Projekt z kursu Struktur Baz Danych - Indeksowa Organizacja Plików Szymon Groszkowski 193141

December 9, 2024

1 Cel projektu

Celem projektu jest stworzenie systemu zarządzania plikiem danych, który umożliwia szybki i efektywny dostęp do rekordów dzięki zastosowaniu indeksowania opartego na strukturze drzewa. Rozwiązanie ma wspierać podstawowe operacje, takie jak dodawanie, wyszukiwanie, modyfikowanie oraz usuwanie rekordów, a także umożliwiać przeglądanie zawartości danych w kolejności kluczy.

Dodatkowym wyzwaniem projektu jest optymalizacja operacji dyskowych poprzez zastosowanie mechanizmów buforowania.

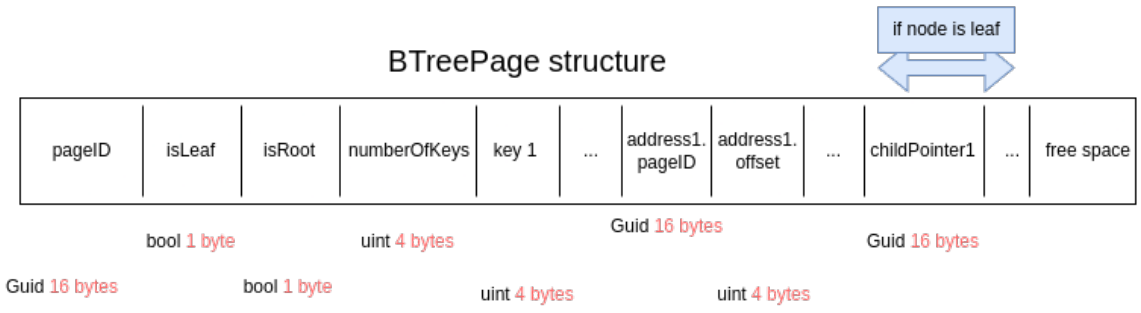
2 Wybrana metoda implementacji

Do realizacji projektu przyjęto implementację indeksowania opartą na strukturze **B-drzewa**.

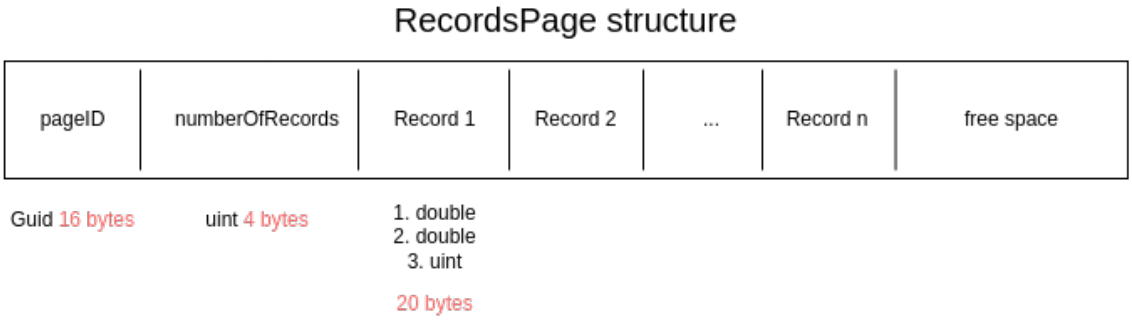
2.1 Opis implementacji

W implementacji założono podział na dwa typy stron:

- **Strony węzłów drzewa** (*BTreeNodePage*) – przechowujące klucze oraz wskaźniki do dzieci i adresów rekordów.



- **Strony danych** (*RecordsPage*) – zawierające zapisane rekordy z ich wartościami i kluczami.



2.2 Usprawnienia

Aby zwiększyć efektywność operacji, zaimplementowano następujące usprawnienia:

- **Przechowywanie korzenia drzewa w pamięci operacyjnej:** Pozwala to na uniknięcie częstych odczytów korzenia z dysku.
- **Mechanizm cache'owania stron:** Wprowadzono strategię LRU (Least Recently Used), która pozwala na przechowywanie często używanych stron w pamięci RAM, co znacznie redukuje liczbę operacji wejścia/wyjścia.
- **Mechanizm ponownego wykorzystania zwalnianego miejsca:** Przy usuwaniu rekordów, zwolnione miejsce jest zapisywane na stosie wolnych obszarów. Pozwala to na późniejsze wykorzystanie tych przestrzeni przy wstawianiu nowych rekordów, co redukuje fragmentację i minimalizuje konieczność alokacji nowych stron.

3 Plik konfiguracyjny

Plik konfiguracyjny służy do definiowania parametrów działania programu oraz ustalania szczegółowych ustawień dla generowania danych, buforowania pamięci RAM oraz struktury drzewa indeksowego. Parametry konfiguracyjne są zapisane w formacie JSON, co ułatwia ich edycję oraz odczyt przez program. Poniżej przedstawiono kluczowe pola pliku konfiguracyjnego:

- `PageSizeInNumberOfRecords` – liczba rekordów mieszczących się na jednej stronie.
- `RecordSizeInBytes` – rozmiar pojedynczego rekordu w bajtach.
- `RAMSizeInNumberOfPages` – liczba stron, które mogą być przechowywane w pamięci RAM (cache).

- `DataSource` – źródło danych, które może przyjmować jedną z wartości: `GenerateRandomly`, `ProvideManually`, `LoadFromFile`.
- `FilePathToRecords` – ścieżka do pliku zawierającego rekordy (wymagane dla `LoadFromFile`).
- `NumberOfRecordsToGenerate` – liczba rekordów generowanych losowo (dotyczy, gdy `DataSource` ustawiony na `GenerateRandomly`).
- `FilePathToInstructions` – ścieżka do pliku z instrukcjami dla programu.
- `TreeDegree` – stopień drzewa indeksowego (`d`).
- `EnableCaching` – flaga włączająca mechanizm buforowania stron w RAM.
- `EnableNodeCompensation` – flaga włączająca mechanizm kompensacji.
- `EnableSpaceSaving` – flaga włączająca mechanizm oszczędzania miejsca.

Przykładowa zawartość pliku konfiguracyjnego:

```
{
  "Settings": {
    "PageSizeInNumberOfRecords": 5,
    "RecordSizeInBytes": 20,
    "RAMSizeInNumberOfPages": 11,
    "NumberOfRecordsToGenerate": 16,
    "DataSource": "GenerateRandomly",
    "FilePathToRecords": null,
    "FilePathToInstructions": "Data/Instructions.txt",
    "TreeDegree": 2,
    "EnableCaching": true
    "EnableNodeCompensation" : true,
    "EnableSpaceSaving" : true
  }
}
```

4 Plik z instrukcjami

Plik z instrukcjami definiuje operacje, jakie mają zostać wykonane na danych i drzewie indeksowym. Każda linia w pliku odpowiada jednej instrukcji i składa się z komendy oraz jej parametrów. Obsługiwane komendy to:

- `insert` – dodanie rekordu. Wymaga podania współrzędnych `X`, `Y` oraz klucza `Key`.
- `find` – wyszukanie rekordu po kluczu. Wymaga jednego parametru: `Key`.
- `delete` – usunięcie rekordu. Wymaga podania klucza `Key`.

- `update` – aktualizacja rekordu. Wymaga klucza `Key` i nowych wartości `X, Y` Można także podać wartość nowego klucza.
- `print` – wyświetlenie danych. Parametry:
  - `records` – wyświetlenie wszystkich rekordów w kolejności kluczy.
  - `btree` – wyświetlenie struktury B-drzewa.

Przykładowy format pliku instrukcji:

```
insert 15.0 30.0 8
find 5
delete 8
update 5 25.0 3.43 6
print records
print btree
```

4.1 Obsługa instrukcji

Instrukcje są wczytywane i parsowane przez dedykowaną metodę `ParseCommand`, która dzieli linie na komendę oraz parametry. W zależności od typu komendy, program wykonuje odpowiednią operację na danych oraz aktualizuje strukturę B-drzewa. Każda operacja jest logowana, a statystyki I/O są prezentowane po jej wykonaniu.

5 Eksperymenty i wnioski

Eksperymenty zostały przeprowadzone przy użyciu zestawu operacji, obejmującego wstawianie, wyszukiwanie oraz usuwanie rekordów. Wyniki każdego eksperymentu zostały ocenione pod kątem:

- liczby operacji I/O (odczyt/zapis stron dyskowych),
- głębokości drzewa po wykonaniu operacji,

5.1 Wpływ stopnia drzewa

Przetestowano różne wartości stopnia drzewa ($d = 2, 3, 4, 5, 6, 7$) dla zestawu operacji w celu oceny wpływu tego parametru na wydajność systemu. Przyjęto następujące założenia oraz przebieg testów:

5.1.1 Założenia i przyjęte parametry

Dla testów wpływu stopnia drzewa inne parametry systemu ustalono jako stałe:

- **Liczba rekordów:** $10,000$ rekordów testowych, co pozwala na wyraźną obserwację różnic wydajności przy różnych stopniach drzewa.

- **Rozmiar strony dyskowej:** 4 KB, wartość typowa w systemach plików. Odpowiada to ok. 200 rekordom na stronę.
- **Rozmiar pamięci RAM (cache):** Miejsce na 10 stron w pamięci podręcznej, co wymusza konieczność operacji I/O przy większych zbiorach danych.
- **Tryby optymalizacji:** Wykonano testy w dwóch trybach:
  - Z włączonymi optymalizacjami (cache oraz ponowne użycie zwolnionych miejsc na stronach).
  - Bez optymalizacji (standardowa implementacja bez mechanizmu cache).

5.1.2 Przebieg testów

Dla każdej wartości stopnia drzewa ($d = 2, 3, 4, 5, 6, 7$) wykonano następujące operacje:

1. **Wstawianie rekordów:** Wstawiono wszystkie $10,000$ rekordów do drzewa w losowej kolejności, mierząc liczbę średnią operacji I/O oraz głębokość drzewa z 3 prób.
2. **Usuwanie rekordów:** Po dodaniu $10,000$ rekordów usunięto losowe 25% rekordów $(2,500)$ z drzewa, rejestrując średnią liczbę operacji I/O z 3 prób.
3. **Wyszukiwanie rekordów:** Po dodaniu $10,000$ rekordów przeprowadzono $1,000$ losowych wyszukiwań w drzewie, mierząc średnią liczbę operacji I/O z 3 prób.

5.1.3 Wpływ stopnia drzewa na liczbę operacji I/O i wysokość drzewa

Table 1: Wpływ stopnia drzewa na operacje I/O i wysokość drzewa podczas wstawiania 10 tysięcy losowo wstawianych rekordów.

Stopień drzewa	Operacje I/O (bez)	Operacje I/O (z)	Wysokość drzewa
2	119,178 / 42,275	64,884 / 42,275	6
3	95,838 / 38,819	52,518 / 38,819	4.33
4	87,938 / 36,910	47,430 / 36,910	4
5	77,132 / 35,801	42,089 / 35,801	4
6	73,164 / 34,805	38,876 / 34,805	3
7	71,464 / 34,667	38,250 / 34,667	3

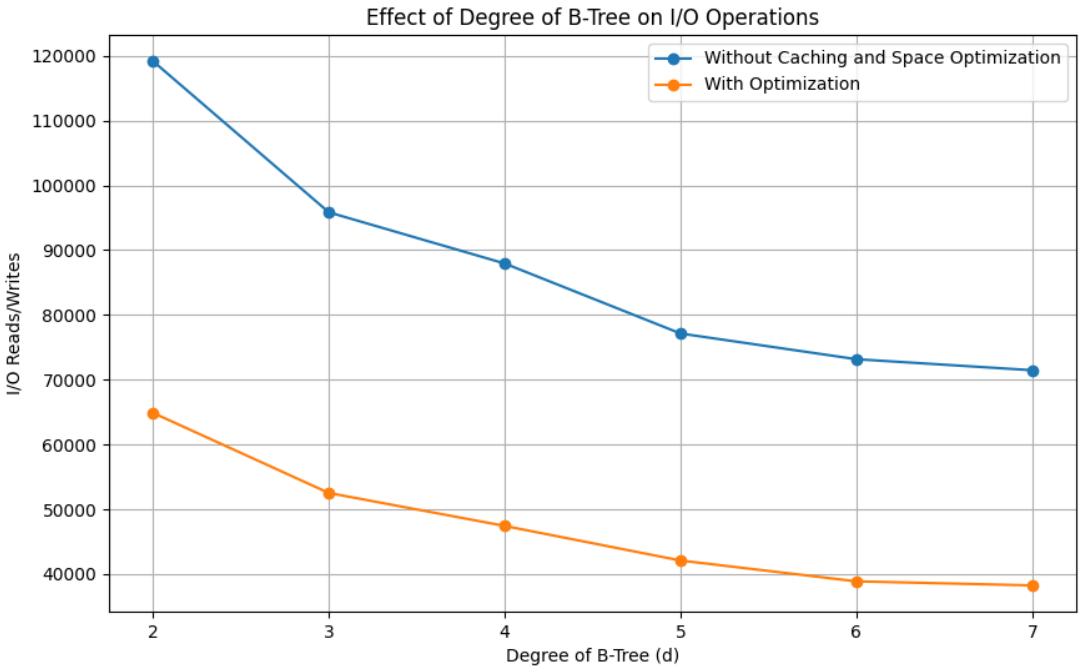


Table 2: Wpływ stopnia drzewa na operacje I/O podczas usuwania 25% rekordów z 10 tysięcy wstawionych rekordów. Wyniki uwzględniają wyłącznie operacje usuwania (operacje I/O podczas wstawiania są odjęte).

Stopień drzewa	**Operacje I/O**	**Operacje I/O (z optymalizacjami)**
2	18,789 / 5,660	16,213 / 5,660
3	15,050 / 5,014	12,316 / 5,014
4	12,785 / 5,170	9,822 / 5,170
5	11,859 / 5,009	7,799 / 5,009
6	9,606 / 4,807	8,007 / 4,807
7	9,527 / 4,823	7,911 / 4,823

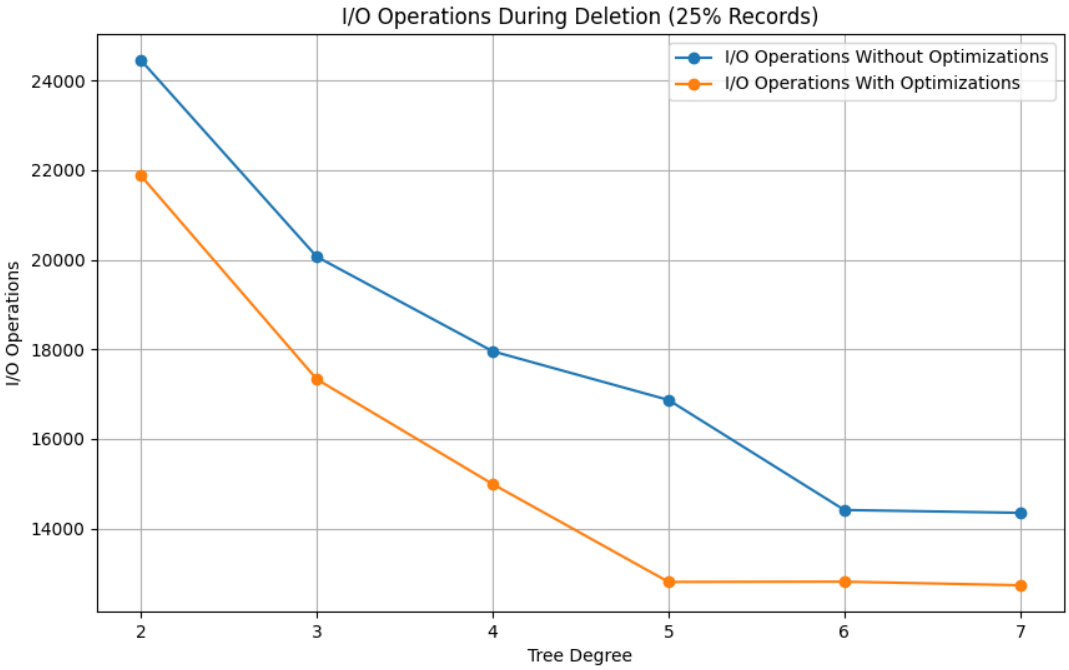
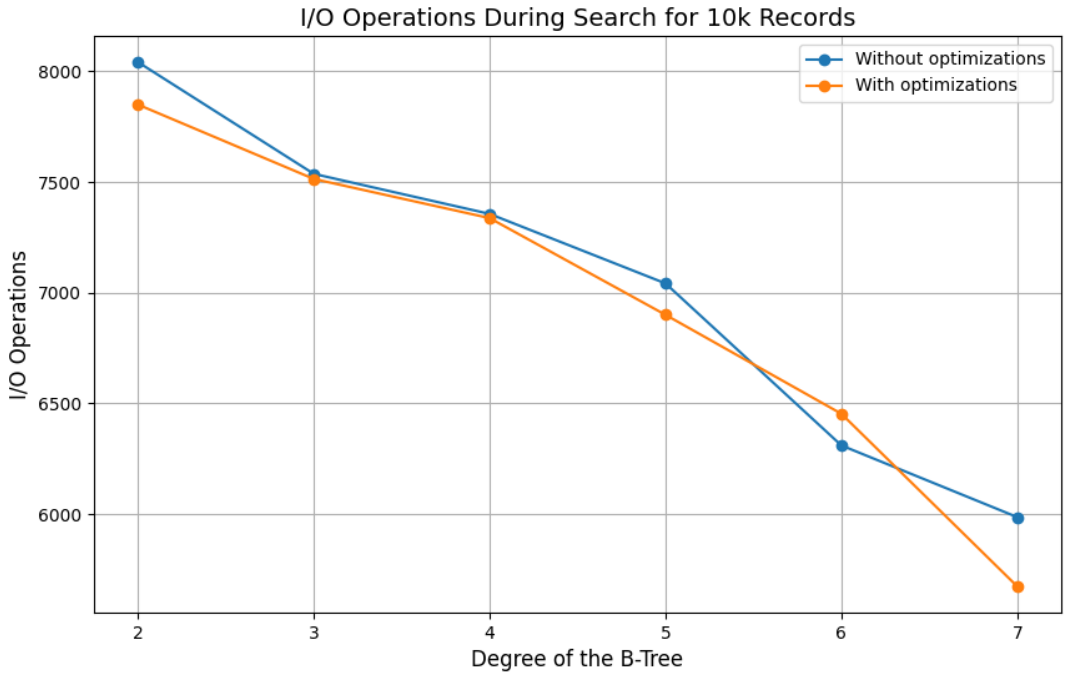


Table 3: Wpływ stopnia drzewa na operacje I/O podczas wyszukiwania 10 tysięcy rekordów. Wyniki uwzględniają wyłącznie operacje wyszukiwania (operacje I/O podczas wstawiania są odjęte).

Stopień drzewa	**Operacje I/O (bez optymalizacji)**	**Operacje I/O (z optymalizacjami)**
2	8,042	7,850
3	7,537	7,514
4	7,356	7,337
5	7,042	6,900
6	6,310	6,453
7	5,986	5,673



6 Podsumowanie eksperymentów

Przeprowadzone eksperymenty miały na celu zbadanie wpływu parametrów implementacyjnych, takich jak stopień drzewa B oraz zastosowanie mechanizmów optymalizacyjnych, na wydajność operacji wejścia/wyjścia (I/O) w trzech scenariuszach: wstawiania, usuwania oraz wyszukiwania rekordów.

6.1 Wpływ stopnia B-drzewa

1. Wraz ze wzrostem stopnia B-drzewa zaobserwowano systematyczne zmniejszenie się liczby operacji I/O we wszystkich testowanych scenariuszach. Wyższy stopień drzewa prowadzi do

zmniejszenia wysokości drzewa, co ogranicza liczbę odczytów i zapisów podczas przeszukiwania węzłów.

2. Największy wpływ na redukcję operacji I/O odnotowano w scenariuszu wyszukiwania oraz usuwania, gdzie wyższy stopień drzewa pozwolił na bardziej efektywne operacje.

6.2 Wpływ mechanizmów optymalizacyjnych

1. Mechanizmy takie jak cache'owanie stron znacząco zmniejszyły liczbę operacji wejścia/wyjścia.

2. Dla niższych stopni drzewa (np. $d = 2$) różnica między wynikami z optymalizacjami a bez nich była największa. Wraz ze wzrostem stopnia drzewa różnice te maleją, co sugeruje, że optymalizacje są bardziej efektywne przy większej liczbie poziomów w drzewie.

6.3 Wnioski końcowe

1. Optymalny stopień drzewa B zależy od scenariusza użytkowania. Wyższe stopnie są bardziej wydajne w operacjach wyszukiwania i usuwania, podczas gdy niższe stopnie mogą być korzystniejsze w aplikacjach wymagających częstych operacji wstawiania.

2. Mechanizmy optymalizacyjne są kluczowe dla zmniejszenia liczby operacji I/O, szczególnie w systemach z dużą liczbą rekordów oraz częstym dostępem do danych.

3. W praktyce rekomendowane jest użycie wyższego stopnia drzewa (np. $d = 6$ lub $d = 7$) w połączeniu z mechanizmami optymalizacyjnymi, co zapewnia najlepszy kompromis pomiędzy wydajnością operacji a zarządzaniem przestrzenią.