Questions and Answers

What is the difference between a list where memory is contiguously allocated and a list where linked structures are used? A contiguously allocated list (array) stores elements in consecutive memory locations, allowing fast random access in O(1) time but making insertions and deletions in the middle inefficient due to shifting elements. A linked list stores elements in nodes with pointers to the next (or previous) node, making insertions and deletions efficient (O(1) in some cases) but requiring O(n) time for random access since traversal is needed.

When are linked lists faster than contiguously-allocated lists? Linked lists are faster when frequent insertions and deletions occur in the middle of the list, as no shifting is required. They are also better for dynamic memory allocation since arrays require resizing. However, arrays are still preferable for fast indexing and better cache locality.

Add 23 to the AVL Tree below. What imbalance case is created with inserting 23?

```
        30

     / \

    25  35

   /

  20
```

Inserting 23 results in:

```
        30

     / \

    25  35

   /

  20

    \

    23
```

The balance factor of 20 becomes -1, and the balance factor of 25 becomes 2, creating an imbalance at 25. This is a Left-Right (LR) Case, requiring a left rotation on 20 followed by a right rotation on 25.

Why is a B+ Tree better than an AVL tree when indexing a large dataset? B+ Trees are optimized for disk-based indexing, while AVL Trees are optimized for memory-based indexing. B+ Trees have a higher branching factor, reducing tree height and minimizing disk I/O. They also store all data in leaf nodes, making range queries and sequential scans more efficient. AVL Trees, being binary, require more nodes and deeper tree structures, leading to more frequent disk accesses.

What is disk-based indexing and why is it important for database systems? Disk-based indexing refers to storing index structures on disk rather than in memory, allowing efficient access to large datasets that exceed RAM capacity. It is important because databases often store terabytes of data, and efficient indexing structures like B-Trees and B+ Trees reduce disk reads, improving query performance.

In the context of a relational database system, what is a transaction? A transaction is a sequence of one or more database operations that execute as a single unit of work, ensuring consistency, isolation, and durability.

Succinctly describe the four components of ACID-compliant transactions. Atomicity ensures all operations in a transaction succeed or none do. Consistency maintains database integrity before and after the transaction. Isolation prevents transactions from interfering with each other. Durability ensures committed transactions persist even in system failures.

Why does the CAP principle not make sense when applied to a single-node MongoDB instance? The CAP theorem applies to distributed systems where consistency, availability, and partition tolerance must be balanced. A single-node MongoDB instance does not experience network partitions, so it inherently provides both consistency and availability unless it crashes.

Describe the differences between horizontal and vertical scaling. Horizontal scaling (scaling out) adds more machines to distribute the load, improving redundancy and fault tolerance (e.g., NoSQL databases like MongoDB). Vertical scaling (scaling up) adds more resources (CPU, RAM) to a single machine, which is simpler but limited by hardware constraints (e.g., traditional RDBMS).

Briefly describe how a key/value store can be used as a feature store. A feature store manages machine learning features efficiently. Key/value stores like Redis store precomputed feature values for quick retrieval in O(1) time. Example: storing user features with user_id as the key and their ML feature set as the value.

When was Redis originally released? Redis was released in 2009 by Salvatore Sanfilippo.

In Redis, what is the difference between the INC and INCR commands? INC is not a Redis command. INCR increments the value of a key by 1. Example:

```
SET counter 10

INCR counter  # counter is now 11
```

What are the benefits of BSON over JSON in MongoDB? BSON (Binary JSON) is used in MongoDB instead of JSON because it allows faster parsing, supports additional data types (Date, Decimal128, ObjectId), and provides efficient storage with smaller document sizes and optimized indexing.

Write a MongoDB query that returns the titles of all suspense movies released between 2010 and 2015.

```
db.movies.find(

  { genre: "Suspense", release_year: { $gte: 2010, $lte: 2015 } },

  { title: 1, _id: 0 }

)
```

This query filters movies where the genre is "Suspense" and the release year is between 2010 and 2015, returning only the title field.

What does the $nin operator mean in a Mongo query? The $nin (not in) operator filters out documents that contain specified values. Example:

```
db.users.find({ age: { $nin: [18, 21, 25] } })
```

This retrieves all users whose age is not 18, 21, or 25.