Relatório: Implementação de Sistema de Indexação e Pesquisa de Dados

Gabriel Iza Fauth
Melchior Boaretto Neto

8 de dezembro de 2025

Resumo

Este relatório descreve a implementação de um sistema de gerenciamento de dados utilizando indexação via Árvore B em memória principal e persistência em arquivo binário. O sistema foi desenvolvido para gerenciar um conjunto de dados de cancelamento de clientes (Churn), permitindo operações de inserção, busca eficiente por chave primária e filtragem sequencial por atributos secundários.

1 Dos dados utilizados

O conjunto de dados selecionado para este projeto é o *Telco Customer Churn*.

1.1 Origem

Os dados foram obtidos originalmente da plataforma Kaggle e representam o perfil de consumo e status de contrato de clientes. O arquivo encontra-se em formato CSV.

1.2 Esquema de Dados Adotado

Foram selecionados os seguintes atributos, mapeados para uma estrutura de tamanho fixo de 83 bytes (antes de eventual padding):

- **ID do Cliente (15 bytes):** Identificador único.
- **Gênero (10 bytes):** Male ou Female.
- **Cancelou (5 bytes):** Indicador de *Churn* (Yes/No).
- **Contrato (40 bytes):** Tipo de contrato (Mensal, Anual, etc.).
- **Valor Mensal (float - 4 bytes):** Custo mensal.
- **Meses (int - 4 bytes):** Tempo de permanência.
- **Idade (int - 4 bytes):** Idade do cliente.

2 Índices e Estruturas de Dados

Toda a eficiência do projeto baseia-se na implementação de uma estrutura de dados hierárquica para indexação.

2.1 Estrutura de Indexação: Árvore B

Foi implementada uma **Árvore B** com grau mínimo $t = 3$. Esta escolha justifica-se pelo fato das operações serem feitas assintoticamente em $O(\log_t n)$. Além disso, diferentemente da classe das *ABPs*, a altura da árvore se mantém extremamente curta.

- **Atributo Indexado (Chave):** O campo `id_cliente` é a chave primária de indexação.
- **Implementação:** A árvore reside em memória principal durante a execução. Cada nodo pode conter entre $t - 1$ e $2t - 1$ chaves. O split de nodos ocorre na descida durante a inserção.

2.2 Persistência de Dados

A persistência utiliza manipulação direta de arquivos binários.

- O sistema converte os objetos da classe `Cliente` para *bytes* utilizando a biblioteca `struct` do Python (Serialização).
- Isso permite acesso aleatório e leitura sequencial rápida, eliminando o *overhead* de *parsing* de texto exigido por arquivos CSV a cada execução.

3 Implementação e Interface

3.1 Tecnologias Utilizadas

- **Linguagem:** Python.
- **Repositório:** `https://github.com/GFauth/Churn-Viewer`

3.2 Interface do Usuário

A interação ocorre via Interface de Linha de Comando (CLI). O sistema apresenta um menu interativo com estas funcionalidades:

1. **Buscar Cliente por ID:** Utiliza o algoritmo de busca da Árvore B.
2. **Filtragem Avançada:** Permite filtrar registros por atributos não indexados (Gênero, Contrato, Valores, etc...) percorrendo a árvore.
3. **Estatísticas:** Cálculo de médias de valores mensais segregados por status de cancelamento.
4. **Gerenciamento de Arquivo:** Opção para resetar a base binária a partir do CSV original.

3.3 Performance de Busca por ID vs Filtragem

A busca por ID comportou-se conforme a expectativa teórica, retornando resultados instantaneamente.

As operações de filtragem (e.g. buscar todos os clientes do sexo masculino) tem desempenho inferior. Como apenas o ID é indexado, filtros quaisquer exigem cobrir toda a árvore, resultando em complexidade linear. Percebe-se, pois, que se a base de dados fosse da ordem de 100 ou 1000 vezes mais massiva, far-se-ia necessário o uso de *Árvores B+* e/ou arquivos invertidos.

4 Conclusão

O projeto cumpriu os requisitos de manipulação de arquivos e implementação de estruturas de dados. Para a base de dados do projeto, a utilização da Árvore B foi eficaz para a pesquisa de registros por chave primária.