1. **Describe in detail how entities, attributes and relationships are organised and represented differently in relational databases compared with document stores such as MongoDB. Explain the advantages and disadvantages of each approach. Illustrate your answer with examples from real applications.**

**Answer:**

Entities, attributes, and relationships are fundamental concepts in data modeling. They describe the structure and organization of data in a database system. Relational databases and document stores such as MongoDB have different approaches to organizing and representing these concepts.

In relational databases, entities are represented as tables with rows and columns. Each row represents a unique entity instance, and each column represents an attribute of that entity. For example, a customer entity can be represented as a table called "Customers" with columns such as "customer\_id," "name," "email," and "phone\_number." Relationships between entities are represented by establishing foreign key constraints between tables. For instance, if a customer places an order, the order table would contain a foreign key reference to the customer table, linking the two tables together.

On the other hand, document stores such as MongoDB represent entities as JSON documents. Each document can have a different structure and set of attributes, and relationships between entities are represented as nested documents or arrays within the parent document. For example, a customer entity can be represented as a JSON document containing attributes such as "customer\_id," "name," "email," and "phone\_number," and an orders array containing nested documents representing orders placed by the customer.

**Advantages of relational databases include:**

Structured data: Relational databases enforce a strict schema for data, which helps ensure data consistency and integrity.

ACID transactions: Relational databases support ACID (Atomicity, Consistency, Isolation, and Durability) transactions, which ensure that data remains consistent even in the face of system failures or concurrent access by multiple users.

Easy data querying: Relational databases use SQL, a standard query language, which makes it easy to perform complex queries on data.

High performance: Relational databases are optimized for handling large volumes of data, and their use of indexes and other optimizations can provide high performance even for complex queries.

**Disadvantages of relational databases include:**

Limited flexibility: The strict schema of relational databases can limit the flexibility of data modeling and make it difficult to adapt to changing business requirements.

Scaling limitations: Scaling relational databases can be difficult, particularly when dealing with large datasets or high levels of concurrent access.

High overhead: The ACID properties of relational databases come at a cost in terms of performance and system overhead.

**Advantages of document stores like MongoDB include:**

Flexibility: Document stores are schemaless, meaning that data can be easily adapted to changing business requirements without the need to modify a strict schema.

Scalability: Document stores can be easily scaled horizontally by adding more nodes, making it easier to handle large volumes of data or high levels of concurrent access.

High performance: Document stores can provide high performance for queries that involve nested data structures or require complex joins.

**Disadvantages of document stores include:**

Lack of structure: The lack of a strict schema in document stores can lead to data inconsistencies and make it difficult to enforce data integrity.

Limited querying capabilities: Document stores often have limited query capabilities compared to relational databases, making it difficult to perform complex queries or aggregations.

Limited transactional support: Document stores often lack transactional support, making it difficult to ensure data consistency in the face of system failures or concurrent access.

1. **Explain what is meant by aggregation when querying a database. Compare and contrast the ways of performing aggregations in SQL and with MongoDB. Illustrate your answer with examples of SQL and MongoDB code.**

**Answer:**

Aggregation refers to the process of performing calculations and summarizations on data in a database to generate meaningful insights. This is often done to obtain useful information from large datasets. Aggregations can include operations such as counting, summing, averaging, grouping, and sorting.

In SQL, aggregations are typically performed using the GROUP BY clause, which groups rows based on a specified column or set of columns, and the use of aggregate functions such as COUNT, SUM, AVG, MAX, and MIN. For example, the following SQL code calculates the total number of orders by customer:

SELECT customer\_id, COUNT(\*) AS order\_count

FROM orders

GROUP BY customer\_id;

In MongoDB, aggregations are performed using the aggregation pipeline, which is a sequence of stages that transform and process the data. Each stage performs a specific operation on the data and passes the output to the next stage in the pipeline. The stages can include operators for grouping, filtering, projecting, sorting, and calculating aggregate functions. For example, the following MongoDB code calculates the total number of orders by customer using the $group operator:

db.orders.aggregate([

{ $group: {

\_id: "$customer\_id",

order\_count: { $sum: 1 }

}}

]);

Note that the syntax and functionality of the aggregate functions may vary slightly between SQL and MongoDB.

There are some key differences between performing aggregations in SQL and with MongoDB. One major difference is that SQL is a relational database management system, while MongoDB is a document-oriented database. This means that SQL uses tables to store data, while MongoDB uses JSON documents. As a result, SQL is better suited for handling structured data and complex joins, while MongoDB is better suited for handling unstructured or semi-structured data.

1. **Describe both the ACID properties of transactions and the components of CAP theorem. Compare how the different elements of each are supported by relational and non-relational databases. Illustrate your answer with examples from real applications.**

**Answer:**

**ACID properties of transactions:**

ACID stands for Atomicity, Consistency, Isolation, and Durability. These properties are essential to ensure the reliability and integrity of transactions in a database.

**Atomicity:** A transaction is an indivisible unit of work that must be treated as a single operation. Atomicity ensures that either all of the changes made to the database during the transaction are committed or none of them are committed. For example, when transferring money from one bank account to another, the transaction must either complete successfully or fail without making any changes.

**Consistency:** A transaction must ensure that the database remains in a consistent state after the transaction is completed. Consistency guarantees that the data in the database satisfies all the constraints and rules defined for the system. For example, if a database requires that every user has a unique ID, the transaction must ensure that the ID is not duplicated.

**Isolation:** A transaction must be executed in isolation from other transactions, which ensures that concurrent transactions do not interfere with each other. Isolation is important to prevent conflicts and maintain the integrity of the database. For example, if two transactions try to update the same record simultaneously, isolation ensures that the changes are applied in the correct order and do not conflict with each other.

**Durability:** Once a transaction is committed, its changes must be permanent and survive any subsequent failures, such as system crashes or power outages. Durability ensures that the data is not lost or corrupted due to hardware or software failures. For example, if a transaction adds a new row to a table, the data must be saved to disk to ensure durability.

Relational databases are designed to provide strong support for ACID properties. They use a write-ahead logging mechanism to ensure that the data is durable, and they employ locking mechanisms to provide isolation and consistency.

Non-relational databases, on the other hand, may not provide full support for all ACID properties. For example, some NoSQL databases sacrifice consistency and isolation in favor of high availability and scalability.

**CAP theorem components:**

The CAP theorem, also known as Brewer's theorem, states that it is impossible for a distributed database system to simultaneously provide all three of the following guarantees:

**Consistency:** Every read receives the most recent write or an error.

**Availability:** Every request receives a response, without guarantee that it contains the most recent version of the information.

**Partition tolerance:** The system continues to operate despite arbitrary partitioning due to network failures.

Relational databases typically prioritize consistency over availability and partition tolerance. They use locking mechanisms and distributed transactions to ensure that the data is consistent, even in the presence of network partitions.

Non-relational databases, on the other hand, may prioritize availability and partition tolerance over consistency. They often use mechanisms such as eventual consistency, where data may be inconsistent for a short period of time before being eventually reconciled.

For example, MongoDB is a document-oriented NoSQL database that provides high availability and partition tolerance, but may sacrifice consistency in some cases. It uses a mechanism called write concern to allow users to specify the level of consistency they require. In contrast, PostgreSQL is a relational database that provides strong support for consistency, but may sacrifice availability and partition tolerance in some cases.

In summary, ACID properties and the CAP theorem represent two different approaches to ensuring the reliability and integrity of data in a database system. Relational databases typically provide strong support for ACID properties, while non-relational databases may prioritize availability and partition tolerance over consistency. The choice of database system will depend on the specific requirements of the application and the nature of the data being stored.