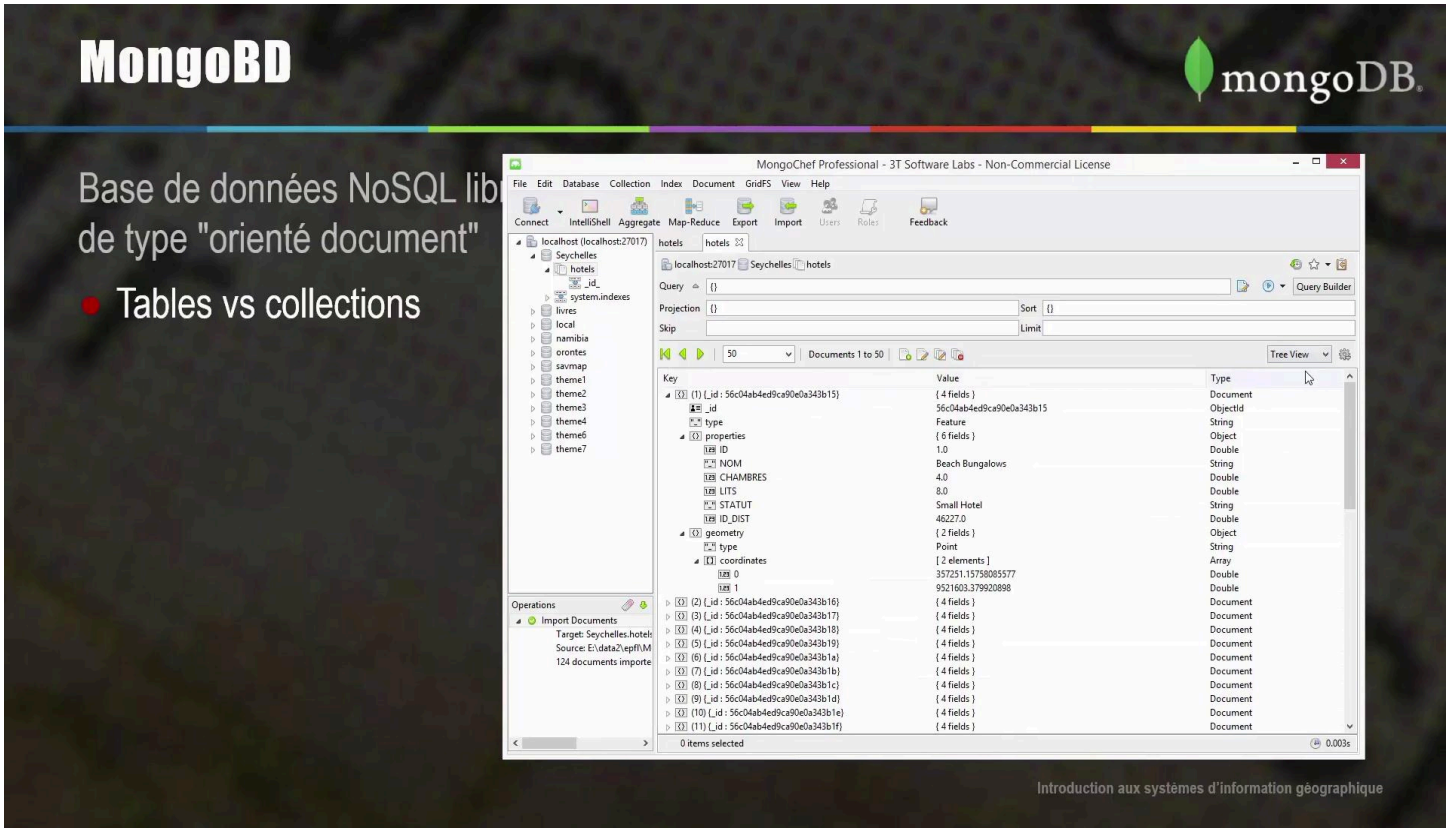
	nom character varying(254)	id integer	chambres double precision	lits double precision	statut character varying(254)
1	Beach Bungalows	1	4	8	Small Hotel
2	Grand' Anse Beach Villa	2	9	18	Small Hotel
3	Le Lagon Baron Hotel	4	10	20	Small Hotel
4	Les Cabanes Des Pecheurs	5	5	10	Small Hotel
5	Britannia	6	12	24	Small Hotel
6	Villa Flamboyant	7	6	12	Small Hotel
7	Black Parrot	8	12	24	Small Hotel
8	Laurier	9	6	12	Small Hotel
9	Le Duc De Praslin	10	15	30	Small Hotel
10	Cafe Des Arts	11	4	8	Small Hotel
11	Village Du Paradis	12	10	20	Small Hotel
12	La Cuvette	13	8	16	Small Hotel

MongoDB est donc une base de données NoSQL libre de type orienté document. Dans cette base de données la notion de table qu'on avait dans les bases de données relationnelles est remplacée par la notion de collection.

Notes

Summary

12m 24s



On a donc ici l'exemple de la table Postgres des hôtels des Seychelles. Si on regarde comment ces données sont gérées dans Mongo on a ici un interface utilisateur qui permet de voir cette collection d'objets, donc les hôtels, 124 objets dans la collection et puis pour chaque objet une structure arborescente identifiant le type d'objet, la collection de ses propriétés, la géométrie et dans la géométrie les coordonnées. Cette représentation arborescente peut être transformée en une représentation au format JSON où l'on retrouve cette idée de collection dans la syntaxe propre au format JSON. Elle peut également être représentée dans une vue tabulaire simplement avec des étages successifs que l'on peut parcourir pour accéder aux coordonnées ou aux propriétés des objets.

Notes

Summary

12m 39s

MongoBD

Base de données NoSQL libre de type "orienté document"

- Tables vs collections
- Schéma de données
- Dénormalisation et jointure

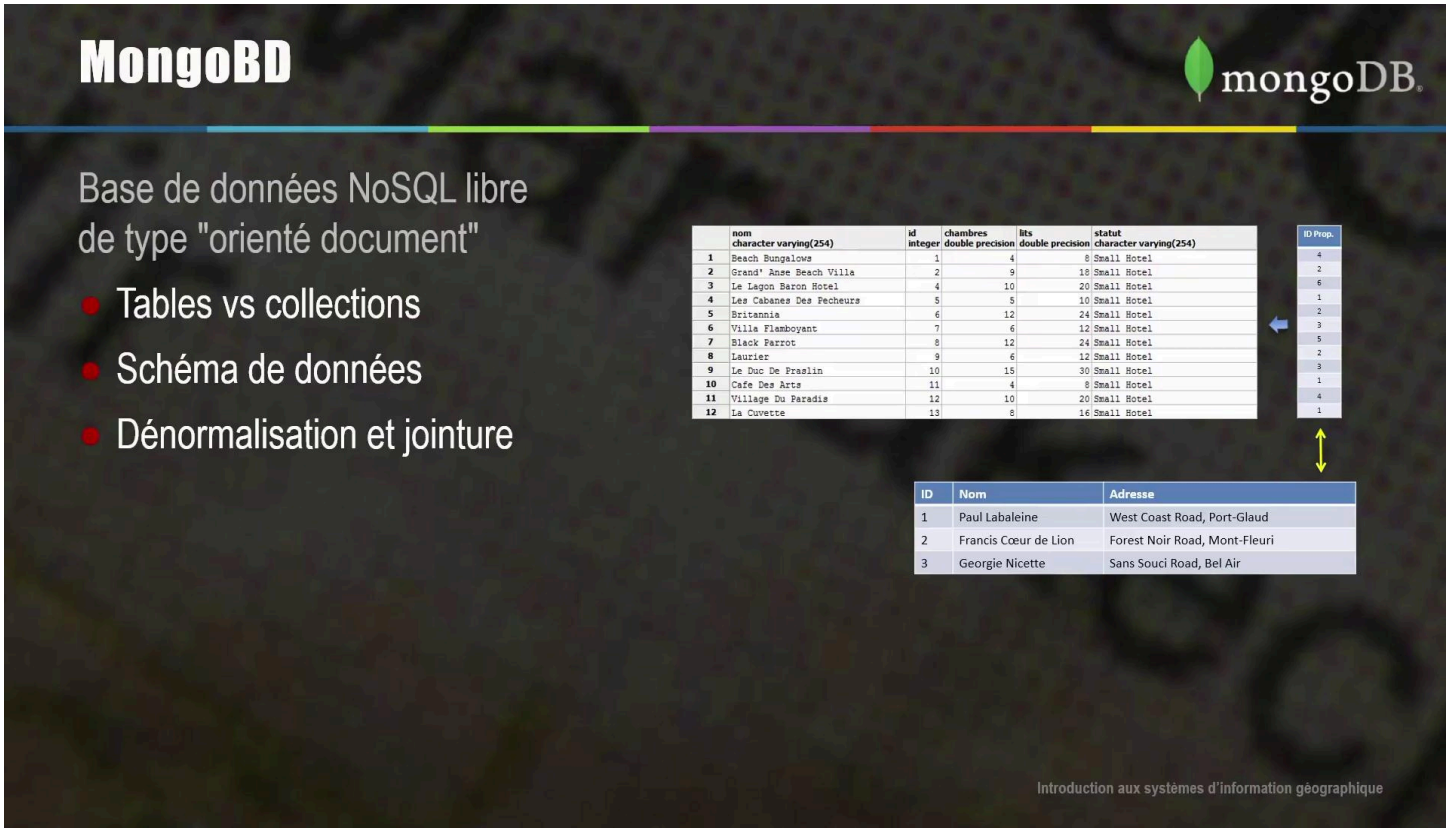
	nom character varying(254)	id integer	chambres double precision	lits double precision	statut character varying(254)
1	Beach Bungalows	1	4	8	Small Hotel
2	Grand' Anse Beach Villa	2	9	18	Small Hotel
3	Le Lagon Baron Hotel	4	10	20	Small Hotel
4	Les Cabanes Des Pecheurs	5	5	10	Small Hotel
5	Britannia	6	12	24	Small Hotel
6	Villa Flamboyant	7	6	12	Small Hotel
7	Black Parrot	8	12	24	Small Hotel
8	Laurier	9	6	12	Small Hotel
9	Le Duc De Praslin	10	15	30	Small Hotel
10	Cafe Des Arts	11	4	8	Small Hotel
11	Village Du Paradis	12	10	20	Small Hotel
12	La Cuvette	13	8	16	Small Hotel

Second élément important la notion de schéma de données qui, dans une base de données relationnelle, est très importante. Elle est incontournable. Dans un système comme Mongo on est libre de stocker les données absolument comme on veut de mettre tout ce qu'on veut dans une collection. Par contre, si l'on veut pouvoir utiliser les données stockées dans une base de données Mongo pour représenter des objets dans une carte on a intérêt à ce que ces objets suivent une certaine structure pour retrouver les coordonnées du point et les propriétés de l'objet. La meilleure manière de retrouver cette structure c'est de stocker au départ des objets qui ont cette structure. En d'autres termes, la base de données Mongo n'est pas contraignante du point de vue schématisation des données par contre de négliger la schématisation rend la vie beaucoup plus compliquée. Dénormalisation et jointure.

Notes

Summary

13m 41s

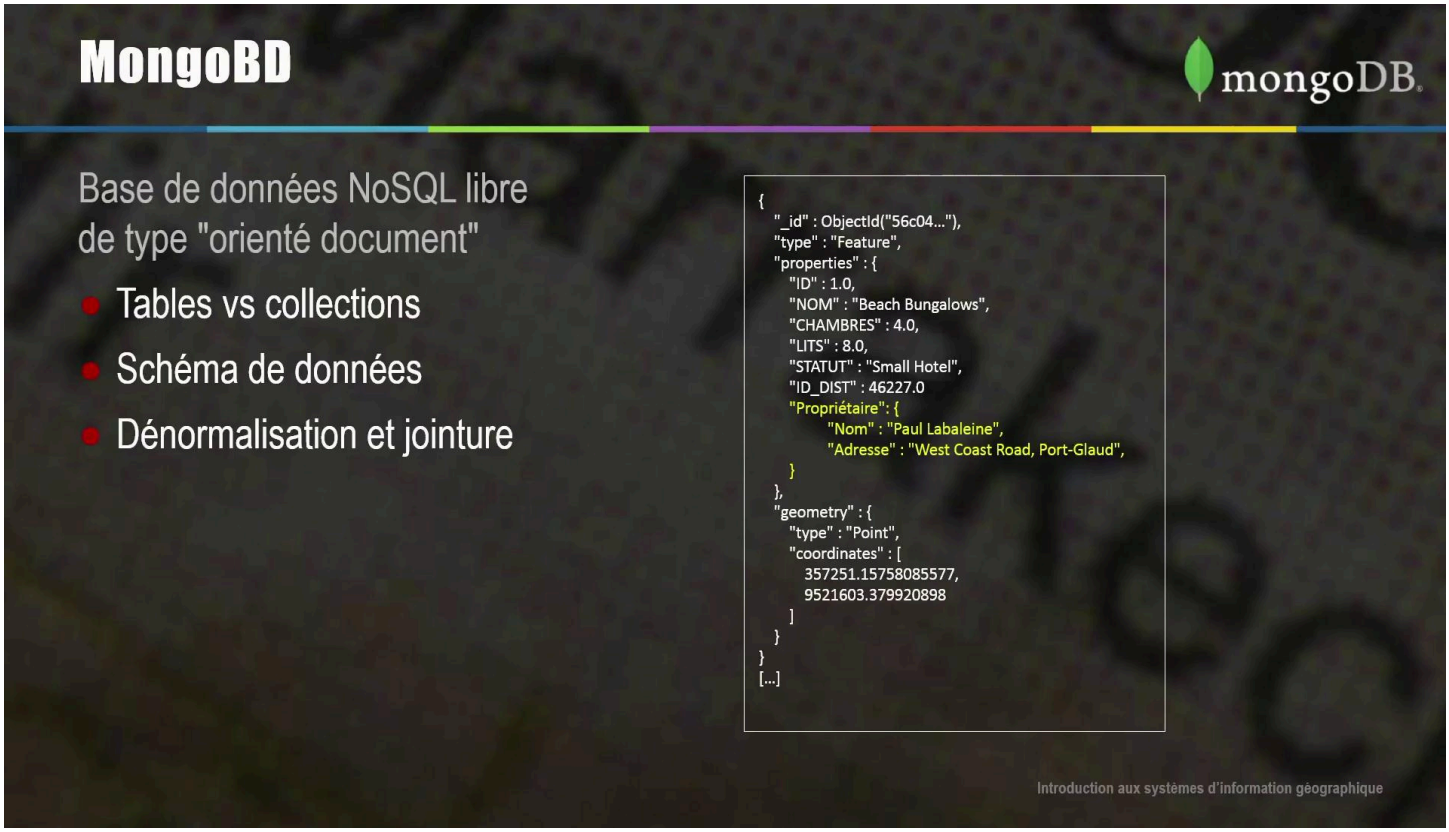


Dans un système de base de données relationnelle, lorsqu'on veut ajouter un élément d'information, comme dans le cas des hôtels des Seychelles, si l'on souhaite ajouter l'information relative aux propriétaires de ces hôtels, comme plusieurs hôtels peuvent avoir le même propriétaire pour éviter la redondance de données on créerait une table séparée indirecte qui recense les propriétaires et on ajouterait dans la table d'origine une colonne avec l'identifiant des propriétaires. On établirait ainsi une jointure entre ces deux tables. Ce processus de séparation de données c'est ce qu'on appelle, dans le monde des bases de données relationnelles, la normalisation. Notons au passage que l'une des faiblesses du monde relationnel, qui a déjà été évoquée lorsqu'on a présenté ce domaine, la difficulté qu'il y aurait ici à stocker l'information dans le cas où un hôtel aurait plusieurs propriétaires.

Notes

Summary

14m 43s

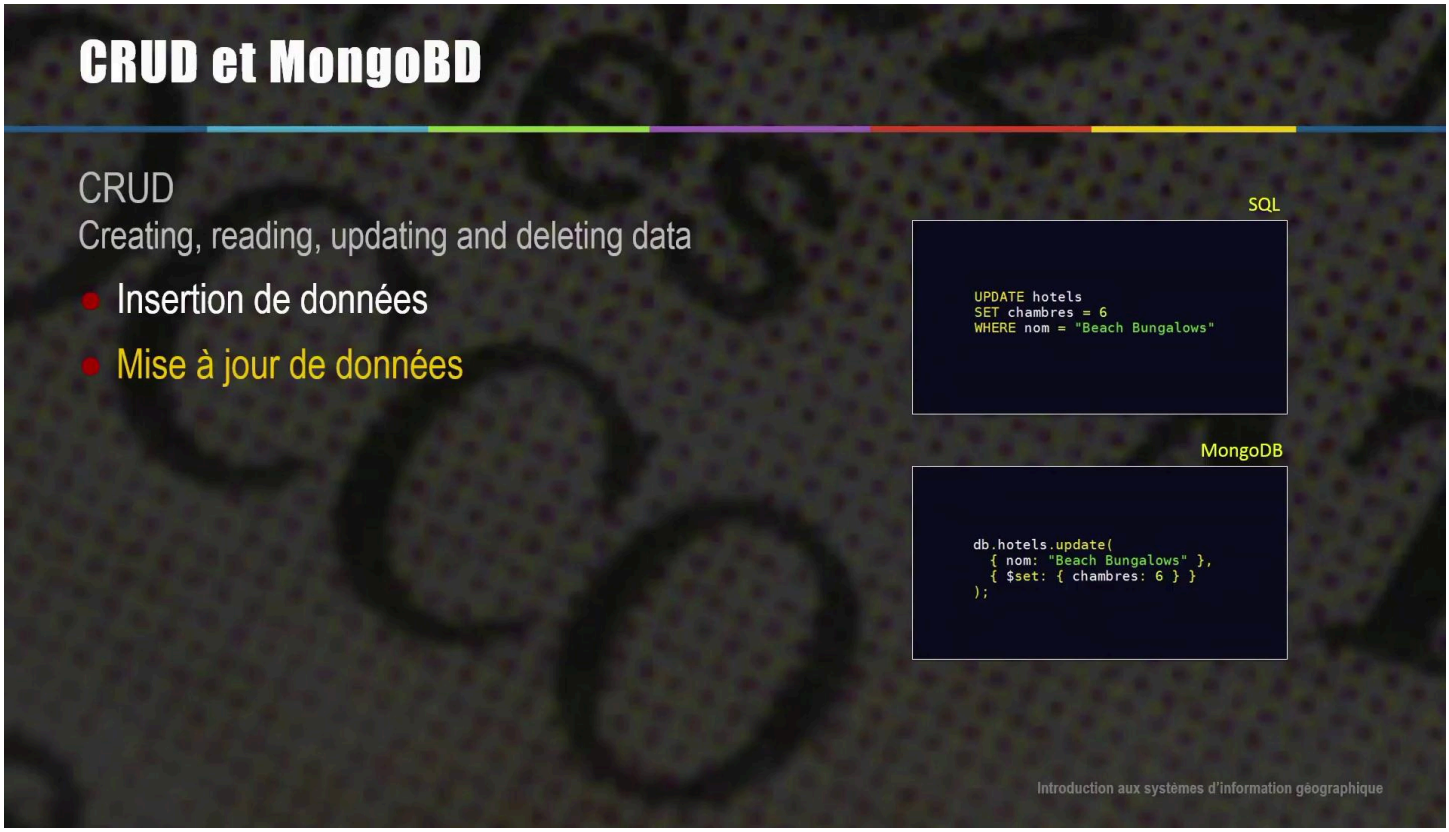


Dans le cas de Mongo on pourrait bien entendu procéder d'une manière analogue en ajoutant dans le document JSON un champ ID du propriétaire et puis en ayant une liste toujours au format JSON de propriétaires avec leurs adresses. Notons au passage que la syntaxe JSON permet facilement de représenter le cas de plusieurs propriétaires. En réalité, dans le monde NoSQL comme on n'implémente pas les jointures on préfère lister explicitement les éléments d'information liés au propriétaire, quitte à vivre avec une certaine redondance. C'est pour cette raison que les systèmes NoSQL se prêtent particulièrement bien à gérer des grands volumes de données mais des données qui ont une faible complexité avec peu de relations tabulaires. Dans le cas où on a des systèmes de données complexes avec des tables qui sont liées les unes aux autres, le système de base de données relationnelle reste plus intéressant en général.

Notes

Summary

15m 32s

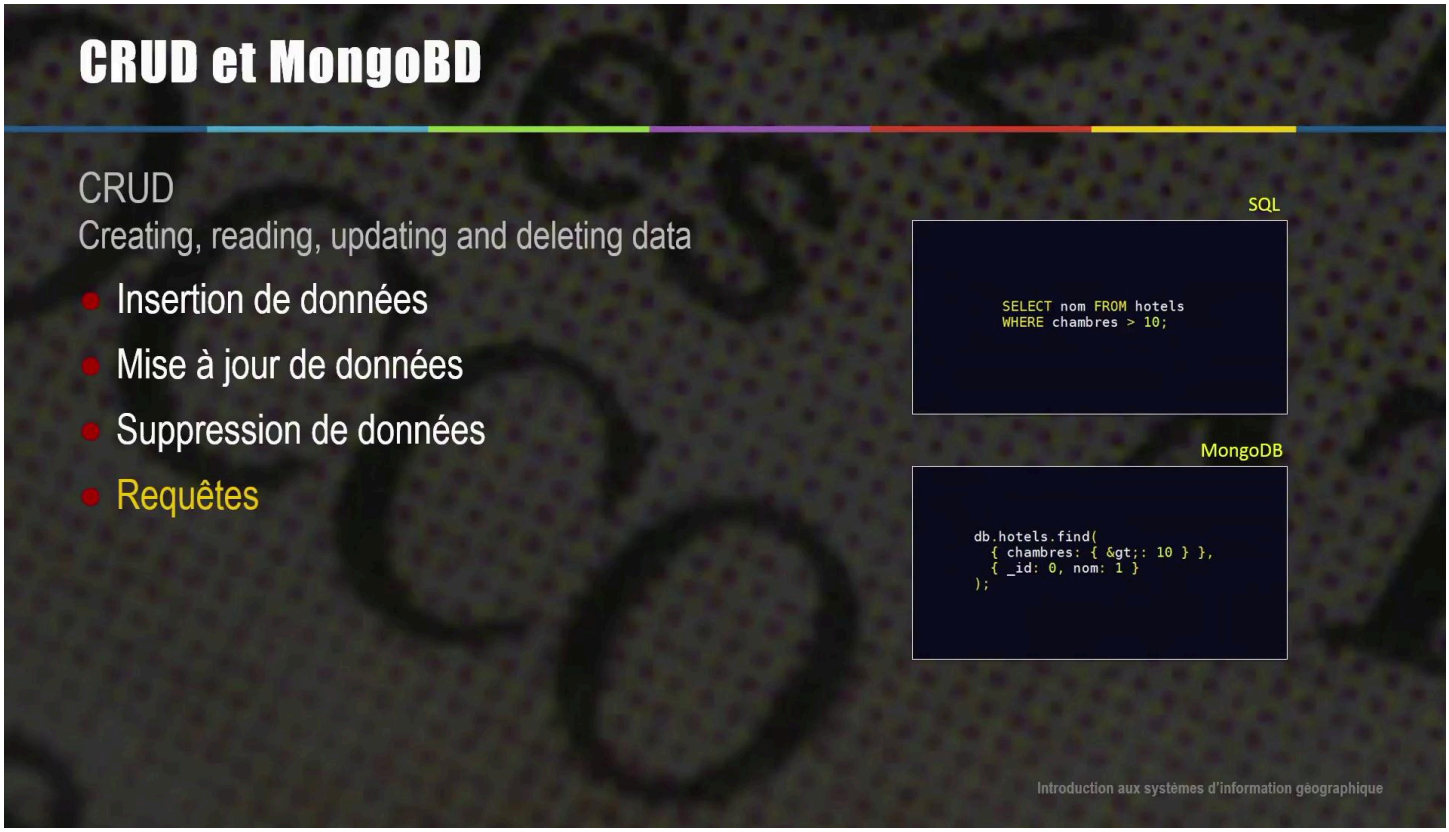


Pour terminer nous allons voir quelques exemples d'implémentation des fonctions de base des bases de données. Donc les fonctions de création, de lecture, de mise à jour et de suppression de données que l'on regroupe sous l'acronyme CRUD. On commence avec l'insertion de données, on va systématiquement comparer la syntaxe SQL avec la syntaxe que l'on retrouve dans Mongo. Dans le SQL, l'introduction d'un nouveau jeu de données dans la table des hôtels implique les mots-clés INSERT INTO, la liste des colonnes et puis la liste des valeurs correspondant à ces colonnes. Dans le cas de Mongo on a quelque chose d'un peu équivalent avec simplement les mots-clés DB qui réfèrent à la base de données en cours, hôtels c'est la collection visée et on insère dans cette collection un nouvel objet qui a trois attributs le nom, le nombre de chambre, le nombre de lit et le statut avec les valeurs de ces 4 attributs. Mise à jour de données. Dans le cas SQL, le mot-clé UPDATE, la table hôtels et puis le mot-clé SET, la variable que l'on souhaite changer, donc la variable chambre qui passe à 6 et la clause conditionnelle pour identifier l'hôtel sur lequel le nombre de chambres a augmenté à 6.

Notes

Summary

16m 43s

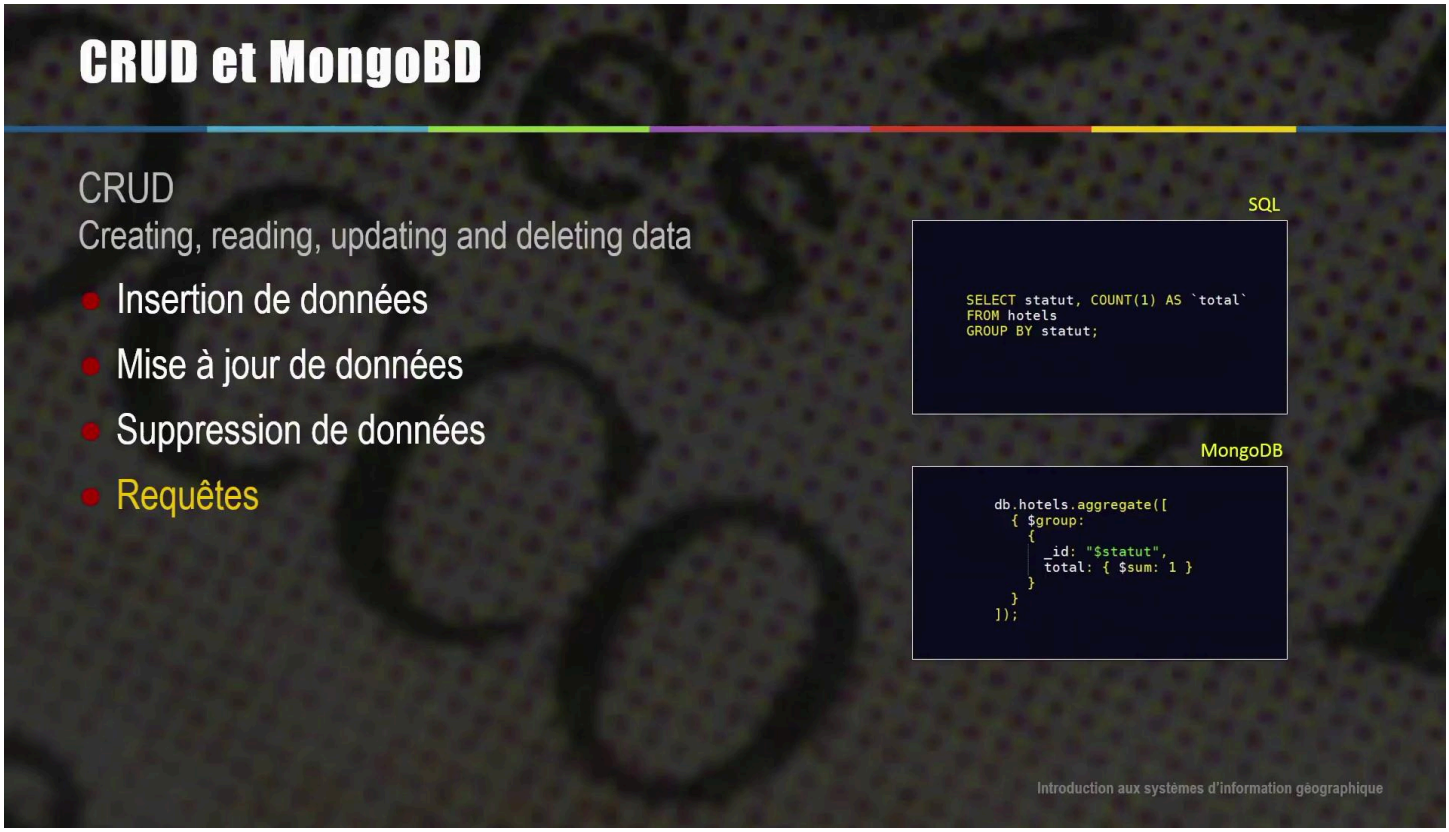


L'équivalent du côté Mongo avec le mot-clé UPDATE également qui s'applique sur la collection hôtels et puis simplement la référence à l'objet qui va changer, donc on va rechercher des objets dont le nom correspond à "beach bungalows" et on va appliquer un set sur la variable chambre qui prend la valeur 6. Donc dans l'esprit, quelque chose d'assez similaire au langage SQL. Pour la suppression de données le DELETE FROM du langage SQL, suivi du nom de la table et puis de la clause conditionnelle que l'on retrouve à peu près à l'identique dans Mongo avec la fonction REMOVE appliquée à la collection hôtels, on supprime le document qui a pour attribut-nom "beach bungalows". Enfin quelques exemples de requêtes plus générales à commencer par la requête de sélection des hôtels dont le nombre de chambres est supérieur à 10. Dans le SQL, SELECT FROM classique avec une clause conditionnelle. Dans Mongo on retrouve le mot-clé FIND qui permet de rechercher des objets. Le critère de recherche dans la première ligne, le champ chambre et puis l'opérateur "plus grand que" caractérisé par l'esperluette qui précède le mot-clé GT et puis le critère 10 et dans la deuxième ligne les choses que l'on souhaite renvoyer comme l'identifiant de l'objet est renvoyé par défaut, on doit mettre un zéro si on ne veut pas avoir l'identifiant et puis un 1 pour renvoyer le nom.

Notes

Summary

18m 09s



Donc c'est 0, 1, true, false. Un second exemple de requête basique similaire, donc compter le nombre d'objets renvoyés par la requête de sélection des hôtels dont le nombre de chambres est supérieur à 10 avec quelque chose d'assez équivalent dans Mongo, une fonction "count" appliquée à la collection des hôtels et portant sur le fait que les chambres... le nombre de chambre est égal à 10. Dernier exemple de requête un peu plus élaboré où l'on cherche à lister le nombre d'hôtels qu'il y a pour chaque statut. Dans le SQL une requête classique avec un groupement par statut et on compte dans chaque statut le nombre de candidat. Dans le cas de MongoDB on utilise le mot-clé la fonction d'agrégation portant sur la collection des hôtels avec comme règle de groupement le statut et comme règle de calcul la somme. Ces quelques exemples montrent donc que dans un logiciel NoSQL comme Mongo qui n'implémente pas le langage SQL on dispose quand même d'un langage de requête qui permet d'interroger les données de manière efficace.

Notes

Summary

19m 50s

CRUD et MongoBD

CRUD

Creating, reading, updating and deleting data

- Insertion de données
- Mise à jour de données
- Suppression de données
- Requêtes

SQL

```
SELECT statut, COUNT(1) AS `total`
FROM hotels
GROUP BY statut;
```

MongoDB

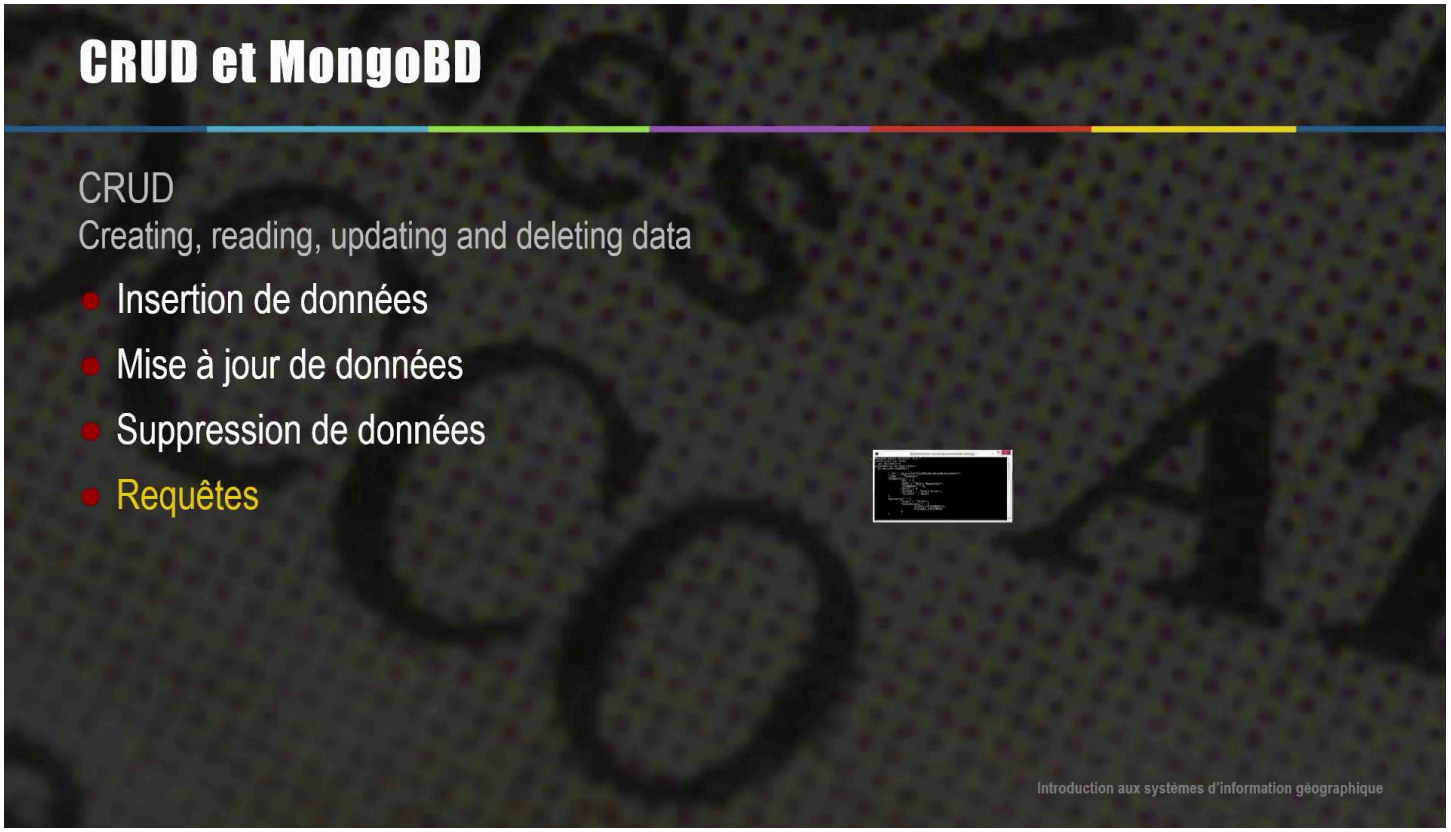
```
db.hotels.aggregate([
  { $group:
    {
      _id: "$statut",
      total: { $sum: 1 }
    }
  }
]);
```


Langage qui est d'ailleurs implémenté dans l'interface graphique.

Notes

Summary

21m 10s



Mais avant de passer à l'interface graphique regardons déjà comment se présentent les choses en ligne de commande qui est quand même le mode d'utilisation initial et principal de Mongo. Avec la commande Mongo on accède au shell de Mongo, USE SEYCHELLES pour dire qu'on veut utiliser la base de données Seychelles qui va s'appeler maintenant DB et DB HOTELS, donc en référence la collection des hôtels et on cherche le premier objet de la collection qui est l'hôtel Beach Bungalow, donc affiché en format JSON.

Notes

Summary

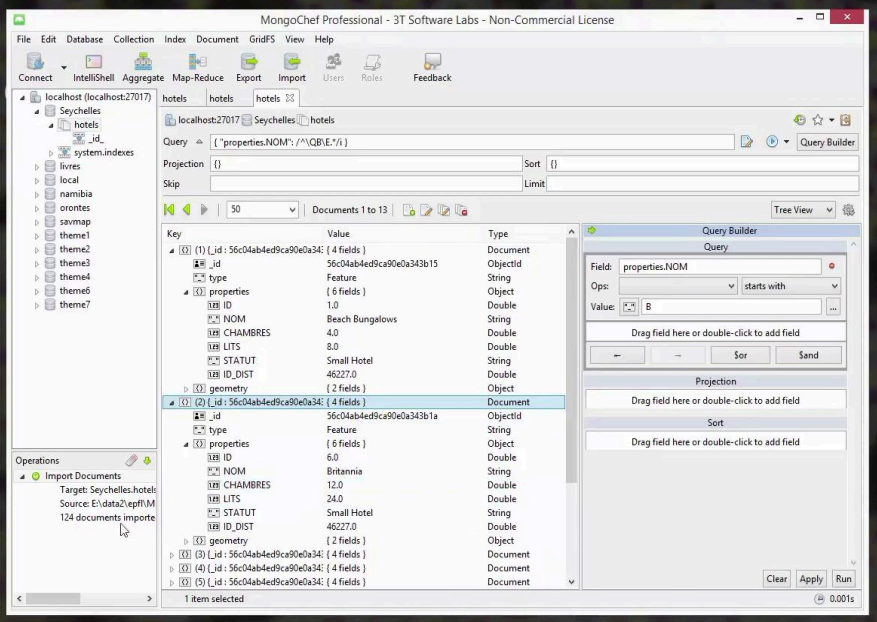
21m 18s

CRUD et MongoBD

CRUD

Creating, reading, updating and

- Insertion de données
- Mise à jour de données
- Suppression de données
- Requêtes



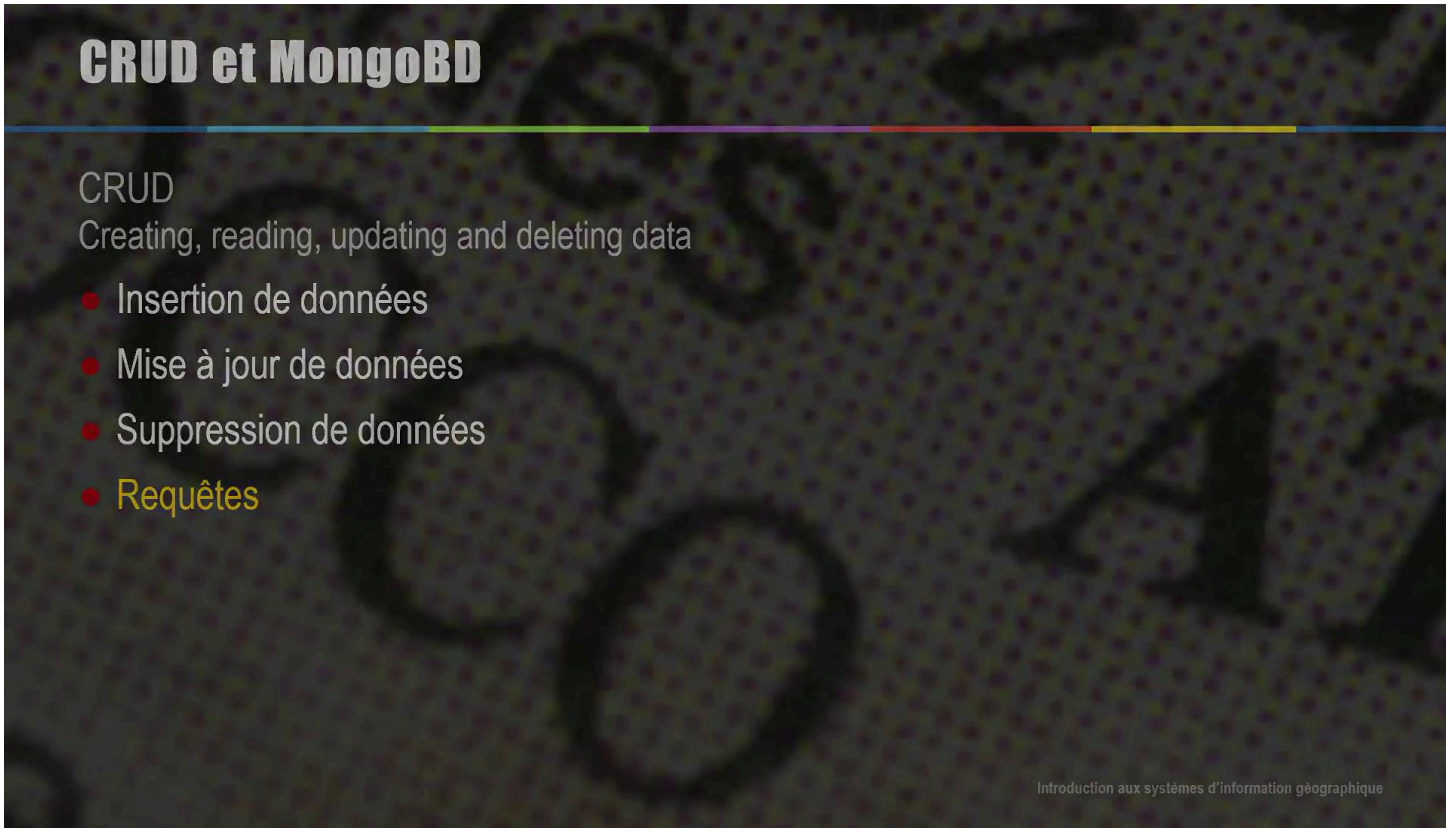
Introduction aux systèmes d'information géographique

Dans le cas d'un interface utilisateur graphique un peu plus élaboré on a un petit outil de création de requête, on peut par exemple glisser le champ nom et rechercher tous les hôtels dont le nom commence par B et on en trouve toute une série habituelle.

Notes

Summary

21m 50s



Voilà.

Notes

Summary

22m 20s