ONE PAGE SUMMARY

IT & Software › NoSQL › Caching Patterns

Introduction to Caching Patterns

Unlocking Performance: Mastering Caching Patterns in NoSQL

DEFINITION

> Caching patterns are strategies used to store frequently accessed data temporarily, improving application performance. They help reduce latency and database load by serving data quickly from memory.

KEY CONCEPTS

Cache Aside

The application checks the cache first; if the data isn't there, it retrieves it from the database and stores it in the cache.

Read-Through Cache

The cache automatically fetches data from the database when a cache miss occurs, simplifying data access for the application.

Write-Through Cache

Data is written to both the cache and the database simultaneously, ensuring consistency between the two.

Write-Behind Cache

Data is written to the cache first, and then asynchronously to the database, improving write performance.

Time-Based Expiration

Cached data is automatically invalidated after a specified time period, ensuring that stale data is not served.

EXAMPLES

→ Using Cache Aside for user profile data retrieval.

→ Implementing Read-Through Cache for product details in an e-commerce app.

→ Applying Write-Behind Cache for logging events in a web application.

MEMORY TIPS

★ Remember 'Cache Aside' as 'Check, then Fetch'.

★ Think of 'Write-Through' as 'Write Together'.

★ Associate 'Time-Based Expiration' with 'Time's Up!' for data freshness.

COMMON MISTAKES

✗ Neglecting cache invalidation leading to stale data.

✗ Overusing caching for rarely accessed data.

✗ Failing to monitor cache performance and hit rates.

QUICK RECAP

Caching patterns enhance application performance by managing data access efficiently. Understanding different caching strategies like Cache Aside and Write-Through is crucial for optimizing NoSQL databases.