# Cours NoSQL

# Introduction à NoSQL

NoSQL (Not Only SQL) fait référence à une variété de systèmes de gestion de bases de données qui ne sont pas basés sur le modèle relationnel traditionnel. Ces bases de données sont conçues pour des besoins spécifiques tels que le traitement de grandes quantités de données, une évolutivité horizontale et des performances élevées.

## Comparaison avec MySQL

| Caractéristique | MySQL | NoSQL |
|---|---|---|
| Modèle de données | Relationnel (tables avec lignes/colonnes) | Non-relationnel (documents, graphes, clés-valeurs, colonnes) |
| Schéma | Schéma fixe et structuré | Schéma flexible |
| Langage de requête | SQL (Structured Query Language) | Varie (ex. MongoDB utilise des requêtes JSON) |
| Scalabilité | Scalabilité verticale (ajouter plus de ressources à un seul serveur) | Scalabilité horizontale (ajouter plus de serveurs) |
| Transactions | ACID (Atomicité, Cohérence, Isolation, Durabilité) | BASE (Basically Available, Soft state, Eventual consistency) |

# Histoire de NoSQL

Les bases de données NoSQL ont émergé à la fin des années 2000 en réponse à l'augmentation massive des données et des besoins en scalabilité et performance. Les pionniers comme Google, Amazon et Facebook ont mené la charge avec des solutions internes qui ont ensuite inspiré des bases de données open-source. NoSQL est devenu populaire avec le mouvement du "Big Data", nécessitant des solutions capables de gérer des volumes de données en constante croissance.

# Types de bases de données NoSQL

## Bases de données orientées documents (ex. MongoDB)

Les bases de données orientées documents stockent les données sous forme de documents (généralement JSON ou BSON). Chaque document est une collection de paires clé-valeur et peut contenir des structures imbriquées.

**Exemple:**

```
{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "address": {
    "street": "123 Main St",
    "city": "Springfield",
    "zip": "12345"
  },
  "phoneNumbers": [
    "555-1234",
    "555-5678"
  ]
}
```

## Bases de données clé-valeur (ex. Redis)

Les bases de données clé-valeur sont les plus simples, associant une clé unique à une valeur. Elles sont idéales pour des caches, des sessions et des données de configuration.

**Exemple:**

```
"user:1000" -> {
  "name": "Jane Doe",
  "email": "jane.doe@example.com"
}
```

## Bases de données orientées colonnes (ex. Cassandra)

Les bases de données orientées colonnes stockent les données dans des colonnes plutôt que dans des lignes, optimisant les lectures et écritures massives. Elles sont particulièrement efficaces pour les analyses de données en temps réel.

**Exemple:**

| Row Key | Column Family: User Details |
|---|---|
| 1001 | name: "Alice" |
| 1002 | name: "Bob" |

## Fonctionnement

Les bases de données orientées colonnes organisent les données en familles de colonnes, où chaque famille de colonnes est un ensemble de colonnes associé à une clé de ligne unique. Cela permet de regrouper les données qui sont souvent accédées ensemble, optimisant ainsi les performances des requêtes.

**Exemple de structure interne:**

```
RowKey: 1001
  ColumnFamily: UserDetails
    name: "Alice"
    email: "alice@example.com"
  ColumnFamily: Address
    street: "123 Main St"
    city: "Springfield"
```

## Bases de données orientées graphes (ex. Neo4j)

Les bases de données orientées graphes stockent les données sous forme de nœuds, de relations et de propriétés, idéales pour les données hautement connectées, comme les réseaux sociaux ou les systèmes de recommandations.

**Exemple:**

```
(Node: Person {name: "John"}) -[RELATIONSHIP: "KNOWS"]-> (Node: Person {name: "Jane"})
```

## Fonctionnement

Les bases de données orientées graphes représentent les entités sous forme de nœuds et les relations entre ces entités sous forme d'arêtes. Chaque nœud et chaque arête peut avoir des propriétés associées, permettant une modélisation flexible et riche des données relationnelles.

# Performances des bases de données NoSQL

Les bases de données NoSQL sont conçues pour offrir des performances élevées en termes de lecture et d'écriture, surtout lorsqu'elles sont mises à l'échelle horizontalement. Elles sont capables de gérer des volumes de données massifs avec une latence faible. Cependant, les performances exactes peuvent varier en fonction du type de base de données NoSQL, de la configuration matérielle, et du modèle de données spécifique utilisé.

## Avantages en termes de performances

- **Scalabilité horizontale** : Capacité à ajouter plus de serveurs pour distribuer la charge.
- **Modèle de données flexible** : Adaptation facile aux changements de schéma et aux besoins de l'application.
- **Haute disponibilité** : Réplication des données et tolérance aux pannes intégrées.

## Limitations

- **Cohérence éventuelle** : Certaines bases de données NoSQL offrent une cohérence éventuelle plutôt qu'une cohérence immédiate, ce qui peut ne pas convenir à toutes les applications.
- **Complexité de gestion** : Gérer une infrastructure distribuée peut être plus complexe.

# Installation de MongoDB et intégration avec PHP

# Installation de MongoDB

## Sur Ubuntu

```
wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb
sudo apt-get update
sudo apt-get install -y mongodb-org
sudo systemctl start mongod
sudo systemctl enable mongod
```

## Sur Windows

1. Télécharger l'installateur depuis le site officiel de MongoDB.
2. Suivre les instructions de l'assistant d'installation.
3. Démarrer le service MongoDB via `Services.msc` ou en ligne de commande.

# Intégration de MongoDB avec PHP

## Installation du driver MongoDB pour PHP

1. Installer l'extension MongoDB pour PHP:

```
sudo pecl install mongodb
```

2. Ajouter l'extension dans le fichier `php.ini` :

```
extension=mongodb.so
```

## Utilisation de MongoDB dans un script PHP

```php
<?php
require 'vendor/autoload.php'; // inclure l'autoloader de Composer

$client = new MongoDB\Client("mongodb://localhost:27017");

$collection = $client->test->users;

$result = $collection->insertOne([
    'name' => 'Alice',
    'email': 'alice@example.com',
    'phone' => '123-456-7890'
]);

echo "Inserted with Object ID '{$result->getInsertedId()}'";
?>
```

Pour installer Composer et l'utiliser:

1. Télécharger et installer Composer depuis getcomposer.org.
2. Créer un fichier `composer.json` :

```json
{
    "require": {
        "mongodb/mongodb": "^1.9"
    }
}
```

3. Installer les dépendances:

## Opérateur `$gt`

L'opérateur `$gt` signifie "greater than" (plus grand que) et est utilisé pour comparer des valeurs dans des documents MongoDB. Il sélectionne les documents où la valeur d'un champ est strictement supérieure à une valeur spécifiée.

## Exemple

Si vous avez une collection d'employés et vous voulez récupérer uniquement ceux dont le salaire est supérieur à 1000, vous utiliserez `$gt` dans votre requête de la manière suivante :

```
$filter = ['salaire' => ['$gt' => 1000]];
```

## Explication détaillée du filtre

- `'salaire'` : Le champ sur lequel vous appliquez la condition.
- `'$gt'` : L'opérateur qui signifie "strictement supérieur à".
- `1000` : La valeur avec laquelle vous comparez les salaires dans la collection. La condition sélectionnera uniquement les documents où le champ `salaire` est strictement supérieur à 1000.

## Autres opérateurs de comparaison

MongoDB fournit plusieurs autres opérateurs de comparaison similaires à `$gt`, notamment :

- `$gte` : Greater than or equal to (supérieur ou égal à).
- `$lt` : Less than (inférieur à).
- `$lte` : Less than or equal to (inférieur ou égal à).
- `$eq` : Equal to (égal à).
- `$ne` : Not equal to (différent de).

## Exemple de code avec d'autres opérateurs

Si vous souhaitez récupérer des employés dont le salaire est compris entre 1000 et 2000 euros, vous pouvez utiliser `$gte` et `$lt` ainsi :

```
$filter = [
    'salaire' => [
        '$gte' => 1000,
        '$lt' => 2000
    ]
];
```

# Application dans le code complet

Voici le code PHP complet avec le filtre pour les salaires supérieurs à 1000 euros et les noms commençant par "B" :

```php
<?php
// Chargement de l'extension MongoDB via Composer autoload
require 'vendor/autoload.php'; // Assurez-vous que Composer est installé et que vous avez la dépendance "mongodb/mongodb" dans votre fichier co

// Connexion à la base de données MongoDB
$client = new MongoDB\Client("mongodb://localhost:27017");

// Sélection de la base de données et de la collection
$database = $client->selectDatabase('nom_de_votre_base_de_donnees');
$collection = $database->selectCollection('employer');

// Définition des critères de sélection
$filter = [
    'salaire' => ['$gt' => 1000], // Utilisation de $gt pour filtrer les salaires supérieurs à 1000
    'nom' => ['$regex' => '^B', '$options' => 'i'] // Filtrer les noms commençant par 'B', insensible à la casse
];

// Récupération des documents correspondant aux critères
$employes = $collection->find($filter);

// Affichage des résultats
foreach ($employes as $employe) {
    echo "Nom: " . $employe['nom'] . "\n";
    echo "Poste: " . $employe['poste'] . "\n";
    echo "Salaire: " . $employe['salaire'] . " euros\n";
    echo "-------------------------\n";
}
```