ENSAO | UMP

THÉORÈME CAP

Présenter par Ibrahim Lahlou
Encadré par le Prof Kamal Azghiou
Pour le Module NoSQL et Introduction au Big Data

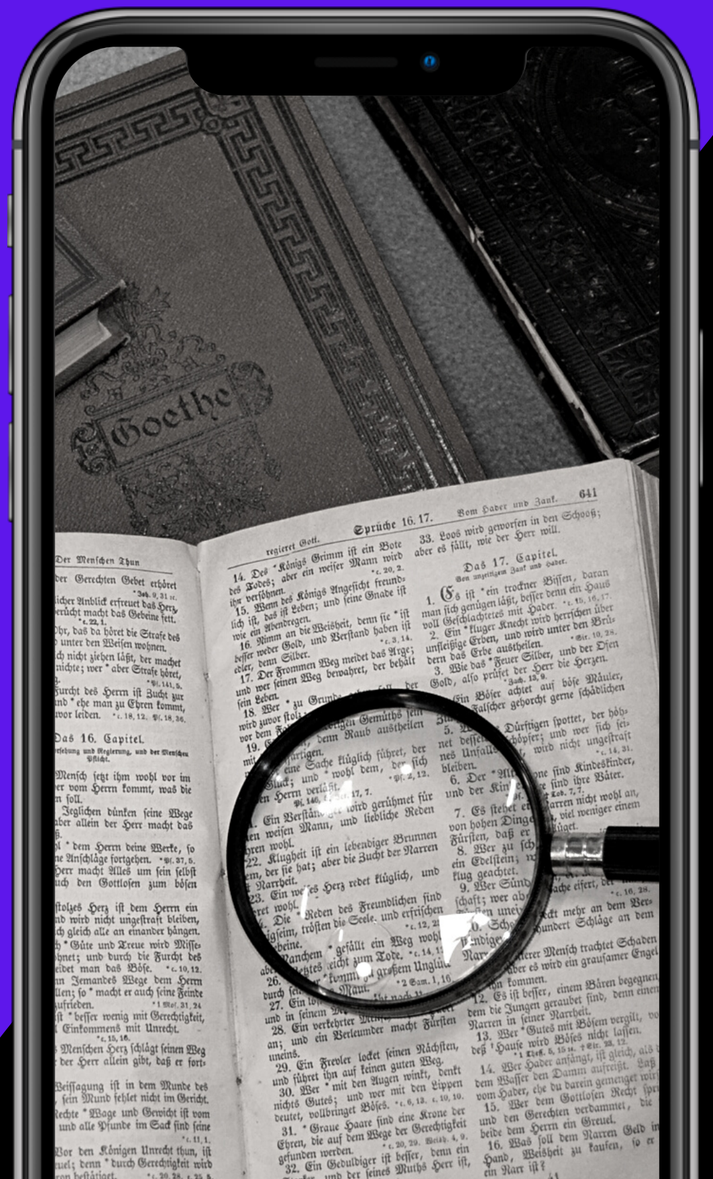
SOMMAIRE

Ce que cette présentation va couvrir

- Introduction (3)
- Concepts du Théorème CAP
  - C'est quoi l'acronyme C.A.P (3)
  - Un peu d'histoire (2)
  - l'énoncé du théorème (1)
  - Démonstration intuitive (4) +Quiz
- Quel Critère a prendre en compte pour choisir sa base de donné ? (4)
- une Extension du Théorème CAP(2)
- Bibliographie

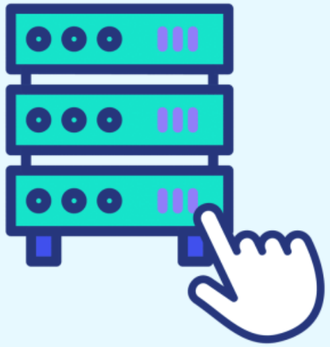
INTODUCTION

Vision Global

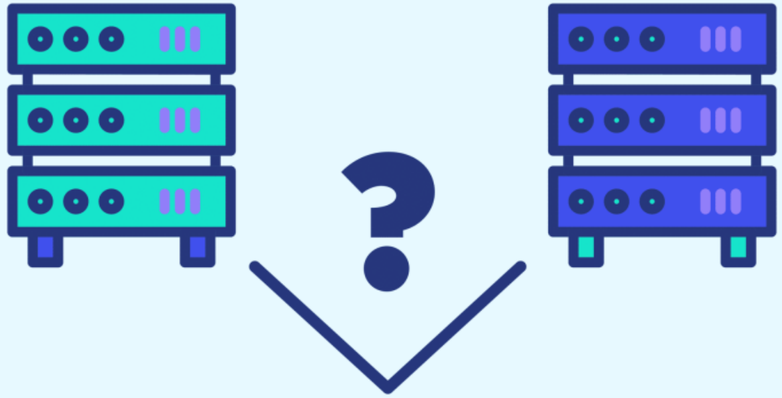


INTODUCTION

C'est quoi le théorèm CAP ?



Comment Choisir sa BDD ?

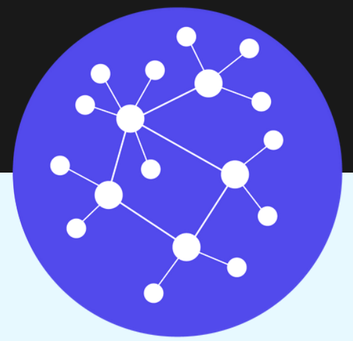


Quel Système Choisir ?

VOCABULAIRE

Systeme Centralisé

un système dans lequel un contrôleur central exerce un contrôle sur les composants de niveau inférieur du système directement , le contrôleur central doit être infaillible

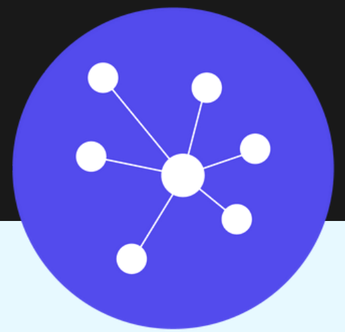
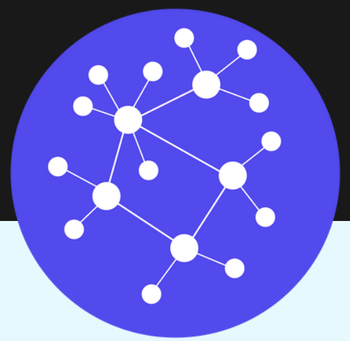


Système Décentralisé

Un système avec peu de coordinateurs centraux coopérant ensemble, donc même si un coordinateur particulier échoue dans cette architecture particulière, le service sera toujours accessible par un autre coordinateur.

Système Distribué

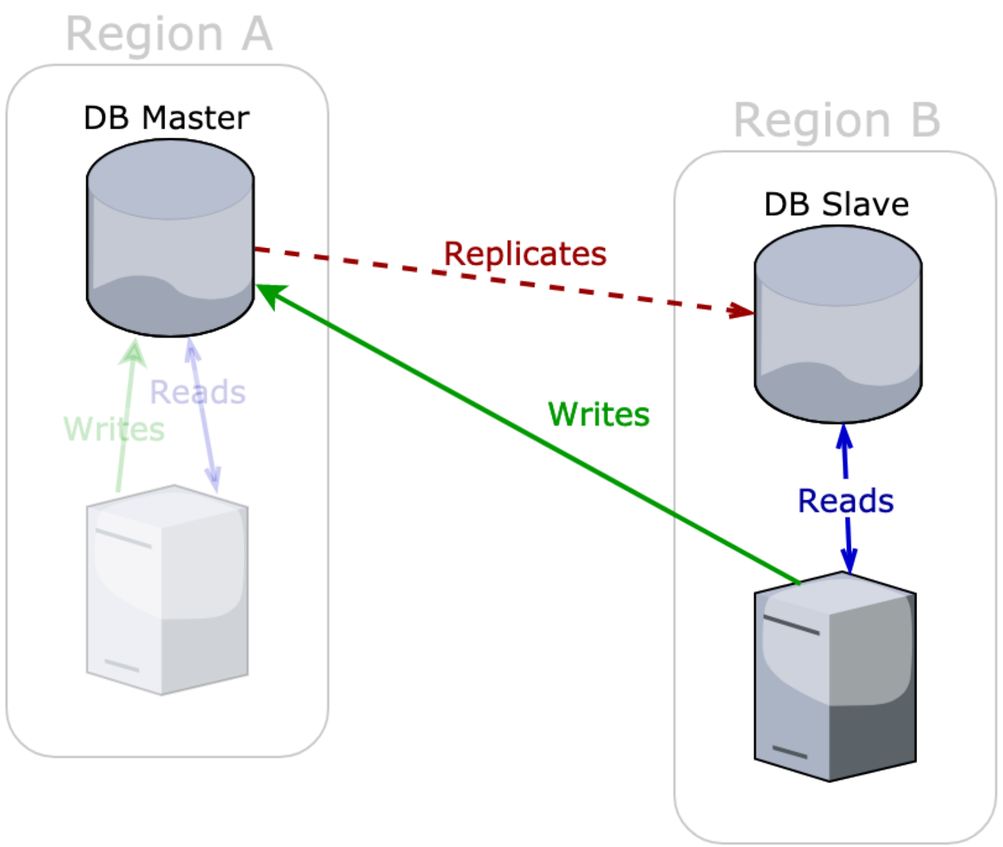
Un systèmes ou chaque points exécute collectivement le travail sans avoir un coordinateur central , il se coordonnent les uns avec les autres

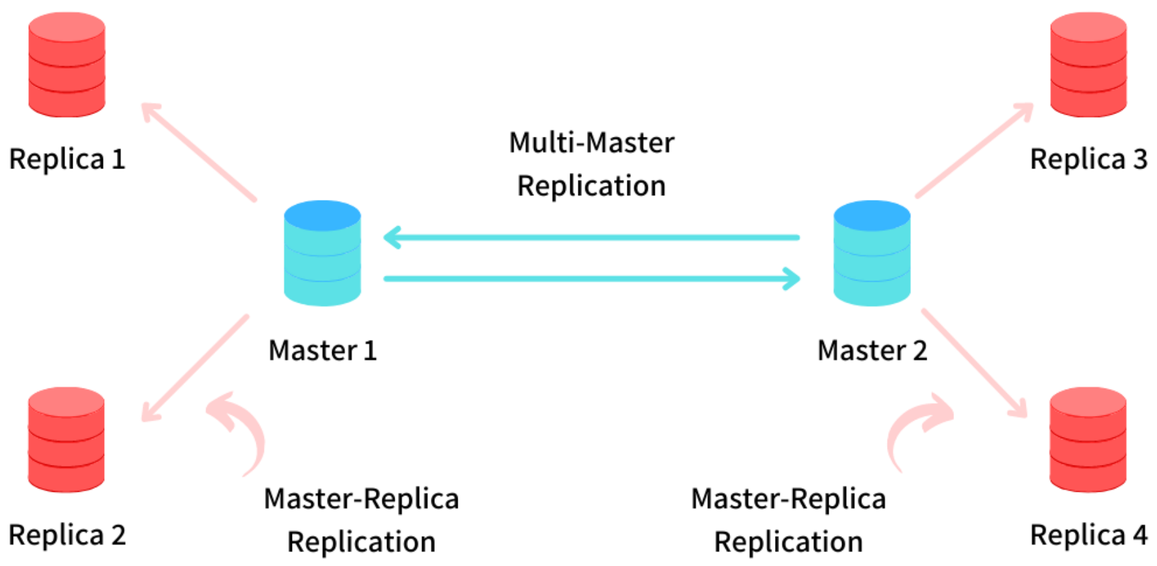



	Systeme Centralisé	Système Décentralisé	Système Distribué
Entretien/Points de défaillance	faciles à entretenir car il n'y a qu'un seul point de défaillance.	plus de point de défaillance mais toujours un nombre fini d'échecs.	ils sont les plus difficiles à entretenir
Tolérance/stabilité aux pannes	ils peuvent être très instables et intolérables en raison d'une défaillance ponctuelle pouvant ruiner l'ensemble du système de travail.	ils sont des systèmes stables par rapport aux systèmes centralisés, comme si l'échec du leader dans le système décentralisé ne nuisait pas au reste du système. Néanmoins, nous aurons un réseau stable fonctionnant en synchronisation.	ils sont beaucoup plus stables et un seul point de défaillance ne fait pas beaucoup de mal.
Facilité de développement/création	Des systèmes centralisés peuvent être créés rapidement, et un moyen facile doit construire un serveur central pour commander/gérer les composants connectés.	Pour les systèmes décentralisés et distribués, il faut d'abord travailler sur les détails de niveau inférieur comme le partage des ressources IT (matériel / logiciel), les transactions et le réseau, et cela impose un certain niveau de difficulté à maintenir ces systèmes à leur état cohérent.	

VOCABULAIRE

- Un aperçu sur l'architecture(Réplication) Maitre/Esclave et multi maitre

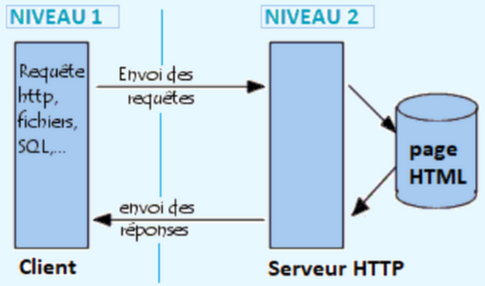




Multi-Master Replication

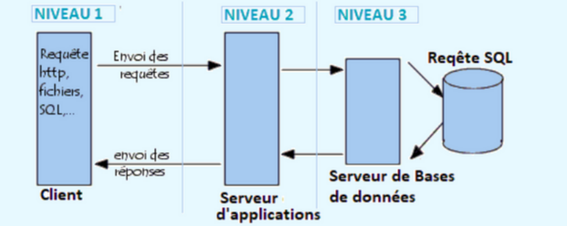
VOCABULAIRE

- Type de Système Distribué



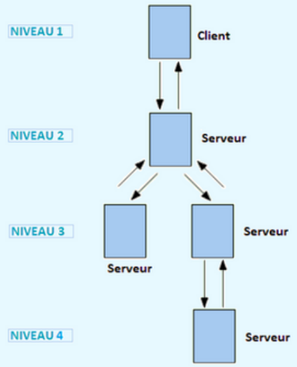
Client-serveur

Les clients contactent le serveur pour obtenir des données, puis les formatent et les affichent à l'utilisateur final. L'utilisateur final peut également effectuer une modification depuis le côté client et la renvoyer au serveur pour la rendre permanente.



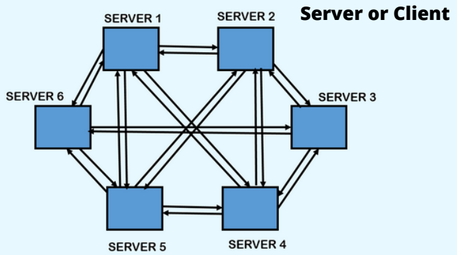
Trois niveaux

Les informations relatives au client sont stockées dans un niveau intermédiaire plutôt que sur le client afin de simplifier le déploiement de l'application. Ce modèle d'architecture est le plus courant pour les applications Web.



N niveaux

Généralement utilisé lorsqu'une application ou un serveur doit transmettre des demandes à des services d'entreprise supplémentaires sur le réseau.



Peer-to-peer

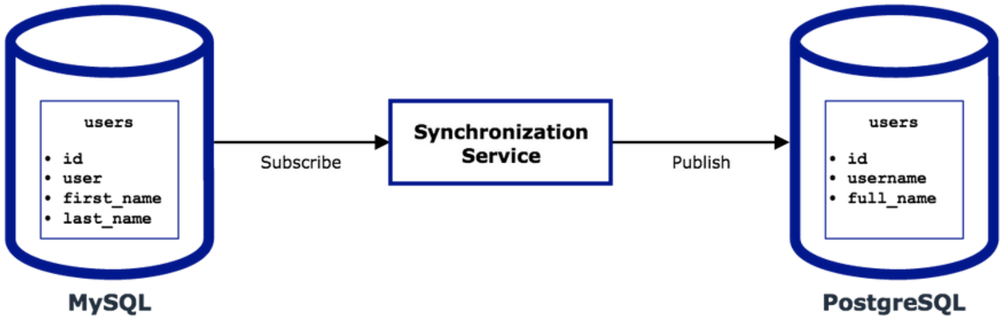
Il n'y a pas de machines supplémentaires utilisées pour fournir des services ou gérer des ressources. Les responsabilités sont uniformément réparties entre les machines du système, appelées pairs, qui peuvent servir de client ou de serveur.

VOCABULAIRE

- Principe de Pub/Sub

La synchronisation de bases de données est un concept ancien, utilisé depuis des décennies via la réplication avec des mécanismes de bas niveau comme le journal binaire. Une approche similaire est le Pub/Sub, où une base de données publie les changements et une autre s'y abonne.

L'utilisation de SQL Pub/Sub de Watermill permet d'implémenter ces modèles, mais présente actuellement des limitations. Seules les instructions INSERT déclenchent l'envoi de nouveaux messages à l'abonné. Toutefois, un support prévu pour les abonnés MySQL binlog et PostgreSQL LISTEN/NOTIFY permettra d'écouter également les mises à jour et les suppressions.



CONCEPT DU THÉORÈM CAP

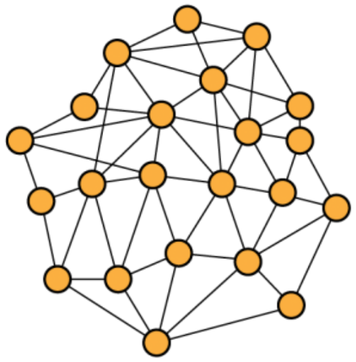
Théorem CAP alias Théorem de Brewer

C'EST QUOI CAP

Concistancy

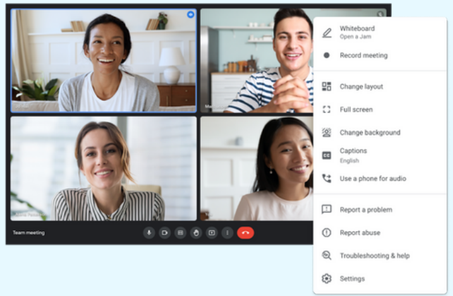
La cohérence signifie que tous les clients voient les mêmes données en même temps, quel que soit le nœud auquel ils se connectent.

Pour chaque instant tout les noeuds du réseau on la même valeurs



= Value: X @ 2018-05-03T08:52:40

Avec plusieurs copies des mêmes données, nous sommes confrontés à des options sur la façon de les synchroniser afin que les clients aient une vue cohérente des données.

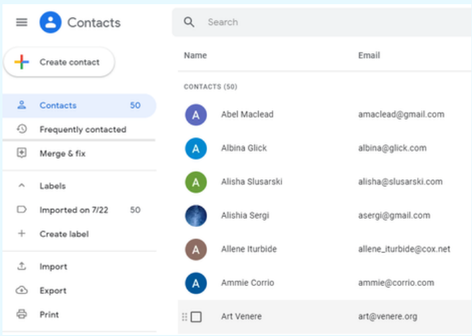


1. Faible cohérence

Après une écriture, les lectures peuvent ou non le voir. Une approche de meilleur effort est adoptée.
Cette approche est observée dans des systèmes tels que les caches.

Une faible cohérence fonctionne bien dans les cas d'utilisation en temps réel

Ex. la VoIP, le chat vidéo et les jeux multijoueurs en temps réel.

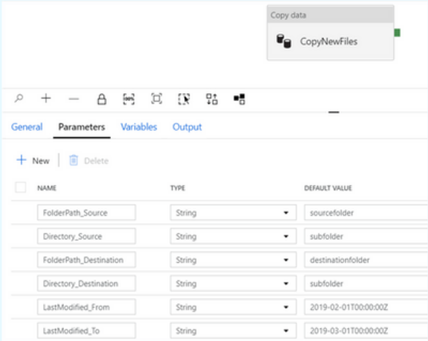


2. Cohérence éventuelle

Après une écriture, les lectures finiront par le voir (généralement en quelques millisecondes). Les données sont répliquées de manière asynchrone.

La cohérence éventuelle fonctionne bien dans les systèmes hautement disponibles.

Ex. Système de messagerie, indexation des moteurs de recherche, DNS.



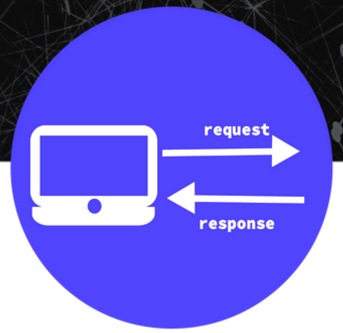
3. Une forte cohérence

Après une écriture, les lectures le verront. Les données sont répliquées de manière synchrone.

Une forte cohérence fonctionne bien dans les systèmes qui nécessitent des transactions.

Ex. Système de fichiers, SGBDR

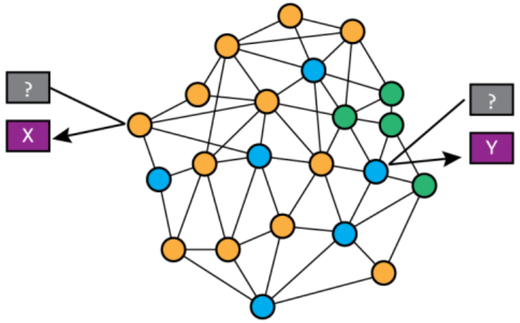
C'EST QUOI CAP



Availability

La disponibilité signifie que tout client qui fait une requête obtient une réponse, même si un ou plusieurs nœuds sont en panne.

Chaque **Requête** partager au réseau reçoit une **Réponse**, sans avoir une garantie que les données retournées sont les plus récentes



= Value: X @ 2018-05-03T08:52:40

= Value: Z @ 2018-05-03T08:32:58

= Value: Y @ 2018-05-03T07:12:12

Un système de haute disponibilité fonctionne en :

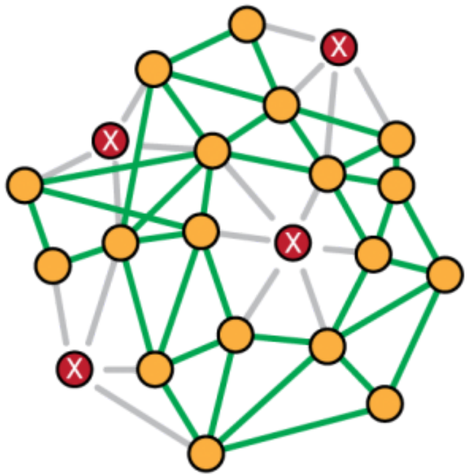
- Disposer de plus d’actifs matériels et logiciels que nécessaire afin d’avoir des sauvegardes capables de prendre le relais si quelque chose arrive au composant principal.
- Effectuer des contrôles réguliers pour s’assurer que chaque composant (primaire et celui en réserve) fonctionne correctement.
- Création d’un mécanisme de basculement automatique qui bascule les opérations vers un composant de sauvegarde en cas de panne du système principal.

C'EST QUOI CAP

Partition Tolerance

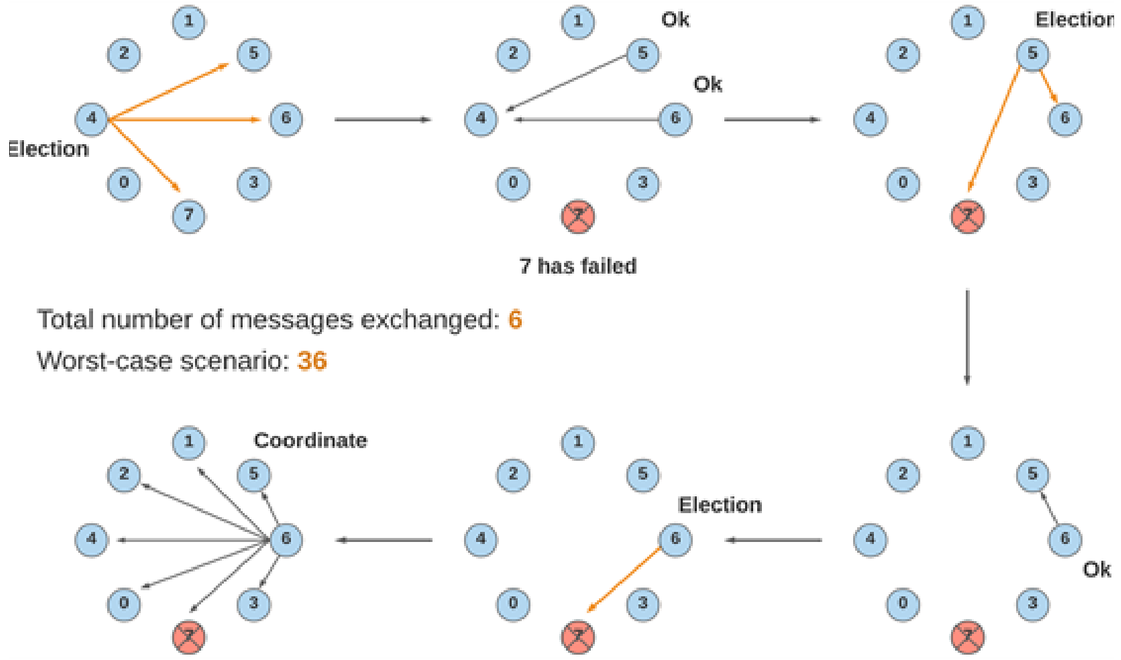
La Résistance à la distribution. Même s'il n'y a pas de communication entre les nœuds, ils continuent à fonctionner indépendamment les uns des autres.

Le réseau continue d'opérer , même si un bon nombre de noeuds ont une panne ou un **échec** de connexion



- **Une partition de réseau** est une situation dans laquelle un réseau est divisé en deux ou plusieurs parties distinctes qui ne peuvent pas communiquer entre elles.
- Cela peut se produire en raison de divers facteurs, tels que des pannes de réseau, des défaillances matérielles ou des bogues logiciels.
- Dans un système distribué, une partition de réseau peut entraîner des incohérences et des conflits dans les données, et rendre difficile, voire impossible, le bon fonctionnement du système.
- Pour atténuer l'impact des partitions du réseau, les systèmes distribués peuvent être conçus pour une haute tolérance aux partitions ou peuvent utiliser des techniques telles que *l'élection d'un leader* ou des *algorithmes de consensus*.

- **L'élection du leader** est un processus dans lequel un nœud leader est choisi parmi un groupe de nœuds dans un système distribué. Le leader est chargé de coordonner les actions des autres nœuds et de prendre des décisions au nom du groupe.
- **Les algorithmes d'élection du leader** peuvent être mis en œuvre à l'aide d'un coordinateur centralisé ou d'un algorithme distribué. Ils constituent une technique importante pour garantir la fiabilité et les performances des systèmes distribués.



UN PEU D'HISTOIRE

- Le théorème **CAP** a été énoncé en 1999 par **Fox** et **Brewer**
- Il a été démontré mathématiquement par **Gilbert** et **Lynch** en 2002
- Le théorème **CAP** est souvent utilisé pour justifier les plates-formes de données NoSQL qui ne maintenez pas la cohérence des données, mais fournissez plutôt une cohérence éventuelle ou d'autres modèles de cohérence faible pour les données répliquées .

UN PEU D'HISTOIRE

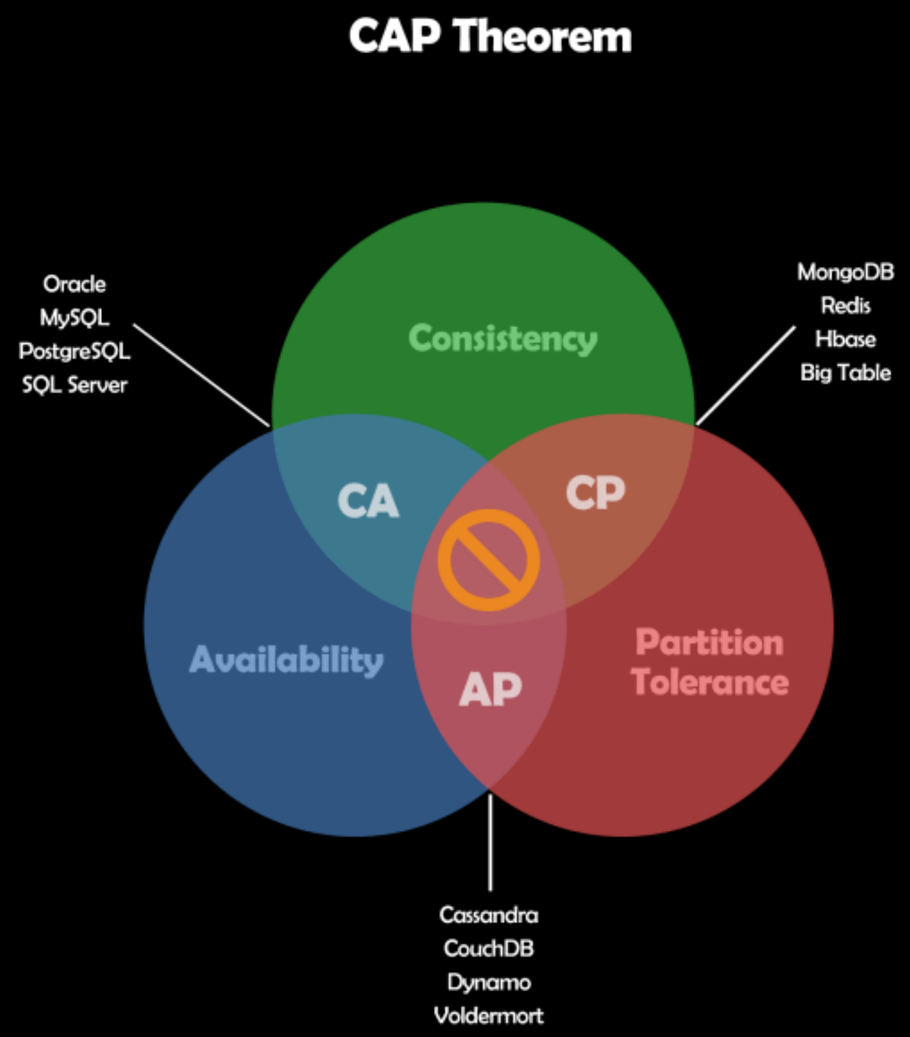
Eric Brewer

Scientifique , Professeur et Vice Président de l'infrastructure de la société Google .

Inktomi YAHOO! Google

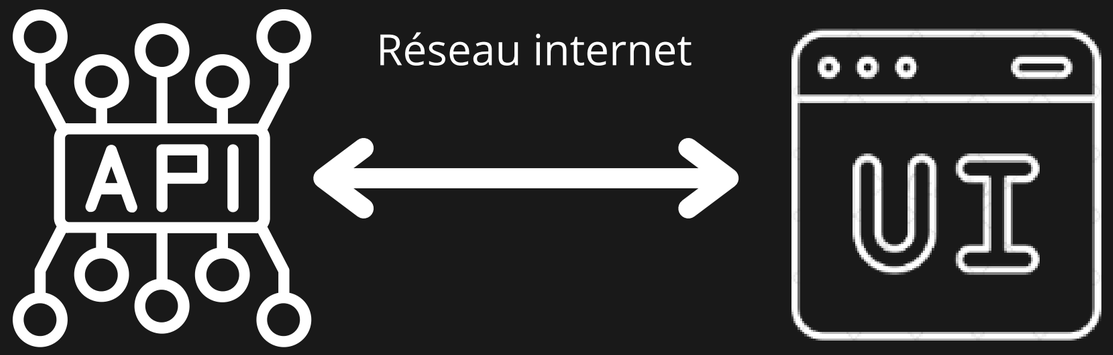
2000 |Conférence sur Services réseau évolutifs basés sur des clusters basé sur les observation de running Inktomi et il a formalisé un théorème très intéressant reposant sur 3 propriétés fondamentales pour caractériser les bases de données

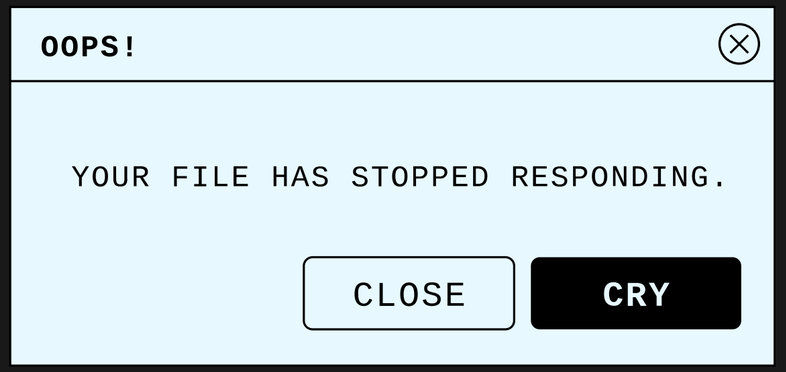
ENONCÉ DU THÉORÈME



"DANS TOUTE BASE DE DONNÉES, VOUS NE POUVEZ RESPECTER AU PLUS QUE 2 PROPRIÉTÉS PARMI LA COHÉRENCE, LA DISPONIBILITÉ ET LA DISTRIBUTION.."

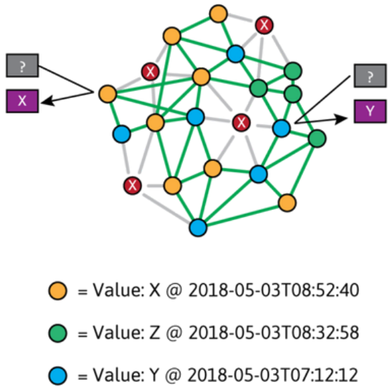
Exemple 1 : Site Web Frontend



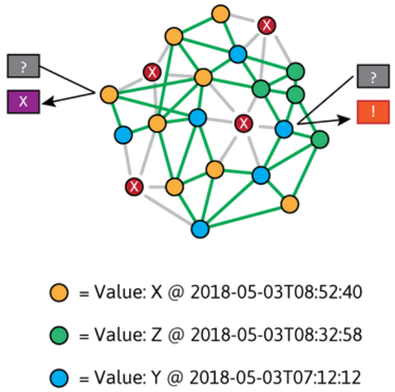


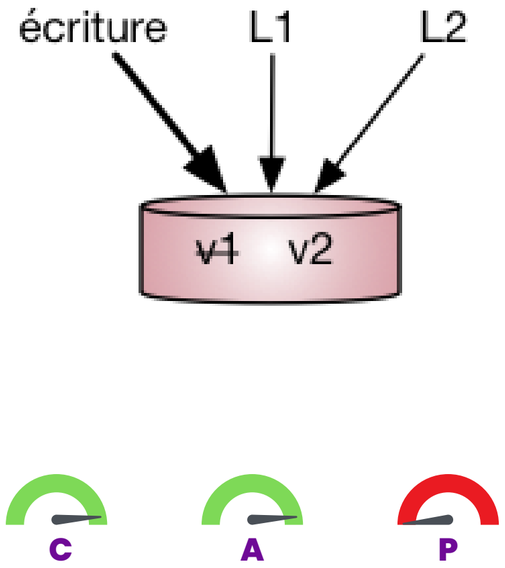
Exemple 2 : Architecture d'un système distribué

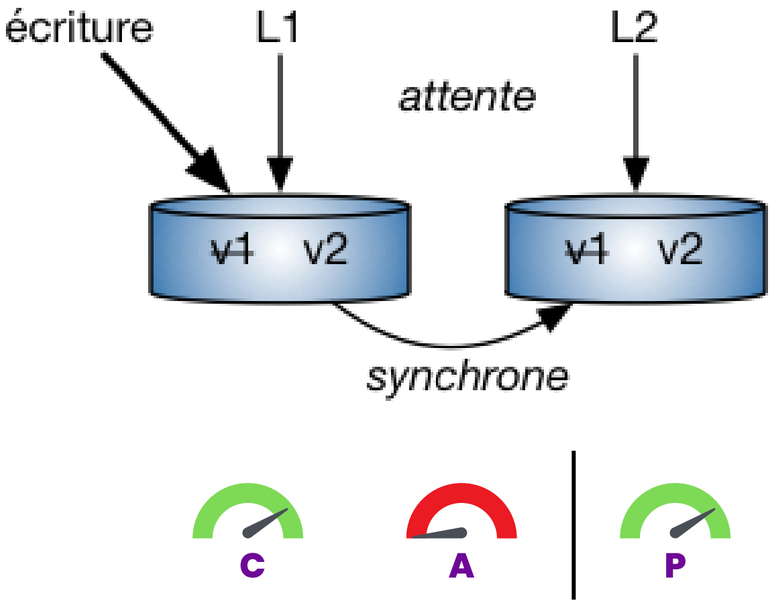
Chaque Requête adressée au réseau reçoit une **Réponse**, même si le réseau ne peut pas garantir qu'elle est à jour en raison du partitionnement du réseau (nœuds **défaillants**).

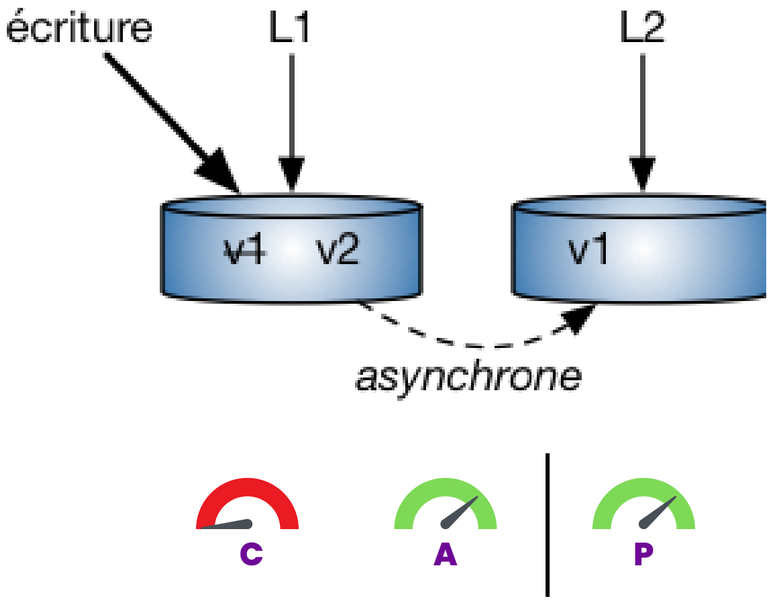


Le système renverra **une erreur** ou un délai d'attente si des informations particulières ne peuvent pas être garanties comme étant à jour en raison du partitionnement du réseau (nœuds **défaillants**).









NEWS ALERT

QUIZ

NEWS ALERT

Question 1 : Comment pouvons-nous améliorer l'évolutivité en utilisant des partitions de données ?

Question 2 : Pensez-vous qu'il est préférable d'avoir un système qui a une cohérence éventuelle ou faible ? Expliquer pourquoi?

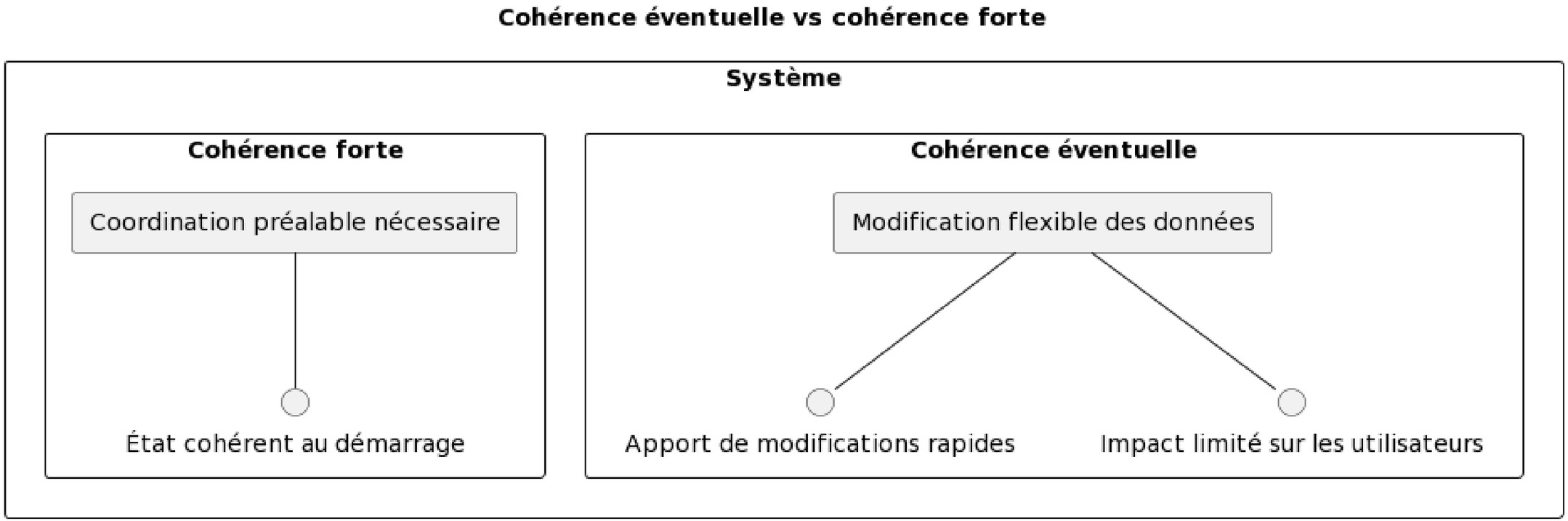
ANSWER

Réponse Question 1 :

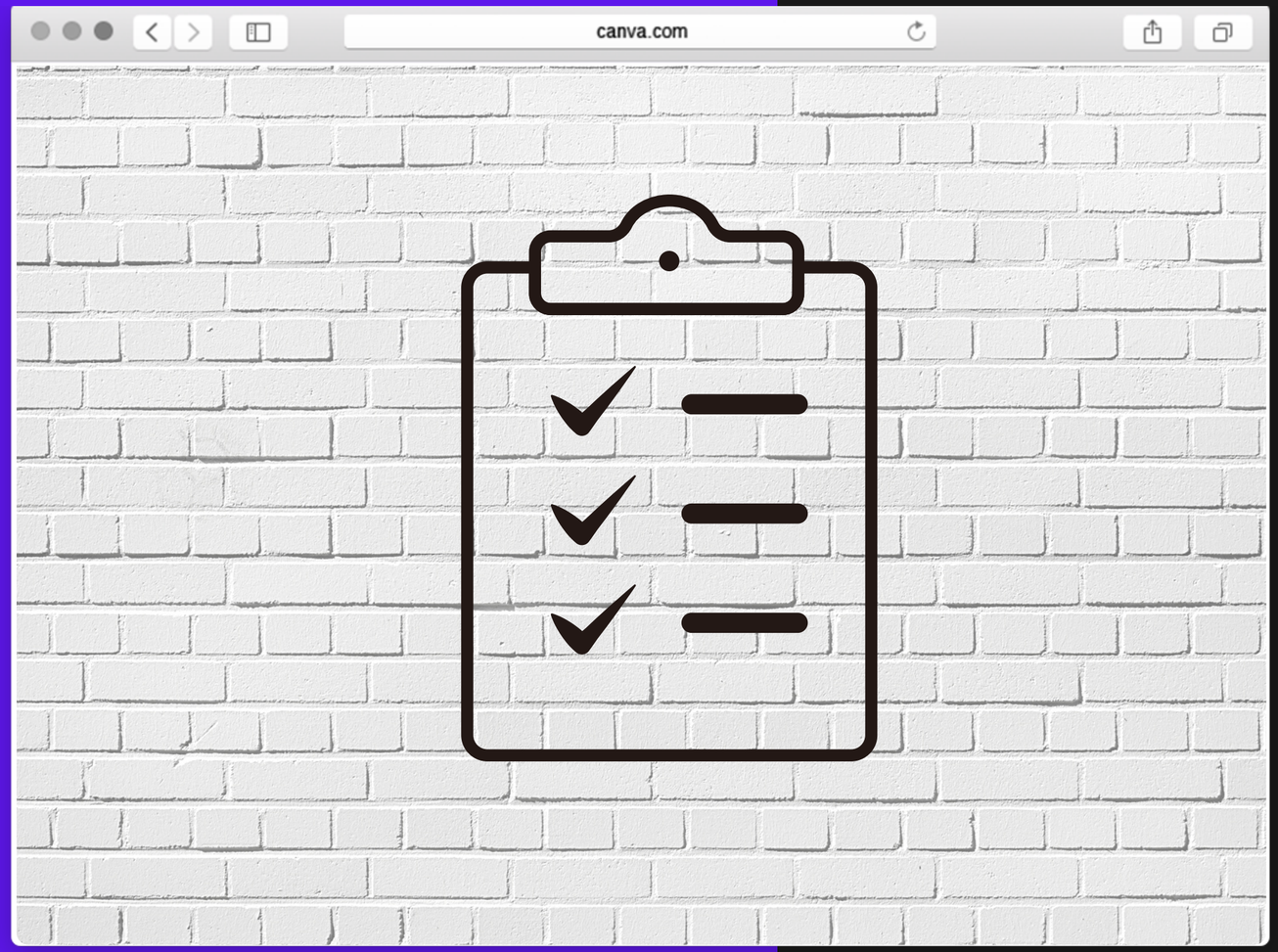
- Les partitions de données peuvent améliorer l'évolutivité en permettant aux données d'être réparties sur plusieurs serveurs.
- De cette façon, si un serveur devient surchargé, les autres peuvent prendre le relais.
- De plus, les partitions de données peuvent aider à améliorer les performances en permettant l'accès aux données en parallèle.

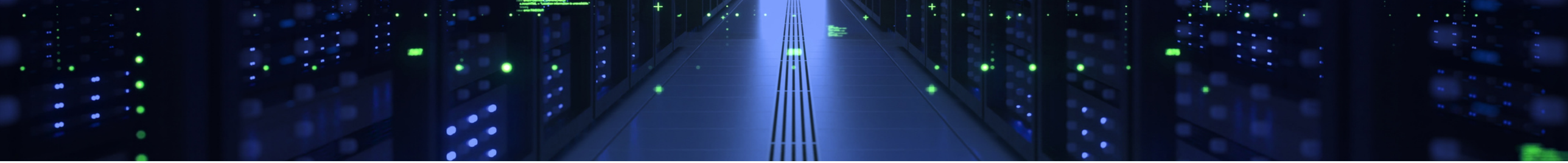
ANSWER

Réponse Question 2 :

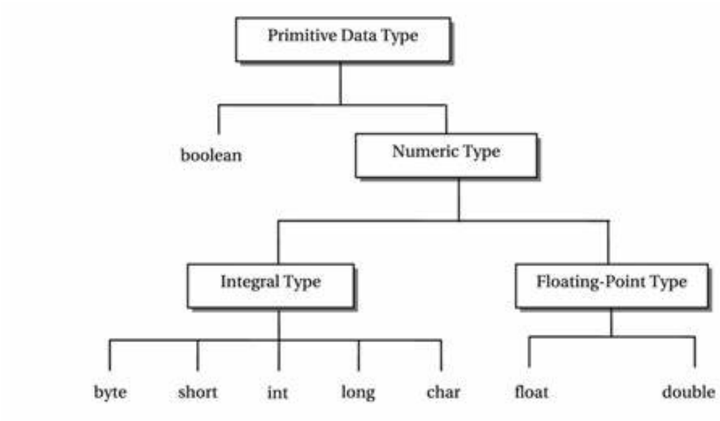


CRITÈRE A PRENDRE EN COMPTE POUR DÉCIDER





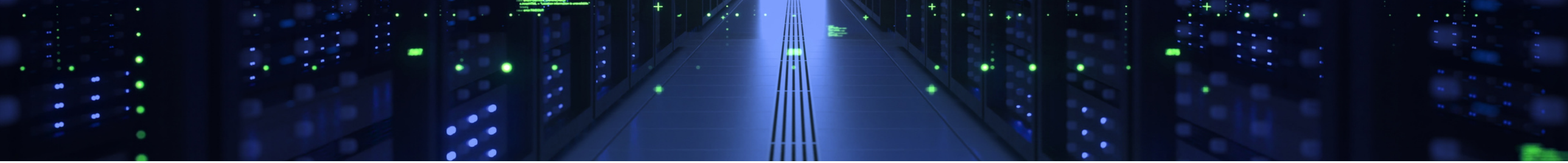
Utilisation des données



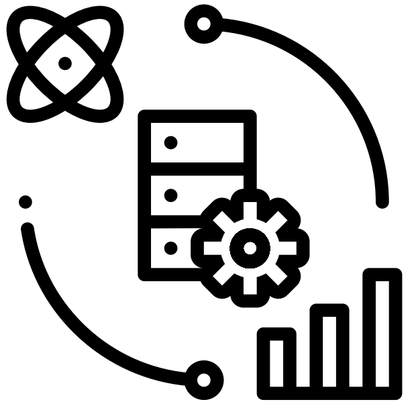
Les Type de données

Des données Perilleuse

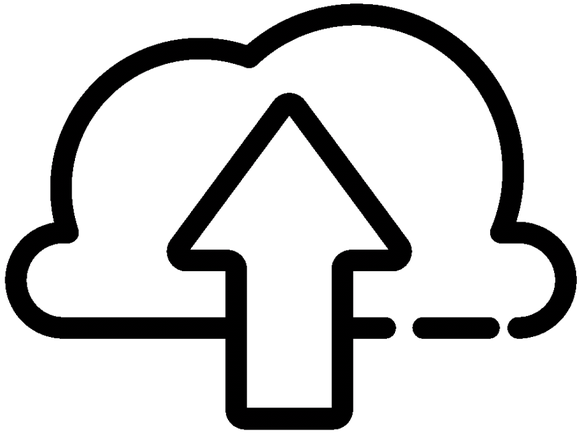
Des données Distribué



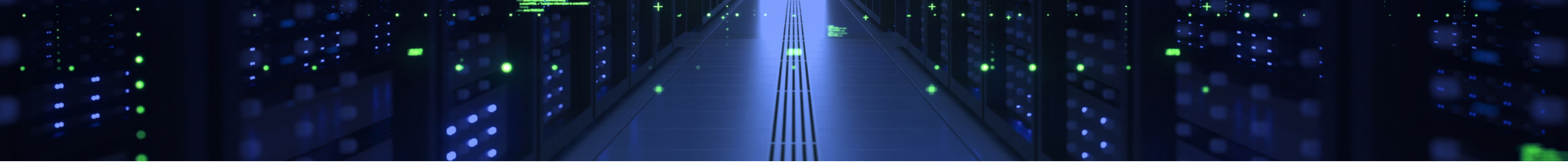
Volume de données



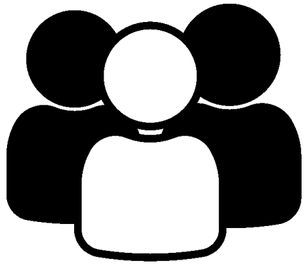
Donné Stockée doivent être
Manipulé a chaque requête



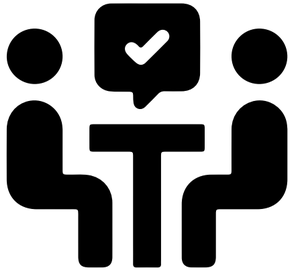
Un référentiel cohérent ne pourra pas traiter autant de données dans une requête qu’un référentiel qui ne l’est pas.



Maturité de l’équipe produit



La capacité à modéliser le domaine pour qu’il soit persisté dans le référentiel en suivant les bonnes pratiques du référentiel.

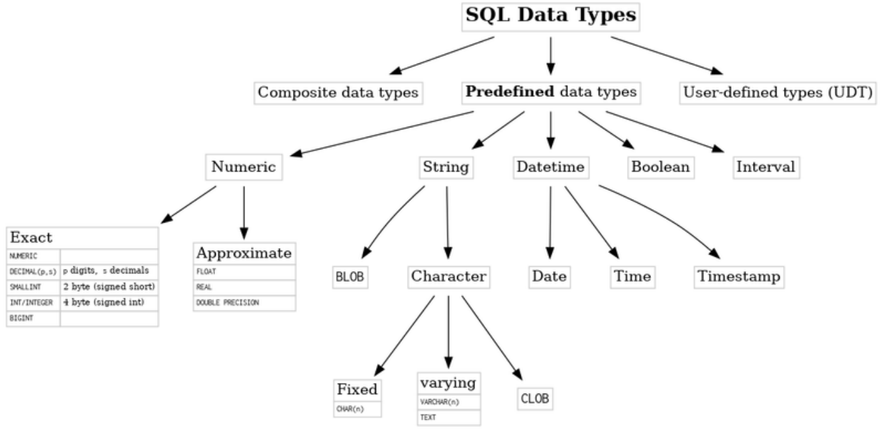


la constitution des agrégats (et donc des entités) doit être faite en accord avec le moteur de persistance.

La capacité à travailler de manière asynchrones. L’impact est réel pour les développeurs et sur la conception du code,

SGBDS POUR CHAQUE TYPE DE DONNÉ

Pour les données strucutrées



SQL data type

Projection

Jointure

Division

Agregation

Algèbre Relationnel

Personne

ID	Nom	Rôle
543	Daniel	1
544	Thomas	2
545	Chloé	2
546	Julie	3
547	Clement	3

Rôle

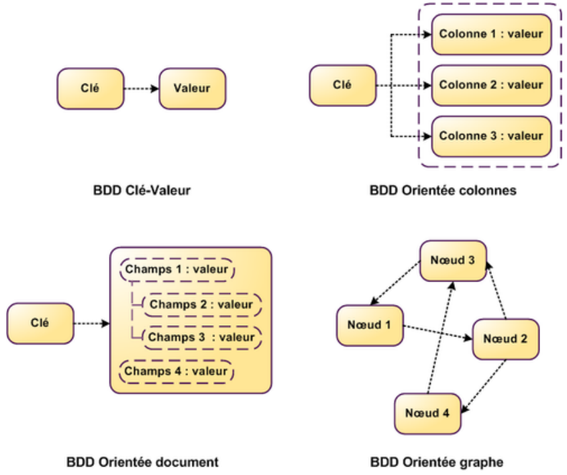
ID	Titre
1	Support
2	Responsable formation
3	Apprenant
4	Responsable académique

SGBDR Tables

MySQL® ORACLE PostgreSQL Microsoft® SQL Server SQLite

SGBDS POUR CHAQUE TYPE DE DONNÉ

Pour les données semi-structuré

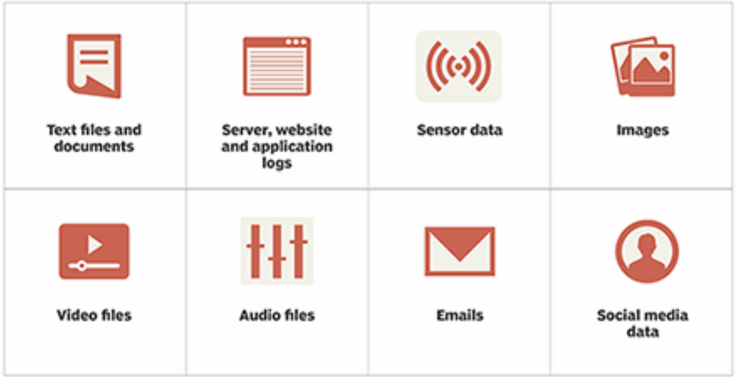


File formats

NoSQL data base structure

SGBDS POUR CHAQUE TYPE DE DONNÉ

Pour les données non-structurées

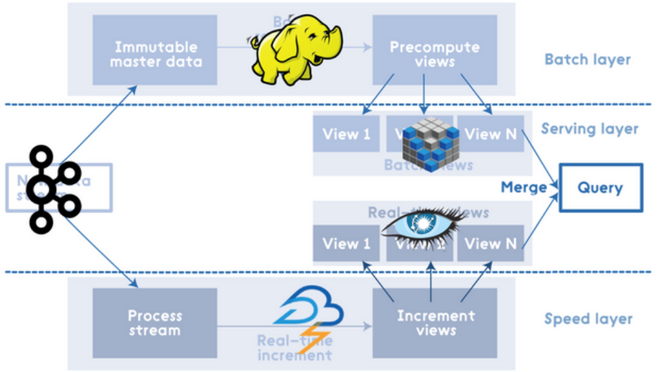


BLOB : Binary Large OBject

	type	blobParts				
Blob =	image/png	+ blob1	blob2	...	str	buffer

CLOB : Character Large OBject

BFILE: Binary File



Lambda architecture example

Amazon S3

Microsoft Azure Blob Storage

HDFS

APACHE STORM™ Distributed · Resilient · Real-time

Les Stratégie de Persistance

Le choix d'une stratégie plutôt qu'une autre va avoir des impacts pour le choix d'un référentiel cohérent ou disponible, c'est un élément important à prendre en compte dans l'architecture générale de l'application.

- Stockage en **un exemplaire** de l'état actuel du référentiel
- Stockage en **plusieurs exemplaires** de l'état actuel du référentiel
- Stockage **des événements** et calcul de l'état actuel du référentiel

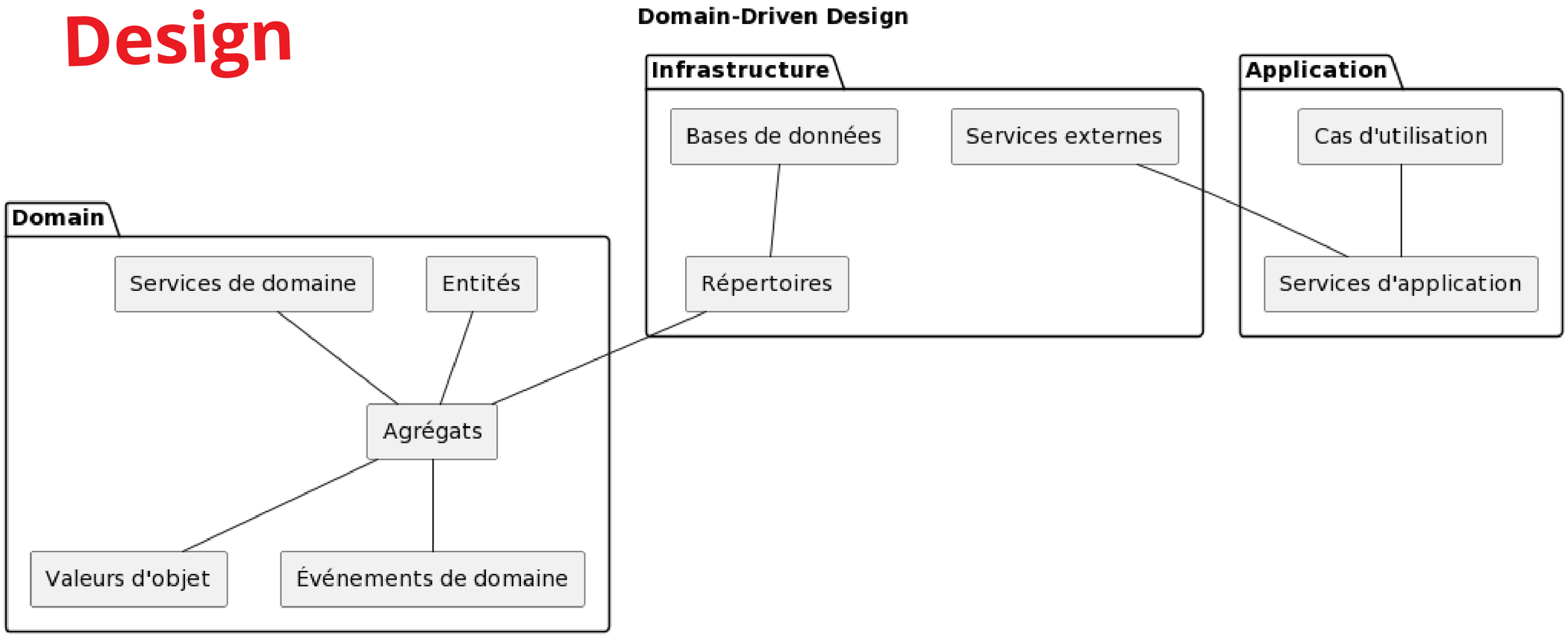
Domain Driven Design

CQRS

Event Sourcing

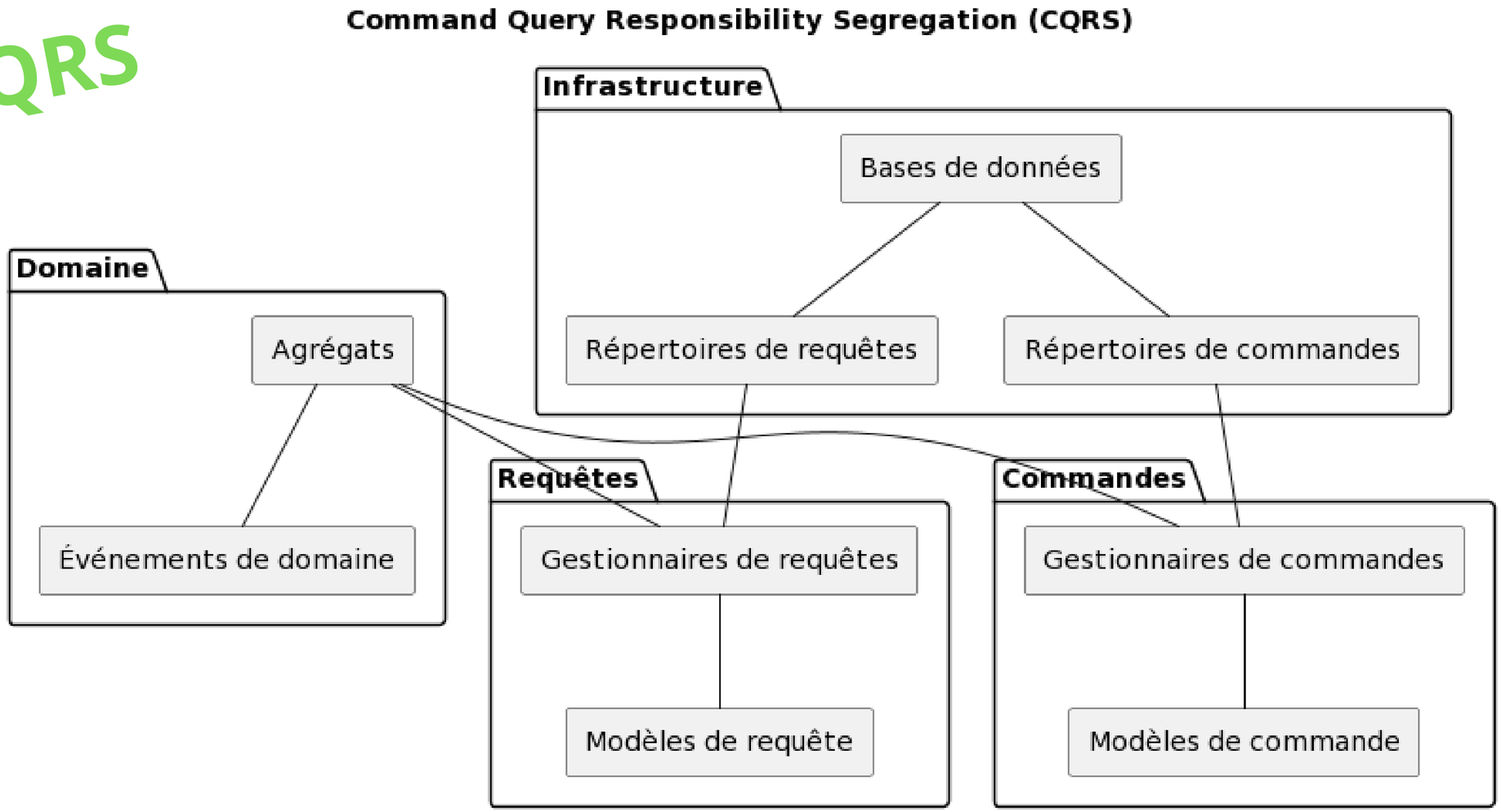
Les Stratégie de Persistance

Domain Driven Design



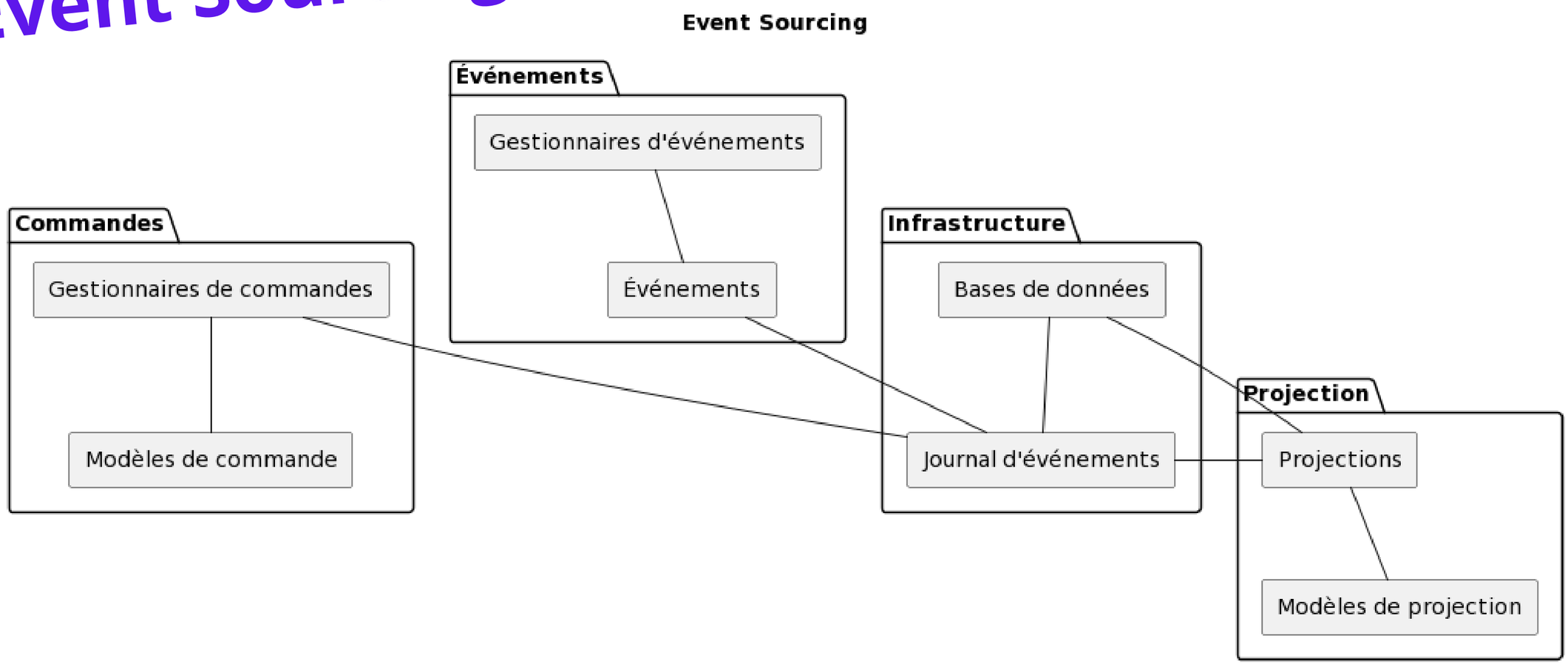
Les Stratégie de Persistance

CQRS



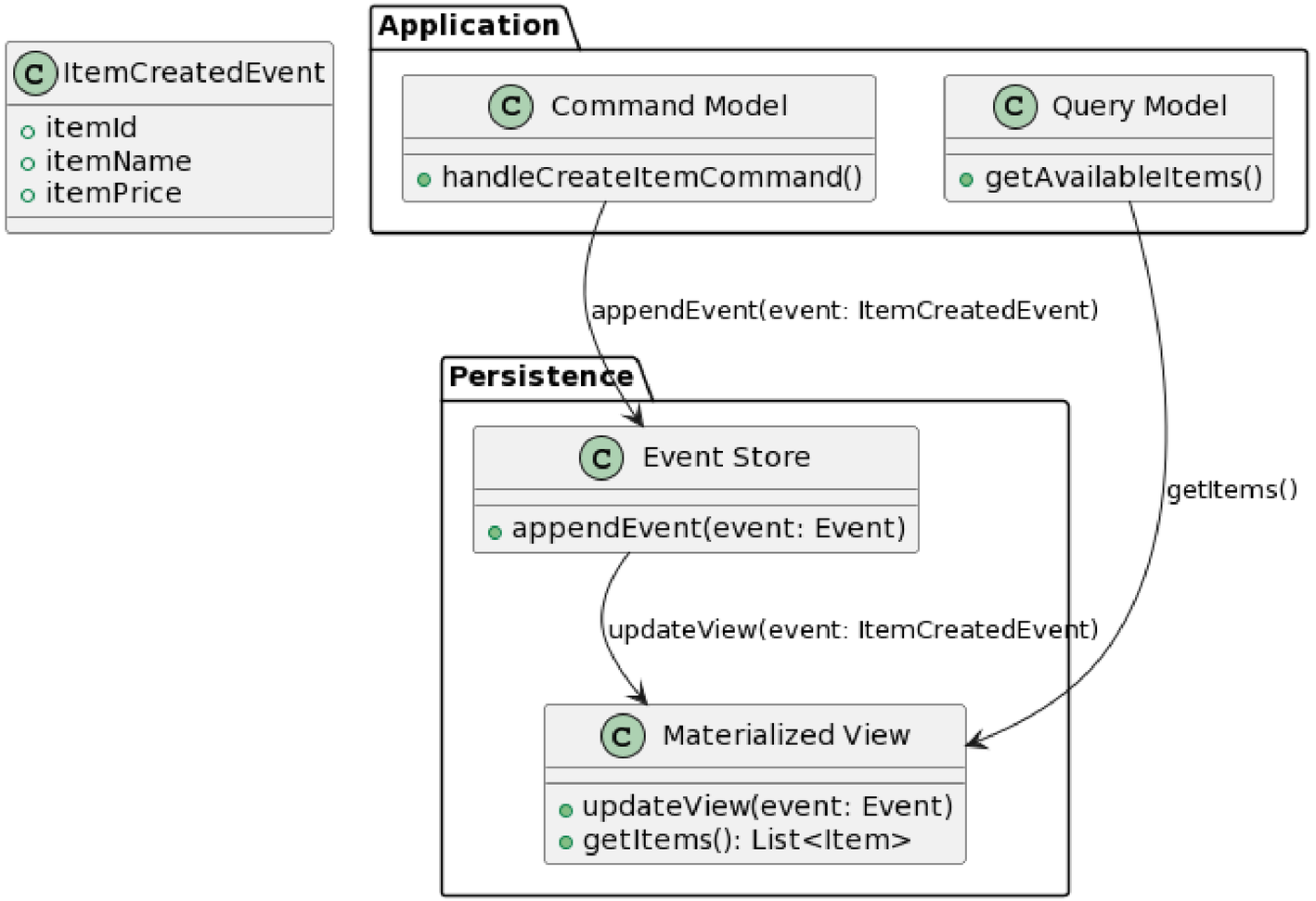
Les Stratégie de Persistance

Event Sourcing

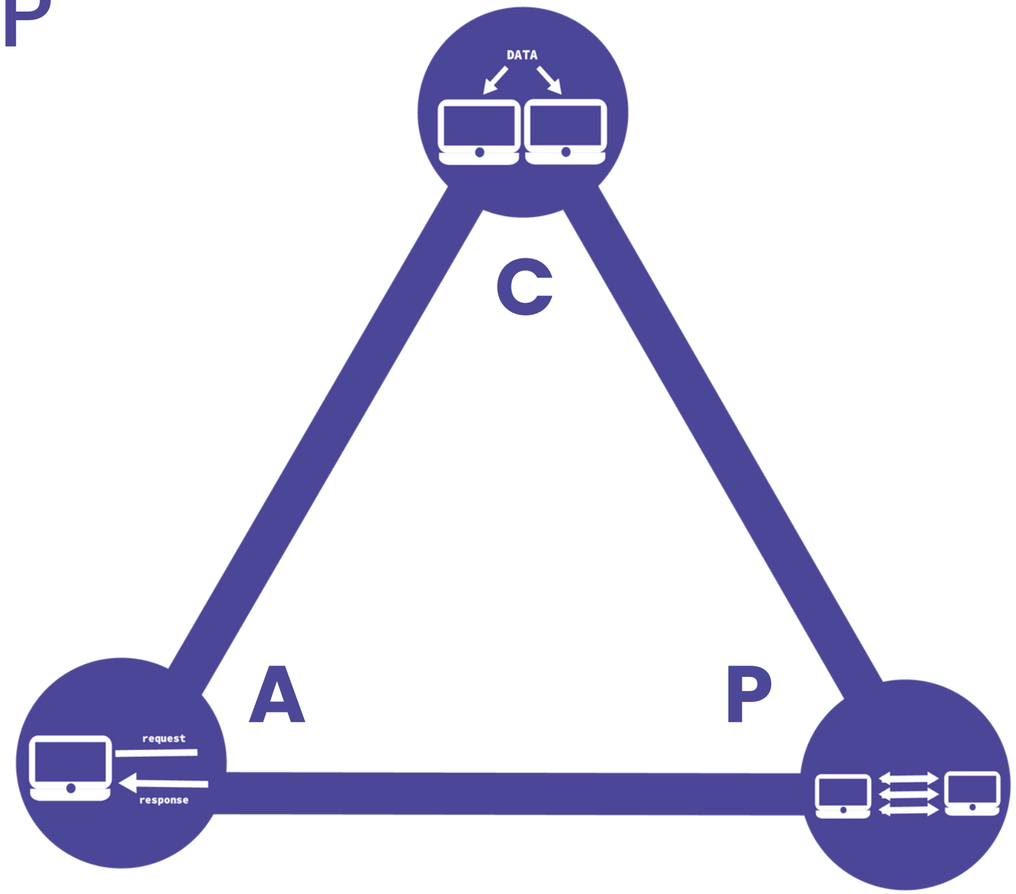


Les Stratégie de Persistance

1. Un nouvel article est ajouté au catalogue d'un site de commerce électronique.
2. Une commande de création d'article est émise.
3. Le "Command Model" reçoit la commande et effectue les validations métier.
4. Le "Command Model" génère un événement "ItemCreatedEvent" qui représente la création de l'article.
5. L'événement "ItemCreatedEvent" est enregistré dans l'"Event Store".
6. Le "Materialized View" correspondant à la liste des articles est mis à jour en fonction de l'événement.
7. Lorsqu'un utilisateur souhaite consulter la liste des articles disponibles, une requête est envoyée au "Query Model".
8. Le "Query Model" interroge le "Materialized View" pour obtenir la liste mise à jour des articles.
9. Le "Query Model" renvoie la liste des articles disponibles à l'utilisateur.



Le Triangle CAP



Raft Consensus

Raft Consensus Algorithm

Raft is a consensus algorithm that is designed to be easy to understand.

Enforced consistency

Consistency model

In computer science, a consistency model specifies a contract betwee...

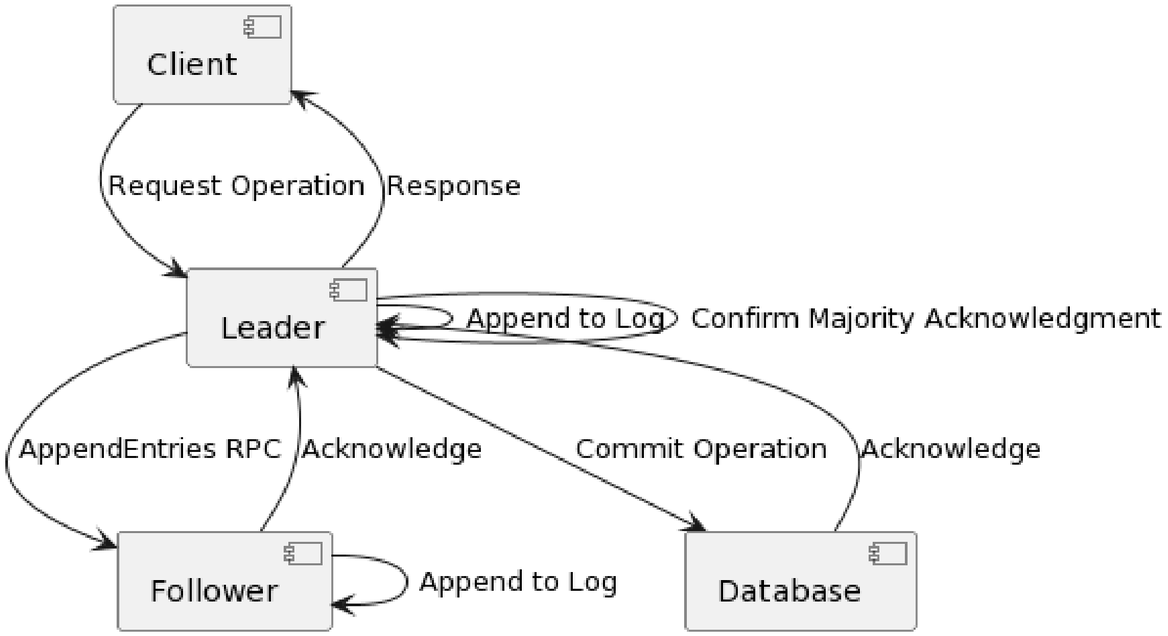
en.wikipedia.org

Eventual Consistency

Raft Consensus

L'algorithme Raft vise à maintenir la cohérence des données en gérant l'élection d'un leader, la réplication des opérations et en garantissant la sécurité des données.

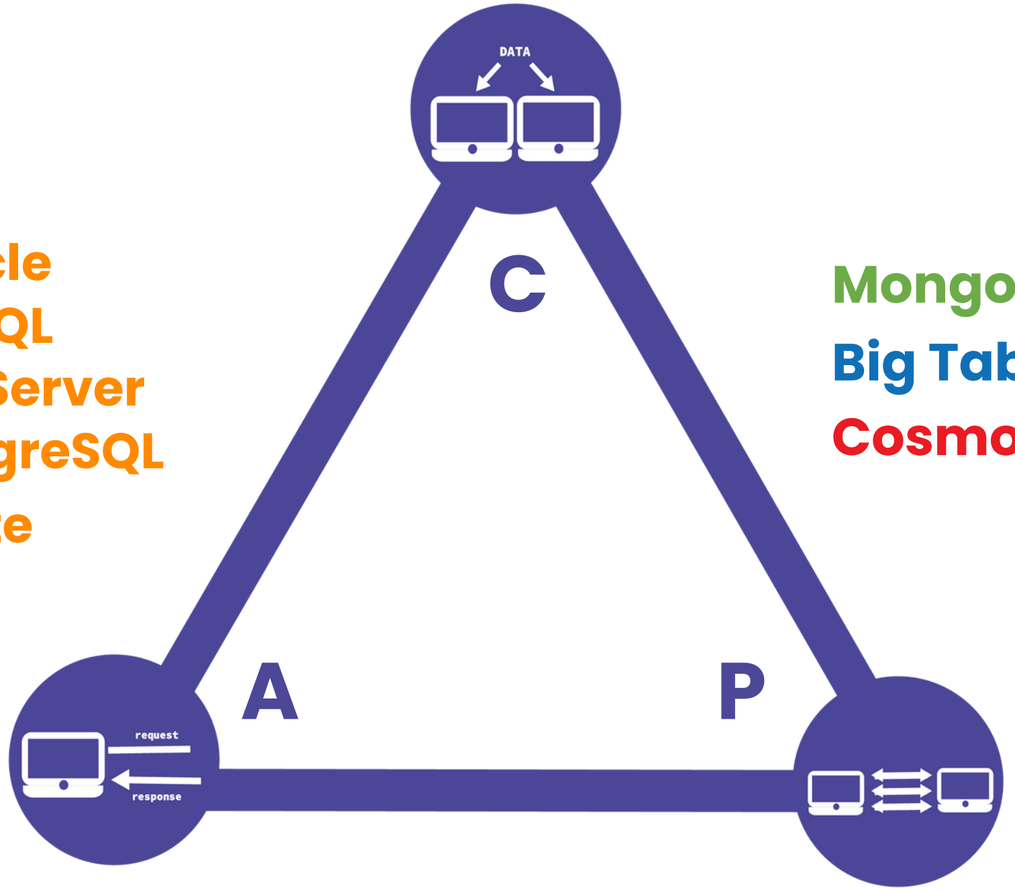
- **Élection du Leader :**
  - Un nœud devient leader s'il reçoit des votes de la majorité des autres nœuds.
- **Réplication des Journaux :**
  - Le leader gère la réplication des opérations entre les nœuds en envoyant des messages.
  - Les opérations sont confirmées lorsque la majorité des nœuds les valident.
- **Sécurité et Cohérence :**
  - Les opérations validées ne sont jamais écrasées par des opérations concurrentes.
- **Maintenance :**
  - Le leader envoie régulièrement des "heartbeats" pour maintenir son rôle.
- **Réplication et Application :**
  - Les opérations confirmées sont appliquées à l'état du système pour mettre à jour les données.



Le Triangle CAP

* : Possibilité de changer le mode

Cohérance ⟷ **Disponibilité**



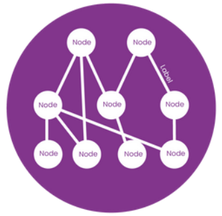
Oracle
MySQL
SQL Server
PostgreSQL
SQLite

MongoDB*
Big Table
CosmosDB*

Clé Valeur

Colonne

Document



Graph

Relationnel

Redis
Riak
Voldmort
SimpleDB

Neo4j

FireStore
DynamoDB
CouchDB*

Cassandra*
Elastic Search

NEWS ALERT

QUIZ

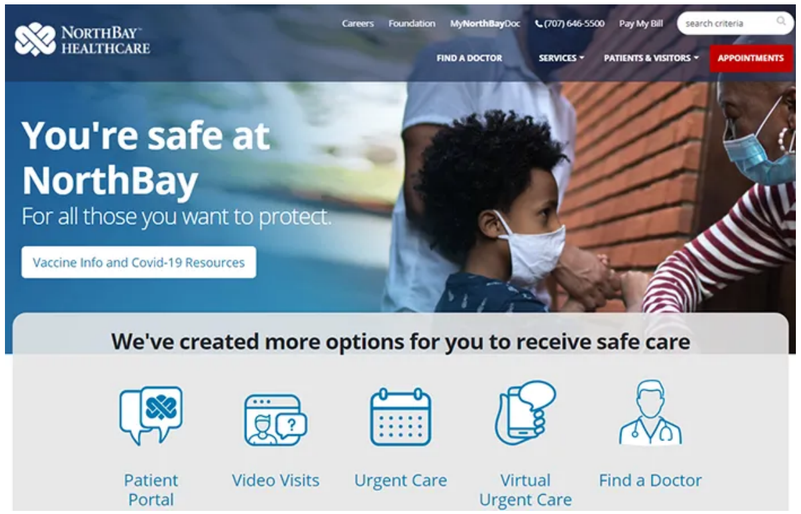
NEWS ALERT

Question 1 : Si une base de données devait fournir deux des trois propriétés (cohérence, disponibilité et tolérance de partition), laquelle serait la plus importante pour votre cas d'utilisation ?

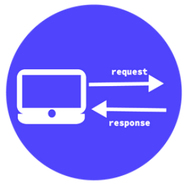
Question 2: Quelles sont les propriété qu'ils peuvent assurer les BDDs traditionnel et les BDDs Nosql

ANSWER

Reponse Question 1 : ça depend



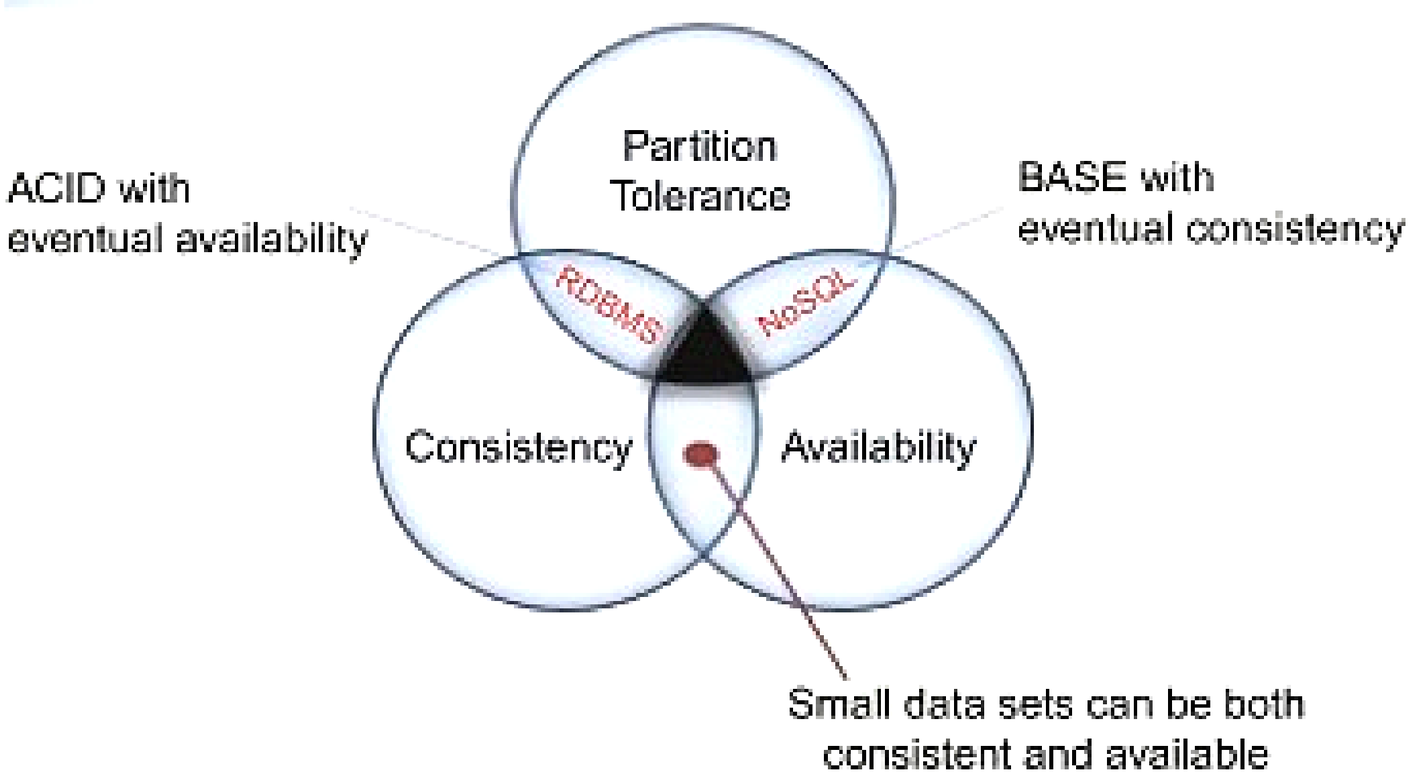
Site E-Commerce

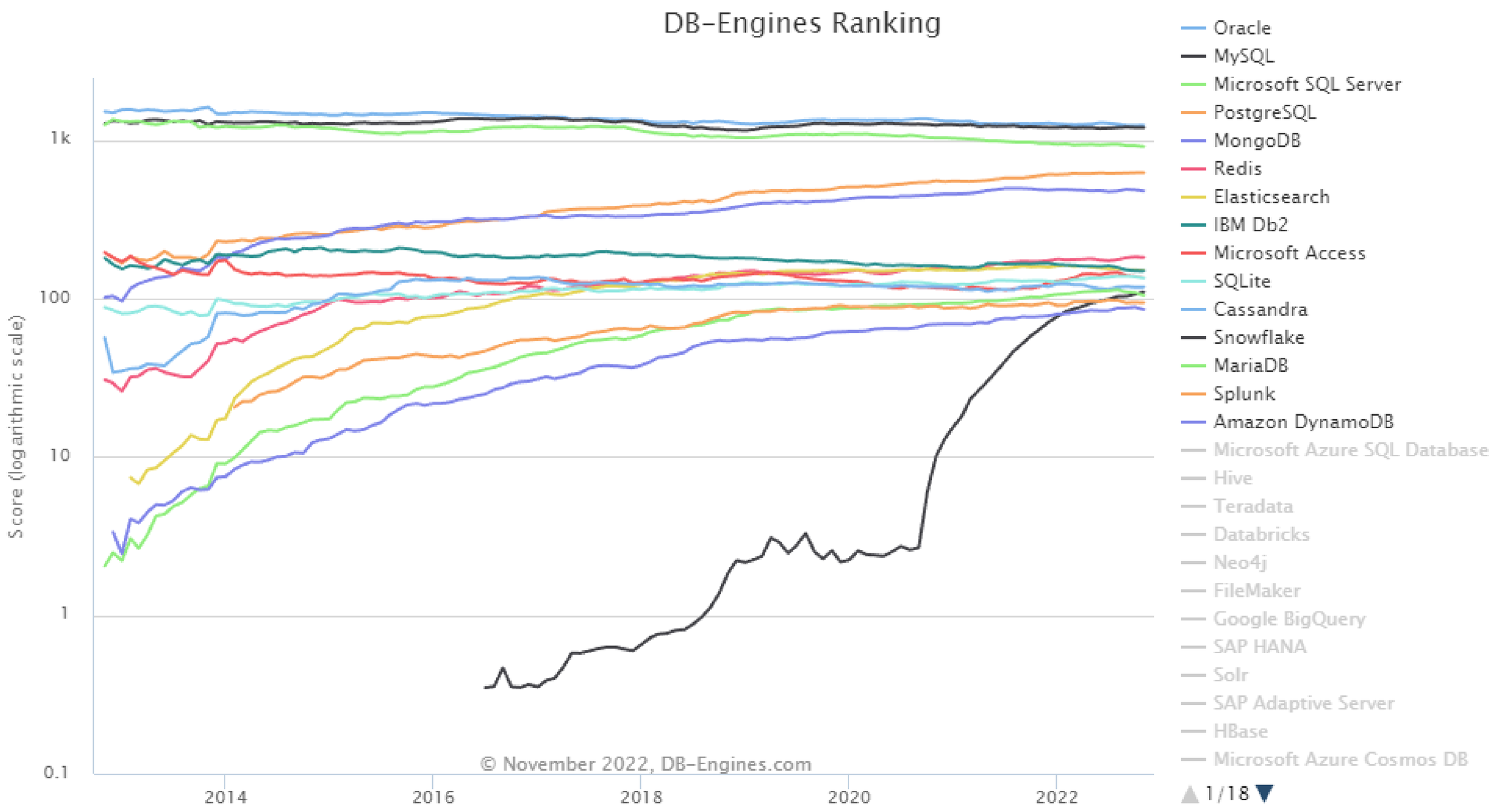


Site Medical

ANSWER

Reponse Question 2:




DB-Engines Ranking
Score (logarithmic scale)
1k
100
10
1
0.1
2014
2016
2018
2020
2022
© November 2022, DB-Engines.com
Oracle
MySQL
Microsoft SQL Server
PostgreSQL
MongoDB
Redis
Elasticsearch
IBM Db2
Microsoft Access
SQLite
Cassandra
Snowflake
MariaDB
Splunk
Amazon DynamoDB
Microsoft Azure SQL Database
Hive
Teradata
Databricks
Neo4j
FileMaker
Google BigQuery
SAP HANA
Solr
SAP Adaptive Server
HBase
Microsoft Azure Cosmos DB
1/18

TOP 1 DES SGBD NOVEMBRE 2022 SELON LE SITE DB-ENGINES

ORACLE

Relationnel

redis

Clé valeur

mongoDB

Documentaire

cassandra

Orienté Colonne

Orienté Graph

elasticsearch

Search Engine

PostGIS

Geospatial

influxdb

Time Series

EventStoreDB

Event

MarkLogic

Resource Description

Y'A T'IL UNE ALTERNATIVE AU THEOREM CAP

L’une des questions auxquelles le théorème CAP n’a pas été en mesure de répondre était

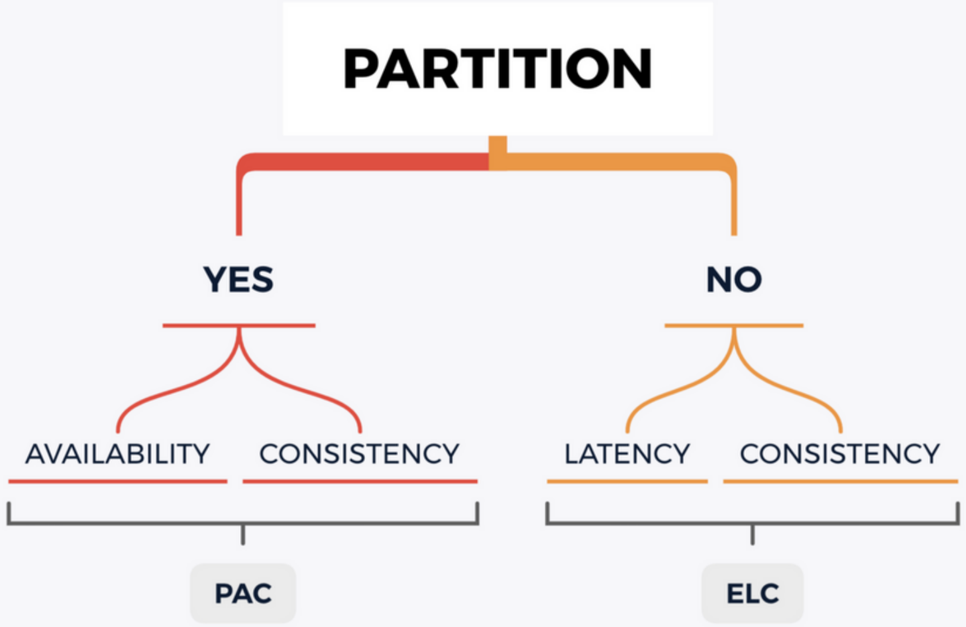
Que se passe-t-il quand il n’y a pas de partition, quelle combinaison logique un système distribua-t-il? .

LE THEOREME PACELC

- La latence est l'une des propriétés souhaitées d'un système distribué.

Latence

La latence est le temps qu'il faut au client pour recevoir une réponse et qui est régi par un certain niveau de cohérence. La latence, dans un sens, représente le degré de disponibilité.



BIBLIOGRAPHIE

- Source
  - Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services
  - cryptographic /Stack overflow
  - IPPON/ Datascientest /DB-Engines
  - Open Classeroom /Medium
- Multimedia
  - Google / Yandex/ Z-library
- Editer
  - Canvas/PPT
  - Illustrator

MERCI POUR VOTRE ATTENTION !

END