# 📖Apêndice A3 – Comparativo CAP dos Bancos de Dados

Este apêndice resume como cada banco de dados abordado no livro se comporta em relação ao **Teorema CAP** (Consistência, Disponibilidade e Tolerância a Partição).

---

## 📊Tabela Comparativa CAP

| Banco de Dados | Categoria | Classificação CAP | Observações |
|---|---|---|---|
| **PostgreSQL** | Relacional | **CA** (Consistência + Disponibilidade) | Em modo standalone oferece forte consistência e alta disponibilidade local. Não tolera partições nativamente; para isso, precisa de soluções externas (Patroni, Citus, etc.). |
| **HBase** | Colunar | **CP** (Consistência + Partição) | Baseado no HDFS. Prefere parar a aceitar requisições em caso de falhas de rede a comprometer consistência. |
| **MongoDB** | Documentos | **CP** (Consistência + Partição) | Replica sets garantem consistência forte no primário. Secundários oferecem **consistência eventual**. Durante eleições, escritas ficam indisponíveis até um novo primário ser eleito. Em sharding, cada shard mantém o mesmo comportamento. |
| **CouchDB** | Documentos | **AP** (Disponibilidade + Partição) | Projetado para replicação distribuída e tolerância a falhas de rede. Garante disponibilidade e tolerância a partição, mas com **consistência eventual**. |
| **Neo4j** | Grafos | **CA**, podendo atuar como **CP** em cluster | Em instância única, é CA. Em clusters com causal clustering (Raft), prioriza consistência e partição, sacrificando disponibilidade em falhas de líder. |
| **DynamoDB** | Chave-valor/ Documento | **AP** (Disponibilidade + Partição) | Inspirado no Dynamo original da Amazon. Garante disponibilidade e tolerância a partições, oferecendo consistência eventual por padrão. Permite consistência forte em leitura, mas com custo de latência. |

| Banco de Dados | Categoria | Classificação CAP | Observações |
|---|---|---|---|
| **Redis** | In-memory | **AP** (quando em cluster) | Em modo standalone → CA (rápido e consistente localmente). Em cluster, prioriza disponibilidade e partição, sacrificando consistência global (replicação assíncrona). Pode ser configurado via `WAIT` para maior consistência. |

---

## 🔗Conclusão

- Bancos **CP** → PostgreSQL, HBase, MongoDB, Neo4j (em cluster).
- Bancos **AP** → CouchDB, DynamoDB, Redis (em cluster).
- A escolha depende do **trade-off desejado**: consistência forte x alta disponibilidade em rede distribuída.

📌Não existe banco “perfeito”: existe o **banco certo para o problema certo**.