MongoDB:

- **Data Model:** MongoDB is a NoSQL database that uses a flexible document data model. It stores data in collections of JSON-like documents, allowing for nested structures.

- **Schema Flexibility:** MongoDB offers schema flexibility, enabling dynamic schema changes. Fields can vary between documents in the same collection.

- **Availability:** MongoDB supports high availability through replica sets. It provides automatic failover and data redundancy.

- **Consistency:** MongoDB offers tunable consistency, allowing for eventual consistency or strong consistency, depending on configuration.

- **Scalability:** MongoDB scales horizontally by sharding data across multiple servers. This enables it to handle large datasets and high traffic loads.

- **Read/Write Performance:** MongoDB provides good read and write performance, particularly for document-based queries. However, complex joins and aggregations can impact performance.

- **Other Features:**
  - Supports geospatial indexing and queries.
  - Rich querying capabilities with a flexible query language.
  - Automatic sharding for horizontal scaling.
  - Support for transactions in recent versions.

Neo4j:

- **Data Model:** Neo4j is a graph database that models data as nodes, relationships, and properties. It is designed for data with complex, interconnected relationships.

- **Schema Flexibility:** Neo4j's schema is flexible and dynamic. It's well-suited for evolving data with rich connections.

- **Availability:** Neo4j offers high availability configurations, including master-slave setups for redundancy.

- **Consistency:** Neo4j provides strong consistency by default. It ensures that data remains consistent even in the presence of failures.

- **Scalability:** Neo4j scales vertically, meaning it works well with datasets that fit in a single server's memory. Horizontal scalability is possible but less common.

- **Read/Write Performance:** Neo4j excels in read-heavy workloads and complex queries involving traversals of graph structures. Write performance can be impacted by the complexity of relationships.

- **Other Features:**
  - Native support for graph traversal and querying.
  - ACID compliance for data integrity.
  - Cypher query language for expressive graph queries.
  - Built-in graph algorithms and visualization tools.

InfluxDB:

- **Data Model:** InfluxDB is a time-series database designed for storing and querying time-stamped data points. Data is organized into measurements, tags, and fields.

- **Schema Flexibility:** InfluxDB offers some flexibility with tags and fields but requires a predefined schema. Schemas can evolve but may require data migration.

- **Availability:** InfluxDB supports clustering for high availability and data redundancy.

- **Consistency:** InfluxDB provides tunable consistency levels, including quorum-based consistency for data replication.

- **Scalability:** InfluxDB scales horizontally by adding more nodes to the cluster. It is optimized for time-series data and high write throughput.

- **Read/Write Performance:** InfluxDB excels in write-heavy scenarios, making it ideal for time-series data. Query performance is strong for time-based aggregations.

- **Other Features:**
  - Built-in support for retention policies and continuous queries.
  - SQL-like query language (InfluxQL) for querying time-series data.
  - Integrates with Grafana and other visualization tools.
  - Good for monitoring and IoT use cases.

Cassandra:

- **Data Model:** Cassandra is a columnar NoSQL database. It uses a tabular data model with rows and columns.

- **Schema Flexibility:** Cassandra provides flexibility in column design within a column family. Columns can be added or modified without affecting existing data.

- **Availability:** Cassandra is known for its high availability and fault tolerance. It uses a peer-to-peer architecture with no single point of failure.

- **Consistency:** Cassandra offers tunable consistency levels, allowing for trade-offs between consistency and availability.

- **Scalability:** Cassandra scales horizontally by adding more nodes to the cluster. It is designed for linear scalability.

- **Read/Write Performance:** Cassandra provides excellent write performance, making it suitable for write-heavy workloads. Read performance is also strong for simple queries.

- **Other Features:**
  - Distributed architecture with no single point of failure.
  - Tunable data replication and consistency.
  - Support for time-to-live (TTL) data expiration.
  - Good for use cases requiring high availability and scalability, like IoT and real-time analytics.