DynamoDB

Amazon DynamoDB

- Banco de dados NoSQL gerenciado, altamente escalável
- Armazena dados em formato de chave-valor ou documentos
- Projetado para latência de milissegundos em qualquer escala

Componentes do DynamoDB

- Tabela: estrutura de armazenamento primária
- Chave primária: única obrigatória (simples ou composta)
- Itens e atributos: semelhante a linhas e colunas

Capacidade e Throughput no DynamoDB

CARACTERÍSTICA	ON-DEMAND	PROVISIONED
Definição	Escala automaticamente com base no uso	Capacidade de leitura e escrita definida manualmente
Gerenciamento	Zero configuração — ideal para workloads imprevisíveis	Necessita planejamento de throughput e autoscaling opcional
Performance	Escala rápida, custo por request	Mais controle de custo em workloads estáveis
Burst Capacity	Implícito (responde a picos automaticamente)	Acumula créditos de capacidade para lidar com picos curtos
Adaptive Capacity	Ativo por padrão — otimiza partições automaticamente	Ativo por padrão — redistribui throughput por chave quente
Quando Usar	• Tráfego imprevisível • Workloads esporádicos ou novos • Apps serverless e APIs públicas	• Workloads previsíveis e controlados • Sistemas com throughput estável • Necessidade de controle granular de custo

Índices no DynamoDB

- GSI (Global Secondary Index): consulta por outras chaves
- LSI (Local Secondary Index): mesma chave primária, diferentes atributos de ordenação
- Índices não replicam todo o item — somente atributos escolhidos

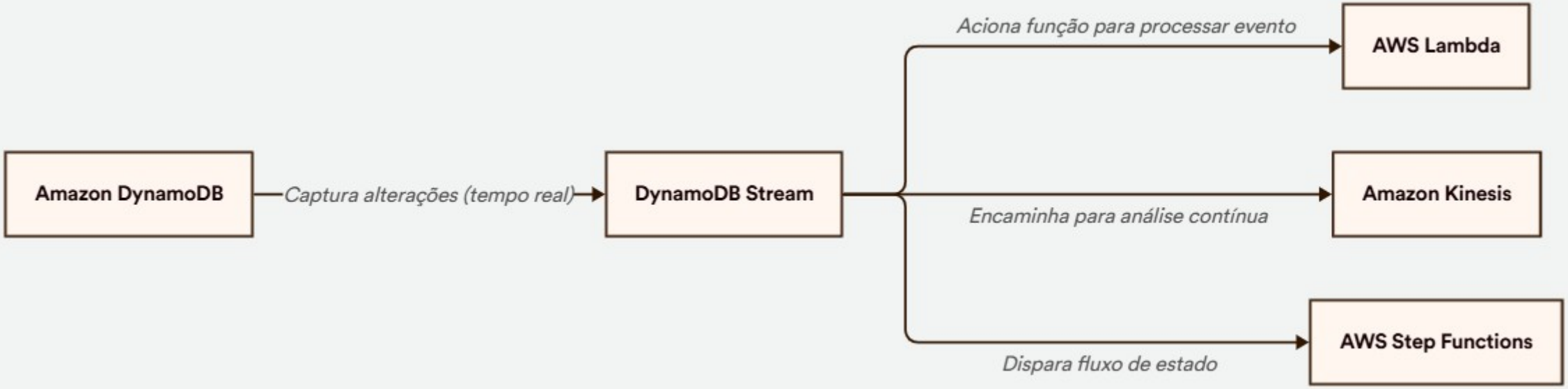
CARACTERÍSTICA	GSI (GLOBAL SECONDARY INDEX)	LSI (LOCAL SECONDARY INDEX)
Chave de Partição	Pode ser diferente da tabela base	Mesma chave de partição da tabela
Chave de Ordenação	Opcional e customizável	Obrigatoriamente diferente da usada na tabela base
Criação	Pode ser adicionada a qualquer momento	Deve ser definida no momento da criação da tabela
Capacidade	Tem sua própria capacidade provisionada (ou on-demand)	Compartilha a capacidade da tabela base
Escopo	Global – pode acessar qualquer item da tabela	Local – limitado à partição original
Destaques na Prova	• Índice mais flexível • Independente da chave primária • Usado para consultas amplas	• Precisa ser criado junto com a tabela • Usado para ordenar itens dentro da mesma partição • Mais limitado que GSI

DynamoDB – Configurações Especiais e Recursos Avançados

TIPO OU RECURSO	DESCRIÇÃO
Com Streams habilitado	Captura mudanças na tabela (inserções, atualizações, exclusões) via **DynamoDB Streams**.
Com TTL (Time to Live)	Exclui automaticamente itens expirados com base em um atributo de tempo (Unix timestamp).
Com Indexes	Suporte a **GSI** (Global Secondary Index) e **LSI** (Local Secondary Index) para consultas alternativas.
Com DAX (cache acoplado)	Integração com **DynamoDB Accelerator (DAX)** para leituras em microssegundos com cache gerenciado.
Com Encryption at Rest	Criptografia ativada por padrão usando **AWS KMS**, sem impacto no desempenho.
Com Partição	Tabelas são automaticamente particionadas com base na chave de partição e no volume de dados/tráfego.

Streams e Integrações

- DynamoDB Streams capturam alterações em tempo real
- Pode acionar AWS Lambda para processar eventos
- Integrações comuns: Lambda, Kinesis, Step Functions



Amazon DynamoDB vs. Amazon DocumentDB

CARACTERÍSTICA	AMAZON DYNAMODB	AMAZON DOCUMENTDB
Modelo de Dados	Chave-valor e documento (NoSQL)	Orientado a documentos JSON (NoSQL)
Compatibilidade	Nativo da AWS	Compatível com MongoDB (API)
Escalabilidade	Altamente escalável com auto scaling e throughput configurável	Escalabilidade limitada horizontalmente; instâncias fixas
Desempenho	Latência de milissegundos com leitura e escrita em alta escala	Boa performance para cargas moderadas; otimizado para leitura
Gerenciamento	Totalmente gerenciado e serverless	Gerenciado com instâncias específicas e configuração de cluster
Casos de Uso	• Aplicações serverless • Gaming, IoT, mobile • Sessões, caching, logs	• Migração de bancos MongoDB • Catálogos com estrutura flexível • Apps que exigem queries ricas
Destaque na Prova	• Escalabilidade extrema • Índices secundários, TTL, DAX • CAP: Prioriza disponibilidade e particionamento (AP)	• Alternativa gerenciada ao MongoDB • Requisições via driver Mongo • CAP: Prioriza consistência e disponibilidade (CP)

Boas Práticas e Dicas de Prova

- Capacidade sob demanda evita erros de throughput por subprovisionamento
- Use GSIs para flexibilizar as consultas – mas atente ao custo
- LSIs são limitados a 5 por tabela e só funcionam com chave composta
- Streams + Lambda são comuns em arquitetura orientada a eventos
- DynamoDB ≠ DocumentDB: entenda quando usar cada um!