**Comparaison entre MongoDB (NoSQL) et SQL**

.

**Fonctionalités de SQL**

* **Architecture** : Structurée en tables avec relations (clés primaires et étrangères).
* **Langage** : Utilise SQL pour les requêtes (SELECT, JOIN, etc.).
* **Intégrité des données** : Forte grâce à des schémas rigides et normalisation.
* **Exemples** : MySQL, PostgreSQL, Oracle DB.
* **Cas d'utilisation** : Systèmes nécessitant des relations complexes (ERP, banques).

**Fonctionnalités de MongoDB**

* **Architecture** : Basée sur des documents JSON/BSON.
* **Langage** : Requêtes en JSON-like.
* **Flexibilité** : Absence de schéma rigide (permet des évolutions rapides).
* **Scalabilité horizontale** : Optimisé pour la gestion de grandes quantités de données.
* **Cas d'utilisation** : Applications en temps réel, big data, IoT.

**Comparaison SQL vs MongoDB**

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspect** | **SQL** | | **MongoDB** |
| **Structure des données** | Tables, lignes, colonnes | | Documents JSON/BSON |
| **Langage de requête** | SQL (standardisé) | | JSON-like (MongoDB Query Langage) |
| **Schéma** | Rigide | | Flexible |
| **Performance** | Bonne pour transactions complexes | | Optimisé pour données massives |
| **Scalabilité** | Verticale | | Horizontale |
| Données non structurées (IoT, big data) | |

**Conclusion**

* **SQL** : Idéal pour les bases relationnelles et les systèmes nécessitant une forte intégrité des données.
* **MongoDB** : Meilleur choix pour les données non structurées et les applications nécessitant une scalabilité élevée.
* **Choix dépend des besoins spécifiques** :
  + Relations complexes → SQL.
  + Flexibilité et vitesse → MongoDB.