### **1. Forma de armazenamento de arquivos**

* **Colunar, em arquivo local**DuckDB adota um formato colunar, onde dados de cada coluna são armazenados juntos e comprimidos, reduzindo significativamente o I/O para consultas que tocam apenas algumas colunas de tabelas muito largas, já que é feita uma leitura seletiva de de colunas, reduzindo significativamente o volume de dados transferidos.
* **Arquivos compactos e multi-plataforma**Os dados ficam armazenados em um único arquivo “.db” no sistema de arquivos local, facilitando o versionamento, a portabilidade entre ambientes e o compartilhamento entre membros da equipe.

### **2. Linguagem e processamento de consultas**

* **SQL padrão**Oferece suporte completo a SQL, incluindo agregações, janelas, CTEs recursivas (forma de realizar subconsultas e percorrer estruturas hierárquicas dentro do bando) e funções analíticas, compatível com a maioria dos scripts e bibliotecas de análise de dados.
* **Integração embutida em Python/R**DuckDB roda embutido no processo Python ou R, expondo uma API minimalista (“import duckdb” em Python ou via pacote R). Isso elimina a sobrecarga de comunicação cliente-servidor e permite passar diretamente DataFrames ou tabelas em memória para dentro do DuckDB, agilizando o ciclo de análise exploratória em notebooks Jupyter/RMarkdown.
* **Execução vetorizada e self-contained**O motor de consulta é vetorizado: processa blocos de valores (vetores) por vez, aproveitando melhor caches de CPU e SIMD (Single Instruction, Multiple Data. refere-se a um paradigma de processamento em que uma só instrução de CPU é aplicada simultaneamente a vários elementos de dados), o que acelera leituras e agregações sobre grandes volumes.

### **3. Processamento e controle de transações**

* **Modelo MVCC simplificado**DuckDB implementa controle de concorrência multiversão (MVCC), garantindo que leitores nunca bloqueiem outros leitores ou escritores. Embora o cenário exija baixa frequência de escrita, essa camada trivial de isolamento (“snapshot reads” - estado do banco de dados em um determinado momento) assegura que leituras longas para análise não impactem eventuais cargas de dados pontuais.
* **Transações ACID leves**Suporta transações completas (BEGIN/COMMIT/ROLLBACK), mas sem o overhead de um banco cliente-servidor. Para o perfil de baixa escrita do cenário, a performance de commit é adequada, e falhas no meio de uma transação não deixam o banco em estado inconsistente.

### **4. Mecanismos de recuperação**

* **WAL e checkpoints**DuckDB mantém um WAL (Write-Ahead Log - escreve, em um arquivo de log sequencial, todas as alterações (como inserções e atualizações) que mudarão os dados antes que ela aconteça e seja aplicada fisicamente) para gravações e, periodicamente, compacta os dados em checkpoints. Em caso de falha de processo ou energia, ao reiniciar, aplica automaticamente o WAL pendente para restaurar um estado consistente.
* **Banco self-contained sem dependências externas**Por ser um único arquivo, basta copiar o arquivo “.db” e o WAL associado para backup ou restauração. Não há necessidade de gerenciar instâncias de servidores ou arquivos de log separados.

### **5. Segurança**

* **Autenticação e criptografia de arquivos**Embora DuckDB, nativamente, não tenha controle de usuários nem autenticação por cliente-servidor, é possível:  
  1. **Proteger acesso ao arquivo** via permissões de sistema operacional (chmod/chown em Linux);
  2. **Usar volumes criptografados** (ex: LUKS (Linux), BitLocker (Windows)) para armazenar o arquivo .db, fazendo com que somente quem possuir a senha seja capaz de abrir e ler os dados dos arquivos corretamente.

**Conclusão** O DuckDB alinha-se perfeitamente ao Cenário A, pois:

* Armazena colunarmente grandes volumes de dados históricos de forma compacta e performática.
* Proporciona integração nativa com Python/R em notebooks, acelerando workflows analíticos.
* Garante ACID básico e recuperação automática sem precisar de infraestrutura de servidor.
* Permite segurança via controles de arquivo e criptografia de disco.

Por esses motivos, o DuckDB é a escolha recomendada para suportar leituras intensivas, agregações em larga escala e fácil adoção por analistas de dados, atendendo a todos os requisitos técnicos do cenário.