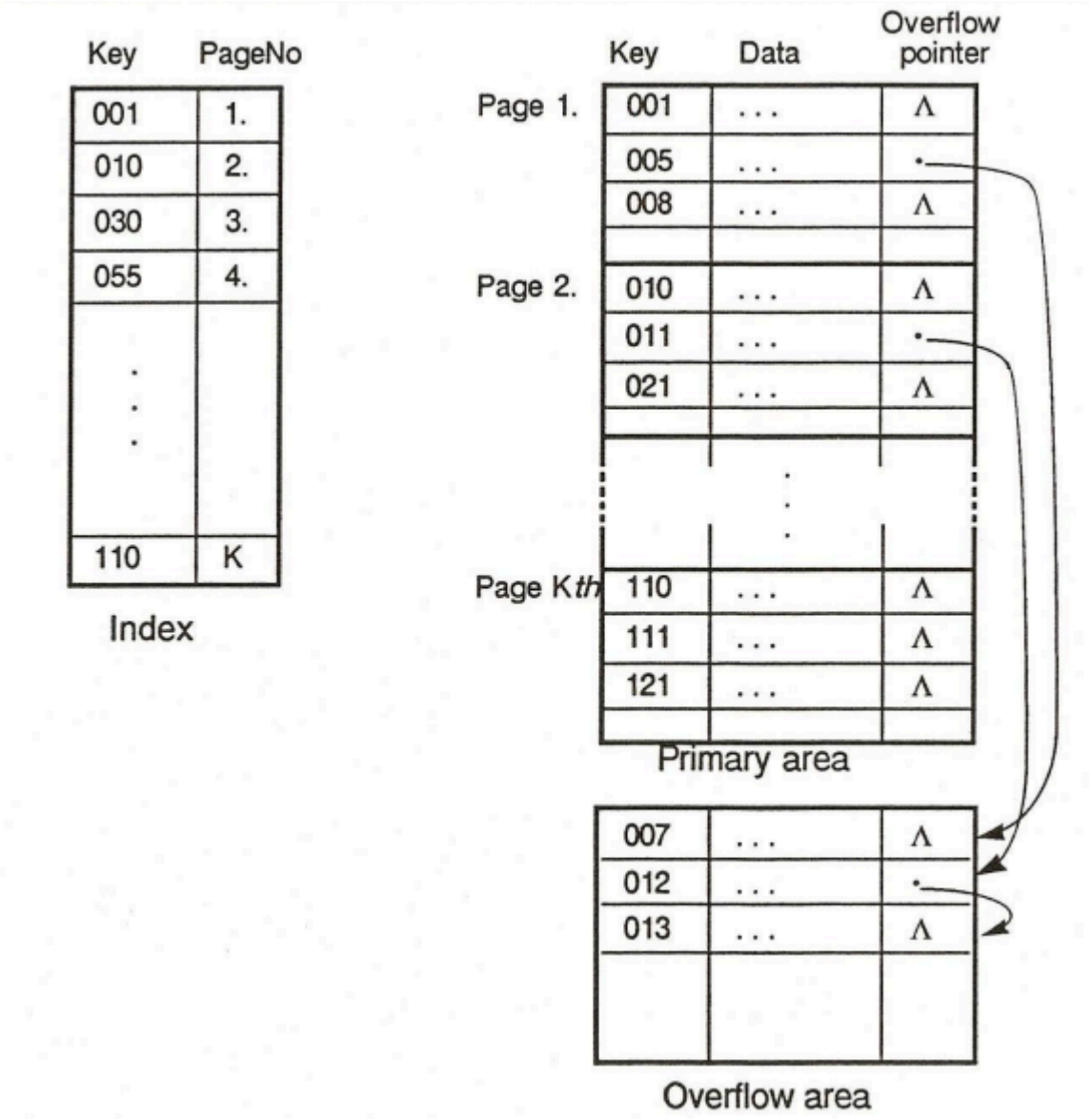
Indeksowo-sekwencyjna organizacja pliku

Antoni Czuba 201096 Sem. V Grupa 5A

I. Opis algorytmu

W projekcie zastosowałem organizacje indeksowo -sekwencyjną która działa w następujący sposób.



System opiera się na architekturze indeksowo-sekwencyjnej, podzielonej na trzy logiczne obszary:

- **Obszar Stron Głównych (Main Pages Area):** Przestrzeń podstawowa, przechowująca strony wypełnione posortowanymi rekordami danych.
- **Obszar Nadmiarowy (Overflow Area):** Przestrzeń pomocnicza, do której trafiają rekordy, dla których zabrakło miejsca w strukturze stron głównych (obsługa kolizji i braku miejsca).
- **Obszar Indeksu (Index Area):** Struktura nawigacyjna przechowująca pary wartości: klucz oraz offset (wskaźnik), wskazujące na pierwszy rekord każdej strony w Obszarze Głównym, co umożliwia szybką lokalizację odpowiedniej strony.

W ramach przyjętej architektury realizowane są cztery fundamentalne operacje:

1. Dodawanie rekordu

Proces przebiega dwuetapowo. Najpierw, na podstawie indeksu, lokalizowana jest strona, której klucz początkowy jest mniejszy od klucza dodawanego rekordu. Następnie podejmowana jest próba wstawienia:

- **Wstawienie bezpośrednie:** Rekord trafia w pierwsze wolne miejsce na stronie, zachowując porządek sortowania (za największym rekordem o kluczu mniejszym od dodawanego).
- **Obsługa przepełnienia:** Jeżeli na stronie brakuje miejsca lub nowy rekord powinien logicznie znaleźć się pomiędzy rekordem n a n+1 (gdzie fizycznie nie ma przestrzeni), rekord ten trafia do **Obszaru Nadmiarowego (Overflow Area)**. Tworzona jest wówczas struktura listy wiązanej (kolejki priorytetowej), której początkiem jest wskaźnik wychodzący od rekordu poprzedzającego n-tego na stronie głównej.

2. Reorganizacja struktury

Operacja konserwacji bazy danych, polegająca na całkowitej przebudowie plików. Proces ten:

- Scal rekordy z Obszaru Głównego i Nadmiarowego.
- Tworzy strony od nowa, eliminując całkowicie rekordy z Obszaru Nadmiarowego.
- Gwarantuje, że liczba rekordów na nowo utworzonych stronach nie przekracza wartości **B*alfa** (gdzie B to pojemność strony, a **alfa** to współczynnik wypełnienia).

3. Usuwanie rekordu (Deletion)

Operacja logiczna, polegająca na oflagowaniu rekordu jako „usunięty”. Rekord fizycznie pozostaje w pliku do momentu najbliższej reorganizacji, podczas której jest pomijany i trwale usuwany ze struktury.

4. Aktualizacja rekordu (Update)

Proces modyfikacji danych rekordu.

II. Implementacja

Rekordy miały taką strukturę

Klucz (int) + Dane (char[9]) + pointer(int) + flagaUsunięcia(bool)

Pointer to offset kolejnego rekordu w overflow area

Jako zbiór danych posłużyły mi tablice rejestracyjne, przechowywałem je w postaci Stringów, o długości 8. Dane jakie generowałem miały następującą postać

XXXYYYYY

Wyróżnik miejsca oznaczony 2-3 znakami **X** składa się z samych liter alfabetu łacińskiego, w przypadku wyróżnika o długości 2 na końcu rekordu dodawana była spacja aby zachować stałą długość rekordu.

Natomiast wyróżnik pojazdu oznaczony 5 znakami **Y** składa głównie z cyfr arabskich, ale mogą pojawić się tam również litery alfabetu łacińskiego..

Dane zapisywałem binarnie aby jak najłatwiej operowało mi się na bajtach.

III. Eksperyment

Enter the key for the new record (integer): 17
Record added to Main file.
Disk IO Operations -> Reads: 3, Writes: 3

Enter the key for the new record (integer): 18
Record added to Main file.
Disk IO Operations -> Reads: 0, Writes: 1

Enter the key for the new record (integer): 20
Record added to Overflow file.
Disk IO Operations -> Reads: 0, Writes: 2

W wyżej pokazanych danych widać jak zmienia się ilość operacji w zależności od poprzednich operacji, pierwszy rekord potrzebował znacznie więcej operacji niż kolejne ponieważ w pamięci operacyjnej nie znajdowała się strona na której zapisywano dane, gdy już ją wybrano kolejne operacje były znacznie bardziej efektywne.

[13] Key: 31 | Data: FMO37816 | Ptr: -1 | Del: false
[14] Key: 50 | Data: LA79288 | Ptr: -1 | Del: false
W głównym sektorze mamy dwa rekordy.

Enter the key for the new record (integer): 32
Record added to Overflow file.
Disk IO Operations -> Reads: 0, Writes: 2

Enter the key for the new record (integer): 33
Record added to Overflow file.
Disk IO Operations -> Reads: 0, Writes: 2

Enter the key for the new record (integer): 34
Record added to Overflow file.
Disk IO Operations -> Reads: 0, Writes: 2

Jak widać operacja dodawania do obszaru nadmiarowego ma stałą liczbę zapisów, wynika to z tego że cały overflow mieści się w jednej stronie która odczytuje program, dzięki temu nie wykonujemy nadmiarowych operacji.

Enter the key for the new record (integer): 41
Record added to Overflow file.
Disk IO Operations -> Reads: 2, Writes: 4

Po dodaniu na tyle dużo rekordów, że w końcu wyszliśmy z zakresu pierwszej strony obszaru nadmiarowego liczba operacji znacznie podskoczyła

```
=== EXPERIMENT RESULTS ===
Operation      | Avg Reads  | Avg Writes
-------------------------------------
Add Record     |       0.88 |       1.79
Modify Data    |       1.08 |       1.15
Delete Record  |       1.00 |       1.12
-------------------------------------
Operation      | Total Reads| Total Writes
-------------------------------------
Reorganization |      36.00 |      52.00
-------------------------------------
Alpha page utill = 12
Blocking factor  = 15
Records before = 120
Records added = 70

=== EXPERIMENT RESULTS ===
Operation      | Avg Reads  | Avg Writes
-------------------------------------
Add Record     |       1.52 |       2.28
Modify Data    |       1.68 |       1.76
Delete Record  |       1.42 |       1.50
-------------------------------------
Operation      | Total Reads| Total Writes
-------------------------------------
Reorganization |      19.00 |      33.00
-------------------------------------
Alpha page utill = 7
Blocking factor  = 8
Records before = 70
Records added = 35

=== EXPERIMENT RESULTS ===
Operation      | Avg Reads  | Avg Writes
-------------------------------------
Add Record     |       2.15 |       2.77
Modify Data    |       2.15 |       2.23
Delete Record  |       2.17 |       2.33
-------------------------------------
Operation      | Total Reads| Total Writes
-------------------------------------
```

Reorganization | 11.00 | 27.00

Alpha page utill = 3
Blocking factor = 4
Records before = 30
Records added = 15
Jak widać wraz ze zmniejszeniem Blocking factor zwiększa się ilość potrzebnych odczytów i zapisów do wykonania operacji dodawania, usuwania i aktualizowania rośnie natomiast odwrotna zależność występuje w przypadku reorganizacji wynika to z większej ilości rekordów w obszarze nadmiarowym, co wymaga dużej ilości odczytów rekordów w różnych miejscach w pliku przez strukturę kolejki priorytetowej.