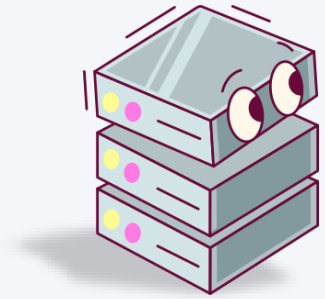
الكلية متعددة التخصصات - ورزازات
ⵜⴰⵖⵍⵉⵜ ⵜⴰⵅⵚⵚⵚⵉⵚⵜ – ⵍⴰⵡⴰⵣⴰⵜ
FACULTE POLYDISCIPLINAIRE DE OUARZAZATE

NOSQL : CASSANDRA



Présenter par :
Youssef MOKNIA
Youssef EL WALI
Halima BANANI
Asmaa AIT EL HAJ
Yassine AMZIR

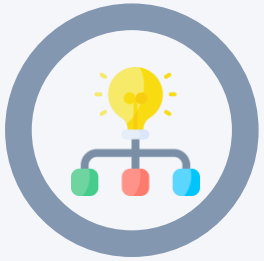
Encadrer par :
Mr. Charaf HAMIDI

2023-2024

PLAN

Introduction à Cassandra

Modèle de données



Théorème CAP

CQL : Cassandra Query Language

Architecture de Cassandra

Fonctionnalités clés

INTRODUCTION À CASSANDRA

Contexte et historique de Cassandra :

Cassandra émerge comme une solution innovante dans le paysage des bases de données distribuées, offrant une alternative robuste aux systèmes traditionnels.

Son développement trouve ses racines chez **Facebook** en 2008, où il a été conçu pour répondre aux besoins de scalabilité et de disponibilité de l'entreprise dans le traitement des données en temps réel






INTRODUCTION À CASSANDRA

Définition de Cassandra :

Apache Cassandra est un système de gestion de base de données (SGBD) de type NoSQL conçu pour gérer des quantités massives de données sur un grand nombre de serveurs, assurant une haute disponibilité en éliminant les points de défaillance unique.






INTRODUCTION À CASSANDRA

Objectifs de Cassandra :

Apache CASSANDRA

Cassandra a été créée dans le but de répondre aux besoins croissants des entreprises en matière de gestion de données massives et diverses. Son objectif principal est de permettre aux entreprises de gérer efficacement des volumes importants de données, qu'elles soient structurées, semi-structurées ou non structurées.






INTRODUCTION À CASSANDRA

Importance de Cassandra dans le domaine des bases de données distribuées

- En tant que système distribué, Cassandra n'utilise pas de maître. Tous les clusters sont dotés d'autorisations identiques et peuvent traiter toutes les requêtes de bases de données qui leur sont adressées. Cela permet d'en augmenter considérablement les performances.

- Les données sont distribuées sur des nœuds. La facilité d'ajout de nœuds assure la bonne évolutivité du système. Après l'installation, il suffit de distribuer les fichiers de configuration vers les nouveaux nœuds. Cassandra met ici des outils adaptés à disposition des développeurs.






INTRODUCTION À CASSANDRA

Avantages et Inconvénients de Cassandra :

Avantages :

- **Évolutivité horizontale :** Capacité à augmenter les performances en ajoutant simplement des nœuds au cluster, offrant une solution économique pour les applications Big Data.
- **Modèle de données flexible :** Utilisation de tables de hachage multidimensionnelles permettant une modélisation variée des données, offrant une flexibilité accrue par rapport aux bases de données relationnelles.
- **Haute vitesse d'exécution :** Performances remarquables, surpassant souvent d'autres bases de données NoSQL dans les cas nécessitant des performances élevées.

Inconvénients






INTRODUCTION À CASSANDRA

Avantages et Inconvénients de Cassandra :

Avantages

Inconvénients :

- **Complexité de la modélisation des données :** La modélisation de données peut être complexe, ce qui rend la configuration et la maintenance du système plus compliquées, surtout pour les utilisateurs inexpérimentés.

- **Gestion d'un cluster de nœuds multiples :** La gestion et l'administration d'un cluster de nœuds multiples peuvent poser des défis, nécessitant une expertise supplémentaire et une attention particulière pour garantir la cohérence et la disponibilité des données.

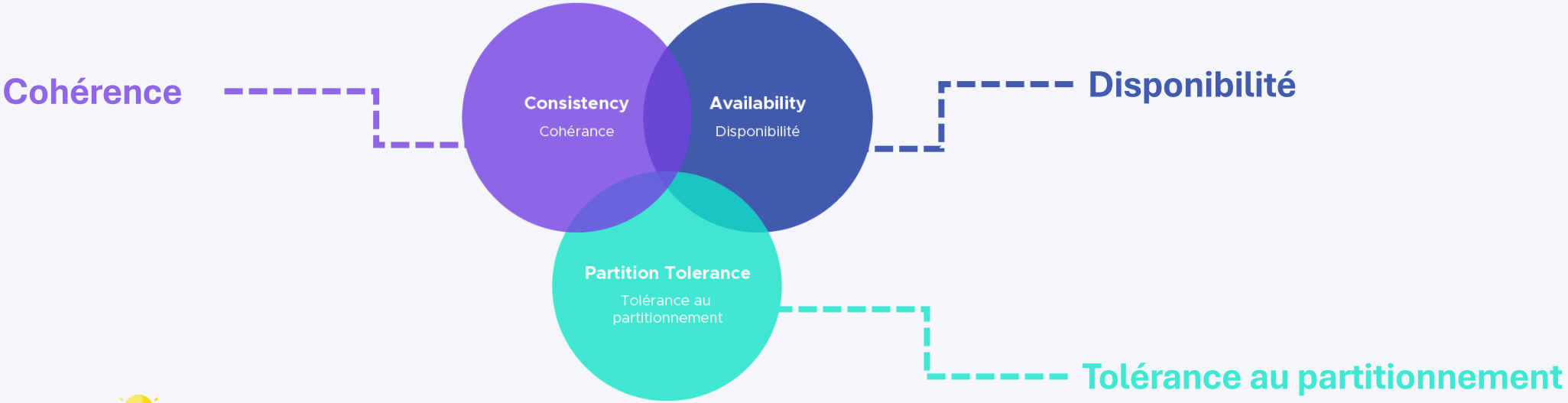





THÉORÈME CAP

- Le théorème CAP, également connu sous le nom de **théorème de Brewer**, est un concept fondamental en matière de conception de systèmes distribués, en particulier dans le domaine des bases de données distribuées. Il a été formulé par le chercheur en informatique **Eric Brewer** en 2000.

- Le théorème CAP stipule que dans tout système informatique distribué, vous ne pouvez garantir simultanément que deux des trois propriétés suivantes :





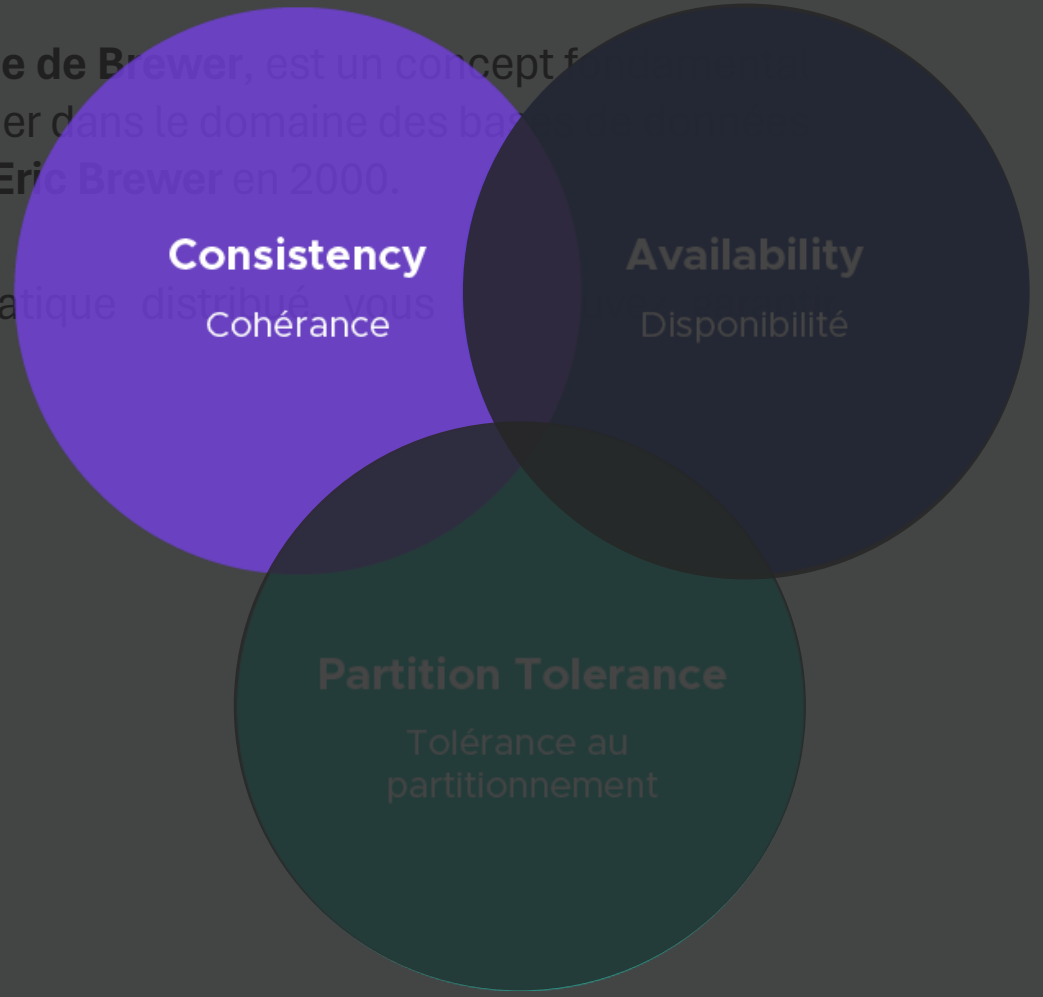



THÉORÈME CAP

- Le théorème CAP, également connu sous le nom de **théorème de Brewer**, est un concept f
en matière de conception de systèmes distribués, en particulier dans le domaine des ba
distribuées. Il a été formulé par le chercheur en informatique **Eric Brewer** en 2000.
- Le théorème CAP stipule que dans tout système informatique distribué, vous
ne pouvez garantir que deux des trois propriétés suivantes :

Cohérence

Tous les nœuds voient les mêmes données en même temps.
En d'autres termes, lorsque vous effectuez une opération de lecture sur un nœud, vous obtenez toujours le résultat le plus récent, indépendamment du nœud que vous interrogez.





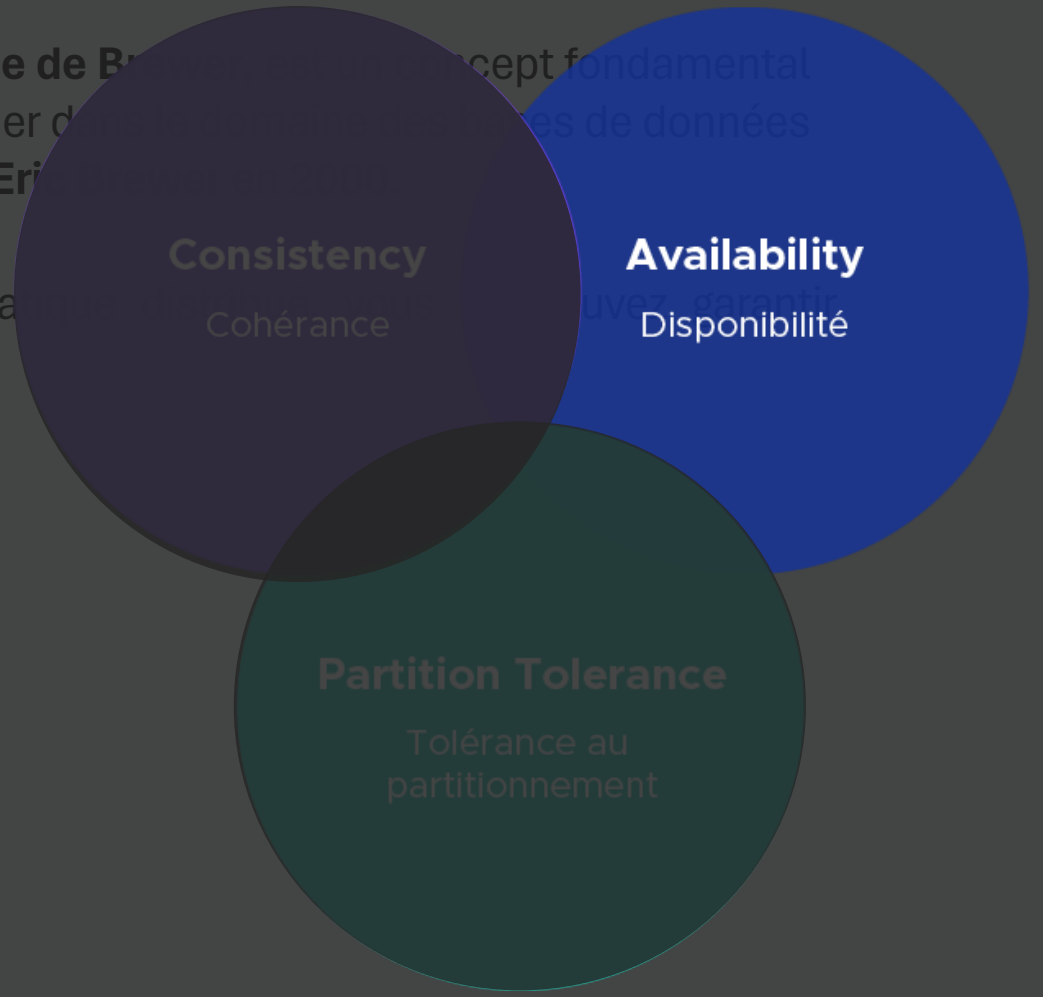



THÉORÈME CAP

- Le théorème CAP, également connu sous le nom de **théorème de B** cept fondamental en matière de conception de systèmes distribués, en particulier d es de données distribuées. Il a été formulé par le chercheur en informatique **Eri**
- Le théorème CAP stipule que dans tout système informa uvez garantir simultanément :

Disponibilité

Chaque requête reçoit une réponse (même si elle peut ne pas être la réponse la plus récente ou la plus précise). Cela signifie que le système reste opérationnel et répond aux requêtes, même en cas de panne d'un ou plusieurs nœuds.





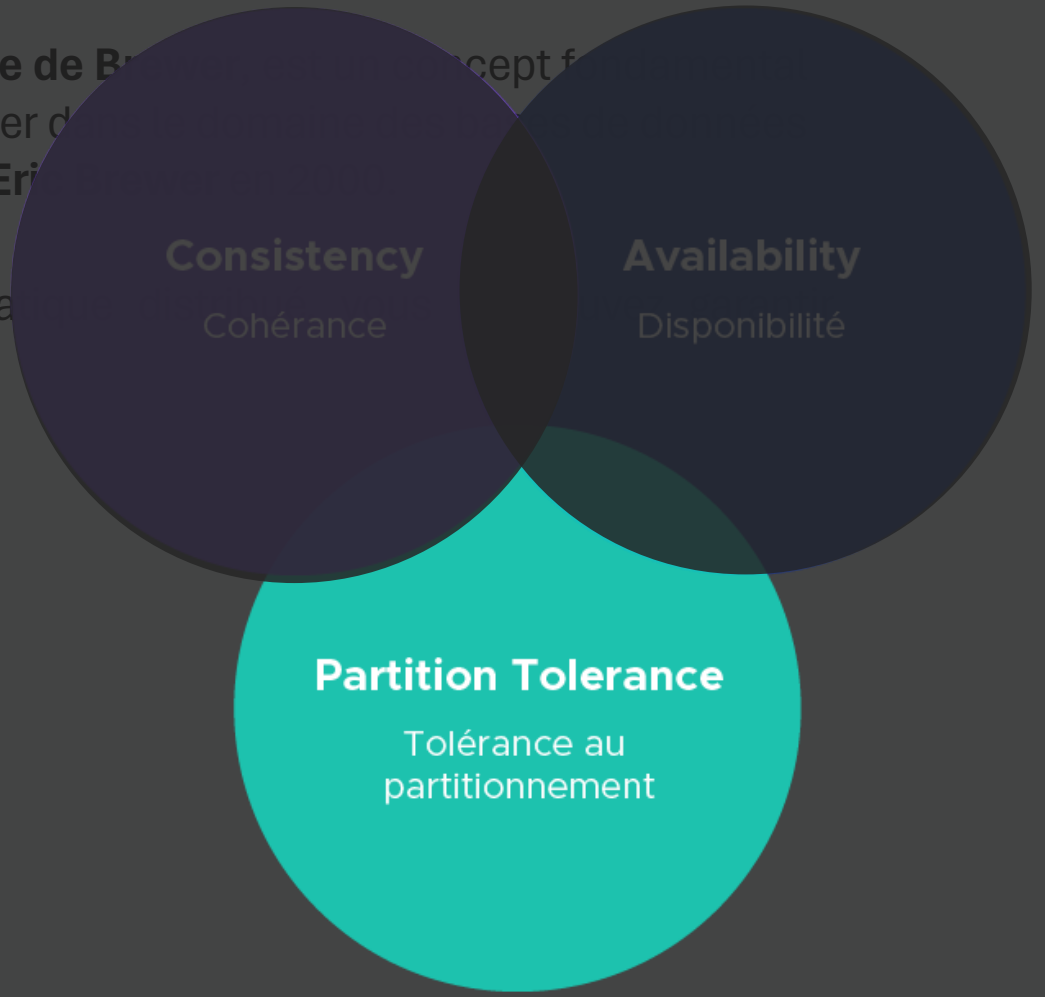



THÉORÈME CAP

- Le théorème CAP, également connu sous le nom de **théorème de B** cept f
  en matière de conception de systèmes distribués, en particulier d
  distribuées. Il a été formulé par le chercheur en informatique **Eri**
- Le théorème CAP stipule que dans tout système informa
  [illegible] que deux des trois propriétés suivantes

Tolérance au partitionnement

Le système continue de fonctionner malgré la perte de messages (ou de connexions) entre les nœuds. En d'autres termes, le système peut être partitionné en plusieurs sous-systèmes, et chaque sous-système continue de fonctionner de manière indépendante.





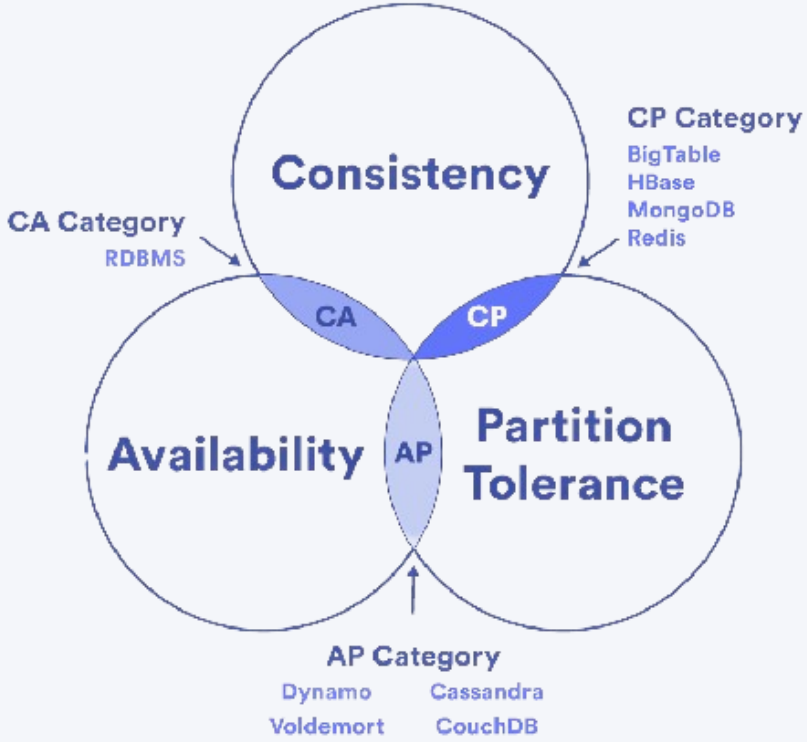



THÉORÈME CAP

Cassandra et le théorème CAP

- Par rapport au théorème CAP, Cassandra est une base de données **AP** ; elle assure la disponibilité et la tolérance au partitionnement, mais ne peut pas garantir la cohérence en permanence.

- Cassandra n'ayant pas de nœud principal, tous les nœuds doivent être disponibles en permanence. Cependant, Cassandra fournit une cohérence à terme en permettant aux clients d'écrire sur n'importe quel nœud à tout moment et en remédiant aux incohérences aussi rapidement que possible.








ARCHITECTURE DE CASSANDRA

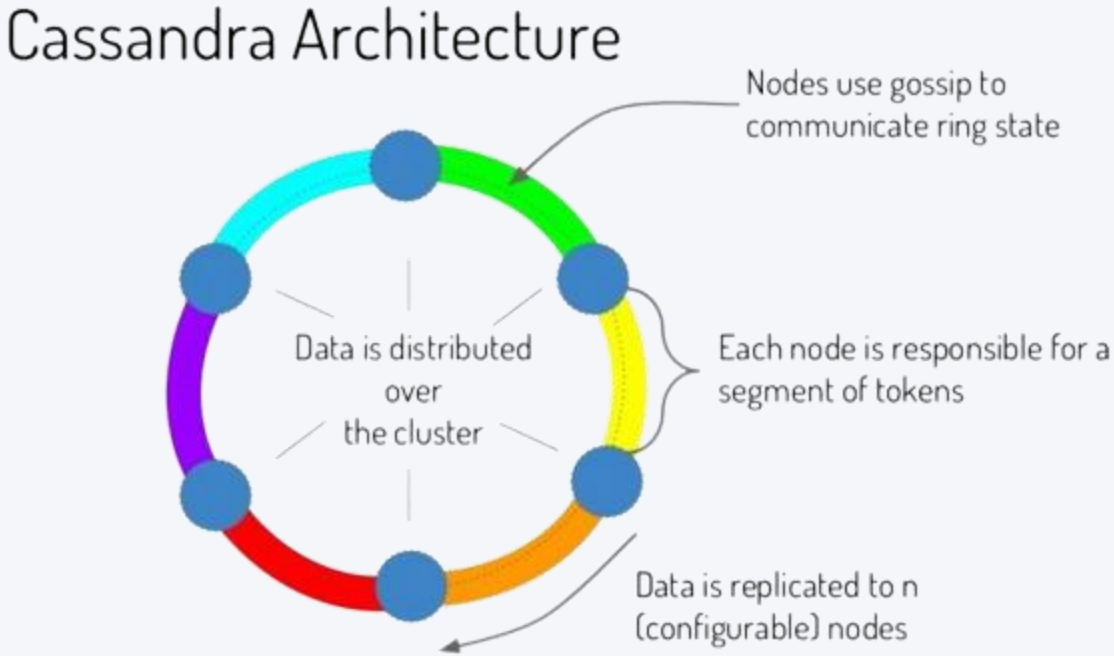
Définition Cluster

- Un cluster est un regroupement de plusieurs noeuds (serveur physique) qui communiquent entre eux pour la gestion de données.

ARCHITECTURE DE CASSANDRA

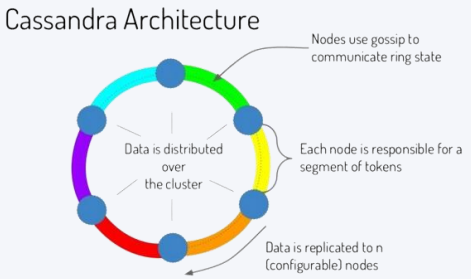
Fondamental :

- Cassandra est une base de donnée contenue dans un cluster.
- Comme de nombreuse base NoSQL, les données sont réparties sur plusieurs nœuds et peuvent être répliquée sur 1 à N nœuds.
- Un utilisateur peut se connecter sur n'importe quel nœud et accéder à l'ensemble des données.



ARCHITECTURE DE CASSANDRA

Fondamental :



Dans un cluster, tous les nœuds sont égaux, ce qui signifie qu'il n'y a pas de nœud maître ou un processus centralisant leur gestion.

En fait, Cassandra utilise un protocole appelé **Gossip** afin de découvrir la localisation et les informations sur l'état des autres nœuds du cluster.

Le protocole **Gossip** est un protocole de communication de type **« *peer-to-peer* »** dans lequel les nœuds échangent périodiquement des informations sur leur état mais également sur ce qu'ils savent des autres nœuds.

Pour être plus précis, le processus s'exécute toutes les secondes afin d'échanger les messages avec au plus trois autres nœuds du cluster. De plus, une version est associée à ces messages afin de permettre d'écraser les informations plus anciennes






MODÈLE DE DONNEES

MODÈLE DE DONNEES

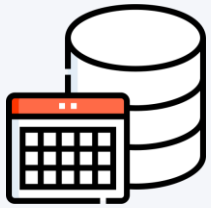
MODÈLE DE DONNEES

Base de données

Dans le domaine des **bases de données**, le modèle de données est essentiellement la façon dont les données sont organisées, structurées et stockées au sein du système de gestion de base de données.

MODÈLE DE DONNEES

Base de données

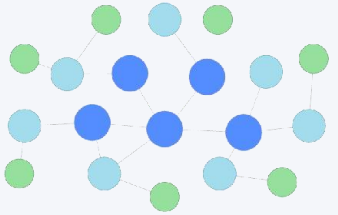


Modèle Relationnel

Row A	Column 1	Column 2	Column 3
	Value	Value	Value
Row B	Column 1	Column 2	Column 3
	Value	Value	Value

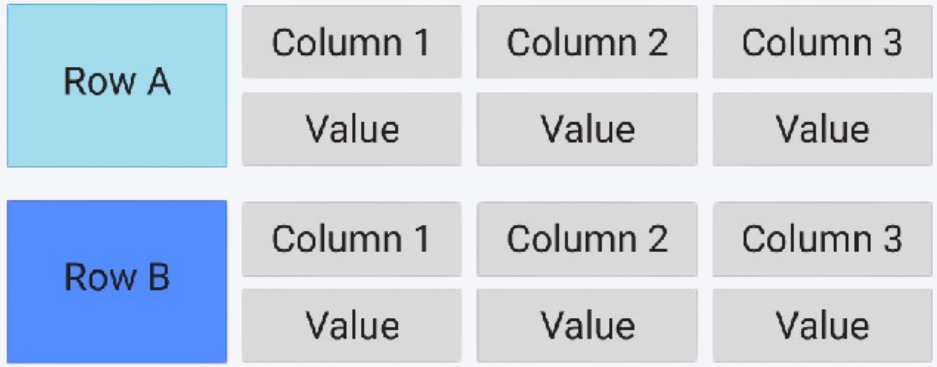
Modèle en Colonnes

Modèle en Documents



Modèle en Graphes

MODÈLE DE DONNEES



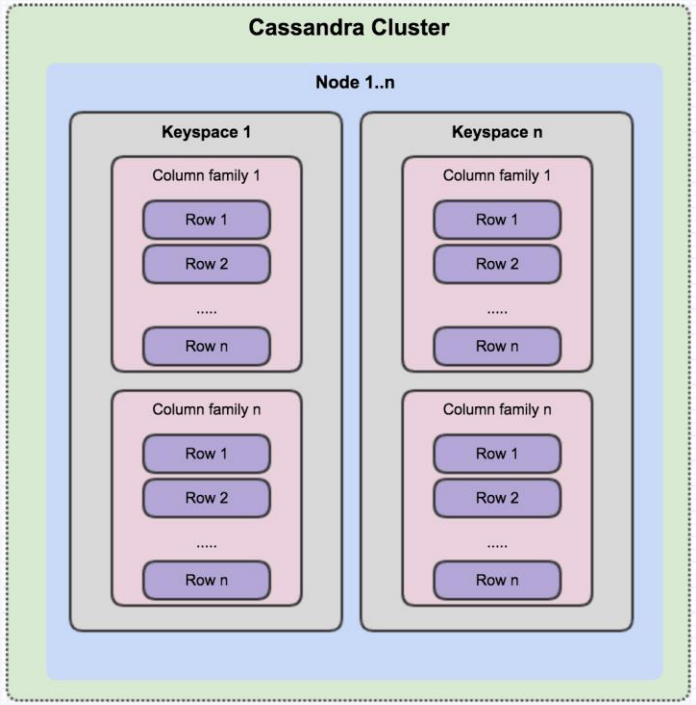
Modèle en Colonnes

- Dans ce modèle, les données sont organisées en colonnes plutôt qu'en lignes, offrant une flexibilité et des performances optimisées.

MODÈLE DE DONNEES

Dans le modèle de données de Cassandra, les données sont organisées en quatre niveaux principaux :

- Keyspace
- Column families
- Row
- Colonnes



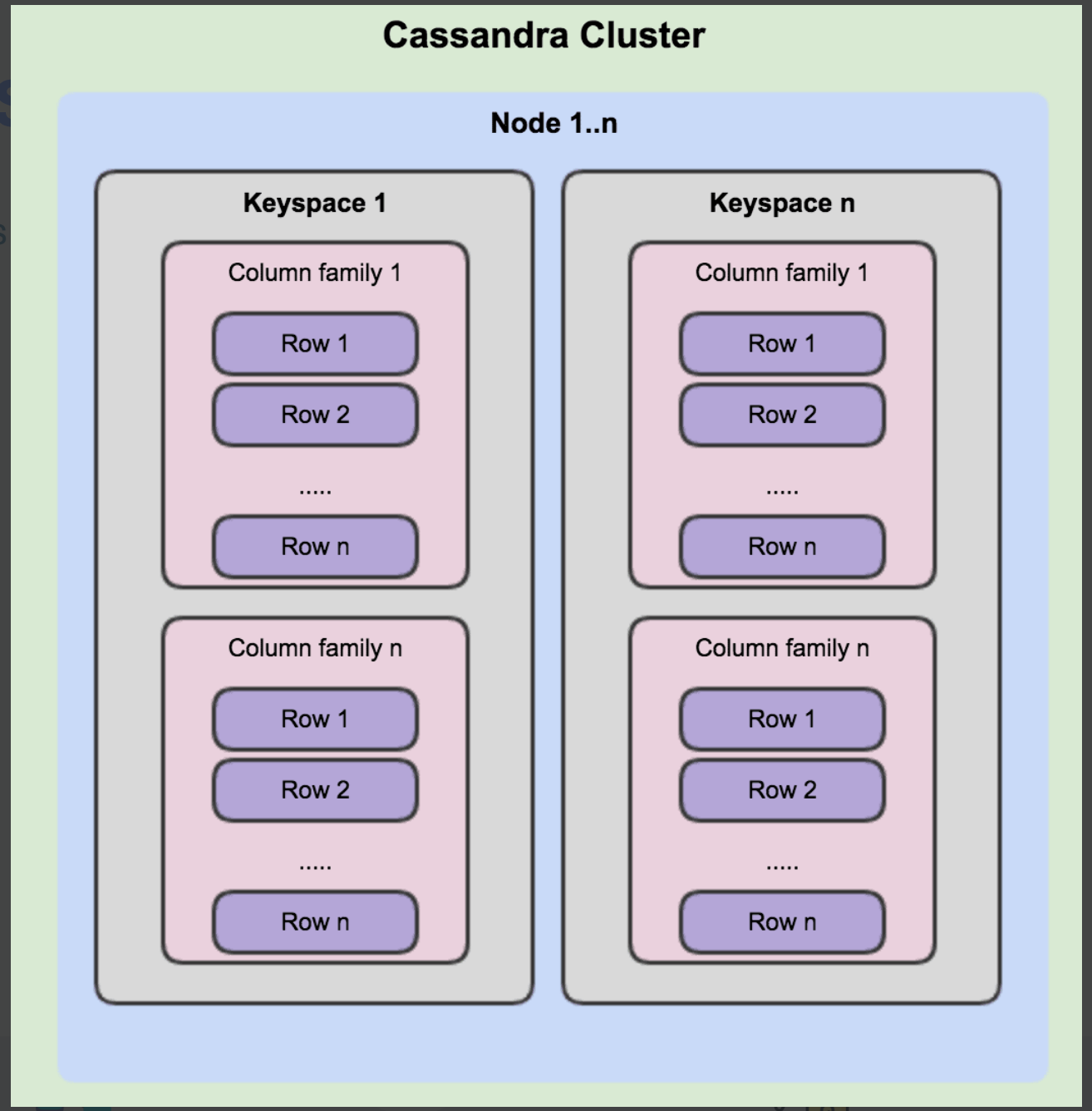





MODÈLE DE DONNEES

Dans le modèle de données
principaux :

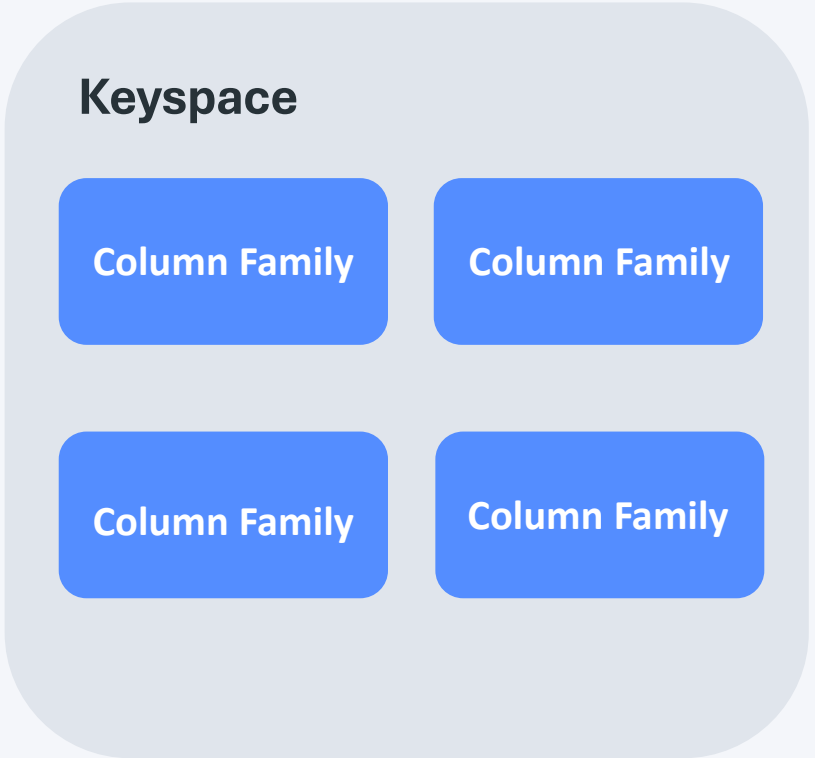
- **Keyspace**
- **Column families**
- **Row**
- **Colonnes**



MODÈLE DE DONNEES

Espace de noms :

- **Un keyspace** est un espace de noms logique qui regroupe des column families.
- C'est l'unité d'organisation la plus haute dans Cassandra.
- Il est comparable à une base de données dans un SGBD relationnel.



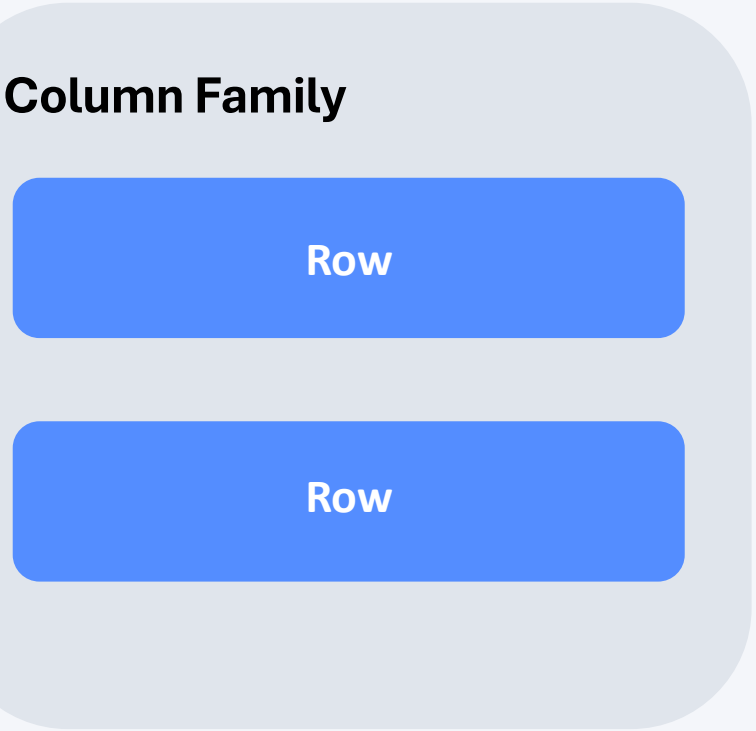





MODÈLE DE DONNEES

Famille de colonnes :

- **Une column family** est une structure de stockage de données dans Cassandra.
- Elle contient un ensemble de lignes, où chaque ligne est identifiée par une clé de ligne unique.
- Comparable à une table dans un modèle relationnel, bien que la structure soit différente.



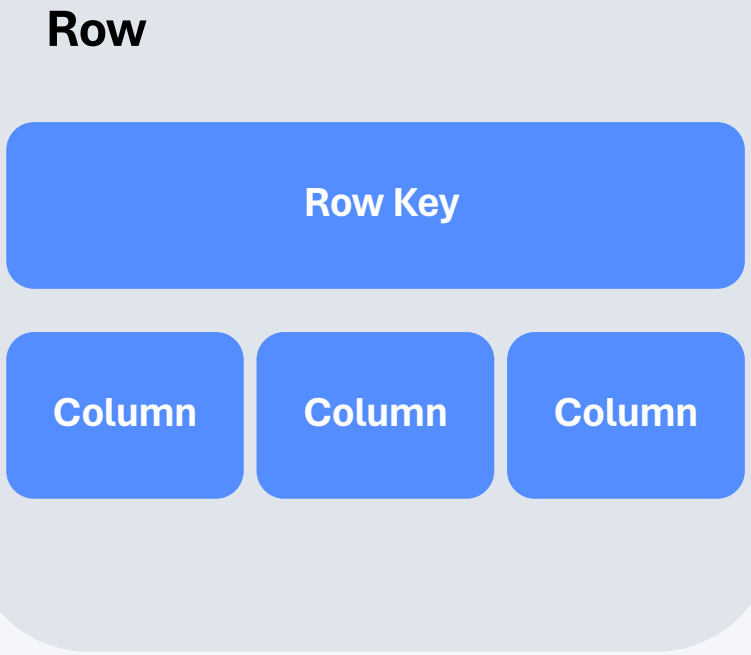





MODÈLE DE DONNEES

Ligne :

- **Une row** est une collection de valeurs pour une clé de partition particulière.
- C'est l'unité d'organisation des données dans une column family.
- Elle est comparable à un enregistrement dans un SGBD relationnel.
- Chaque ligne est identifiée par une clé de ligne unique (row key).



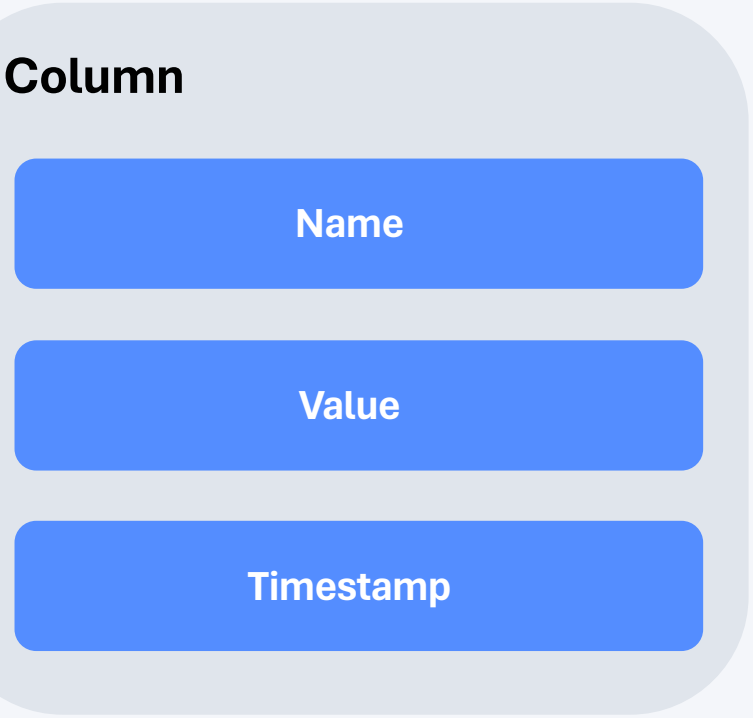


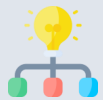



MODÈLE DE DONNEES

Colonne :

- **Une column** est une unité de données dans une row. C'est comparable à un champ dans un SGBD relationnel.
- Une colonne est l'unité de base pour stocker des données dans une row.
- Chaque colonne est composée de trois éléments :
  - **Nom :** Un identifiant unique pour la colonne au sein de la row.
  - **Valeur :** La donnée réelle stockée dans la colonne.
  - **Timestamp :** Une information temporelle indiquant quand la valeur a été insérée ou mise à jour.





MODÈLE DE DONNEES

- **Normalisation**

La normalisation est le processus de réduction de la redondance des données et de garantie de l'intégrité des données en les organisant dans plusieurs tables distinctes.

- **Avantages :**

  Simplicité des écritures, Intégrité des données

- **Inconvénients :**

  Complexité des requêtes, Lectures lentes

Employées

UserID	depID	Nom	Prenom
1	1	Codd	Edgar
2	1	Wild	Thomas

Départements

Departement_ID	departement
1	Informatique
1	Math






MODÈLE DE DONNEES

- **Dénormalisation**

La Dénormalisation est une stratégie de base de données pour améliorer les performances de lecture en ajoutant des copies redondantes de données, au détriment des performances d'écriture.

- **Avantages :**

  Lecture rapide, Requêtes simples

- **Inconvénients :**

  Complexité des écritures, Intégrité manuelle

Employées

UserID	Nom	Prenom	departement
1	Codd	Edgar	Informatique
2	Wild	Thomas	Math
3	Karlson	Mark	Ingénierie
4	Juniper	Jones	Informatique






MODÈLE DE DONNEES

- **La différence entre RDB et Cassandra**

Modèle Relationnel

- Tables avec lignes et colonnes
- Jointures pour relier les données entre tables
- Normalisation pour minimiser la redondance
- Minimiser la duplication

Modèle de Cassandra

- Familles de colonnes et colonnes
- Pas de jointures, les requêtes sont limitées à une table
- La Dénormalisation
- Acceptée pour améliorer les performances






CQL (CASSANDRA QUERY LANGUAGE)

- CQL, ou Cassandra Query Language, est le langage de requête utilisé pour communiquer avec les bases de données Cassandra.
- CQL ressemble beaucoup au SQL, mais il a ses propres spécificités pour fonctionner de manière optimale avec la structure distribuée de Cassandra.
- Types des requêtes

Requêtes DDL
Data Definition Language

Requêtes DML
Data Manipulation Language






CQL (CASSANDRA QUERY LANGUAGE)

- **Requêtes DDL :**

Les requêtes DDL sont utilisées pour définir la structure de la base de données, comme **la création, la modification et la suppression** des éléments de base de données tels que **les cléspace et les tables**.

Exemples de requêtes DDL

Création de cléspace :

```
CREATE KEYSPACE my_keyspace WITH replication = { 'class' : 'SimpleStrategy', 'replication_factor': 1 };
```

- **my_keyspace :** Nom du keyspace.
- **Replication :** Définit la stratégie de copie des données.
- **SimpleStrategy :** Copie les données sur un nombre fixe de nœuds.
- **replication_factor :** 1: Une seule copie des données sera créée.






CQL (CASSANDRA QUERY LANGUAGE)

- **Requêtes DDL :**

Les requêtes DDL sont utilisées pour définir la structure de la base de données, comme **la création, la modification et la suppression** des éléments de base de données tels que **les cléspace et les tables.**

Exemples de requêtes DDL

Modification de cléspace :

```
ALTER KEYSPACE my_keyspace WITH replication = { 'class' : 'SimpleStrategy', 'replication_factor' : 3 };
```






CQL (CASSANDRA QUERY LANGUAGE)

- **Requêtes DDL :**

Les requêtes DDL sont utilisées pour définir la structure de la base de données, comme **la création, la modification et la suppression** des éléments de base de données tels que **les cléspace et les tables.**

Exemples de requêtes DDL

Suppression de cléspace :

```
DROP KEYSPACE my_keyspace ;
```






CQL (CASSANDRA QUERY LANGUAGE)

- **Requêtes DML :**

Les requêtes DML sont utilisées pour manipuler les données stockées dans la base de données, y compris **l'insertion, la mise à jour et la suppression** des données.

Exemples de requêtes DML

Insertion de données :

```
INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...) ;
```






CQL (CASSANDRA QUERY LANGUAGE)

- **Requêtes DML :**

Les requêtes DML sont utilisées pour manipuler les données stockées dans la base de données, y compris **l'insertion, la mise à jour et la suppression** des données.

Exemples de requêtes DML

Mise à jour de données :

```
UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;
```






CQL (CASSANDRA QUERY LANGUAGE)

- **Requêtes DML :**

Les requêtes DML sont utilisées pour manipuler les données stockées dans la base de données, y compris **l'insertion, la mise à jour et la suppression** des données.

Exemples de requêtes DML

Suppression de données :

```
DELETE FROM table_name WHERE condition;
```






FONCTIONNALITÉS CLÉS DE CASSANDRA

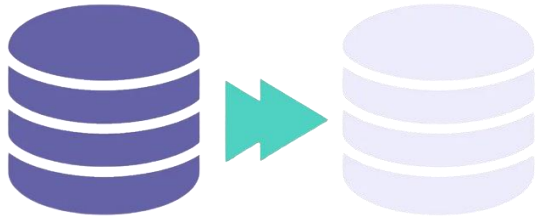
Réplication

Extensibilité linéaire

Cohérence configurable

Performance

Grâce à ses multiples copies des données **réparties sur plusieurs nœuds**, Cassandra permet aux clusters de continuer de fonctionner même en cas de perte d'un ou plusieurs nœuds. Cela permet d'effectuer des maintenances et des mises à jour sur la base de données sans interruption de service.



FONCTIONNALITÉS CLÉS DE CASSANDRA

Réplication

Extensibilité linéaire

Cohérence configurable

Performance

Pour pallier la montée en charge des serveurs, Cassandra bénéficie d'une **extensibilité linéaire**, en ajoutant de nouveau serveur, la puissance disponible augmentera de manière linéaire sans limite.

FONCTIONNALITÉS CLÉS DE CASSANDRA

Réplication

Extensibilité linéaire

Cohérence configurable

Performance

La cohérence des données est configurable, pour chaque requête en écriture ou en lecture, il est possible de choisir le niveau de cohérence des données utilisée. Le niveau de cohérence souhaité influe sur la performance des requêtes.

FONCTIONNALITÉS CLÉS DE CASSANDRA

Réplication

Extensibilité linéaire

Cohérence configurable

Performance

- En plus de fournir une cohérence configurable, Cassandra possède une structure de données qui permet de mettre les données ayant une relation entre elles le plus proche possible les unes des autres. Cela permet de stockées les données qui nous intéresse dans le même nœud et dans l'ordre dans lequel on souhaite les rechercher.
- Étant donné que Cassandra est une base de données distribuée, les performances sont améliorées en regroupant les données en partition au sein de nœuds.
- Attention, il ne faut pas mettre un système de « load balancer » qui enverrait les requêtes à Cassandra, un tel système existe déjà au sein de Cassandra et cela provoquerait des interférences






THANKS FOR YOUR ATTENTION

Created by MOKNIA Youssef

Contributors

- Youssef EL WALI
- Halima BANANI
- Asmaa AIT EL HAJ
- Yassine AMZIR