Implementation of DBMS
Exercise Sheet 6
HIS, WS 2024 / 2025

1. We want to represent physical addresses for a hard disk. For a block address, we need to identify the following entities: the cylinder, the track within a cylinder, and the block within a track. To each of these entities, we allocate one or more bytes to identify it. Our disk has the following properties:

    - 16,384 cylinders
    - 16 tracks in a cylinder
    - 64 blocks in a track

    a) How many bytes do we need for a block address?
    b) We want to construct a record address by adding the position of the byte within a block to the block address of exercise (a). The blocks of the disk consist of 8,192 bytes. How many bytes would we need for the record address?

2. Suppose that blocks can hold either 12 records or 80 key-pointer pairs. We have a data file that is a sequential file and a sparse index on this file. The index has multiple levels up to a level with just one block. Each primary block of the data file has one overflow block. The primary blocks are full, and the overflow blocks are half full. However, records are in no particular order within a primary block and its overflow block. All index blocks are 75% full.

    a) Calculate the total number of blocks needed for a 4,500,000-record file and the index.
    b) Calculate the average number of disk I/Os needed to retrieve a record given its search key by using the index. You may assume that nothing is in memory initially and that the search key is the primary key for the records.

3. In a B+-tree (discussed later in the lecture) of order m, the minimum number of keys in a node can be calculated with the following formulas:

    - Non-leaf node: $\lceil (m+1)/2 \rceil - 1$
    - Leaf node: $\lfloor (m+1)/2 \rfloor$

    What is the minimum number of keys in B+-tree (i) interior nodes and (ii) leaves, when:
    a) m=15
    b) m=20

Implementation of DBMS
Exercise like Sheet 6 HIS, WS 2024 / 2025

---

1) Representing Physical Addresses for a Hard Disk

We want to represent physical addresses for a hard disk. For a block address, we need to identify the following entities: the cylinder, the track within a cylinder, and the block within a track. To each of these entities, we allocate one or more bytes to identify it. Our disk has the following properties:

- 16,384 cylinders
- 16 tracks in a cylinder
- 64 blocks in a track

a) How many bytes do we need for a block address?
b) We want to construct a record address by adding the position of the byte within a block to the block address of exercise a). The blocks of the disk consist of 2048 bytes. How many bytes would we need for the record address?

---

2) Data File with Index and Overflow Blocks

Suppose that blocks can hold either 20 records or 200 key-pointer pairs. We have a data file that is a sequential file and a sparse index on this file. The index has multiple levels up to a level with just one block. Each primary block of the data file has one overflow block. The primary blocks are full, and the overflow blocks are 75% full. However, records are in no particular order within the primary block and its overflow block. All index blocks are 70% full.

a) Calculate the total number of blocks needed for a **2,400,000-record file** and the index.
b) Calculate the average number of disk I/O's needed to retrieve a record given its search key by using the index. You may assume that nothing is in memory initially, and that the search key is the primary key for the records.

---

3) Minimum Keys in a B+-Tree

In a B+-tree (will be discussed later in the lecture) of order n, the minimum number of keys in a node can be calculated with the following formulas:

- Non-leaf node: $\lceil (n+1)/2 \rceil - 1$
- Leaf node: $\lfloor (n+1)/2 \rfloor$

Questions: What is the minimum number of keys in a B+-tree for (i) interior nodes and (ii) leaves, when:
a) n=15
b) n=20

2a) Data File with Index and Overflow Blocks

Suppose that blocks can hold either 25 records or 250 key-pointer pairs. We have a data file that is a sequential file and a sparse index on this file. The index has multiple levels up to a level with just one block. Each primary block of the data file has one overflow block. The primary blocks are full, and the overflow blocks are 80% full. However, records are in no particular order within the primary block and its overflow block. All index blocks are 65% full.

a) Calculate the total number of blocks needed for a **4,800,000-record file** and the index.
b) Calculate the average number of disk I/O's needed to retrieve a record given its search key by using the index. You may assume that nothing is in memory initially, and that the search key is the primary key for the records.

---

3) Minimum Keys in a B+-Tree

In a B+-tree (will be discussed later in the lecture) of order n, the minimum number of keys in a node can be calculated with the following formulas:

- Non-leaf node: $\lceil (n+1)/2 \rceil - 1$
- Leaf node: $\lfloor (n+1)/2 \rfloor$

Questions:
What is the minimum number of keys in a B+-tree for (i) interior nodes and (ii) leaves, when:
a) n=25
b) n=30

sequential file organization, **sparse multilevel indexing**, and **I/O performance analysis**.
Let's go step-by-step through the **concepts** behind both parts (a) and (b).

📘 Concept Breakdown

🧩 1. Sequential File with Overflow Blocks

🔹 Concept:

A **sequential file** stores records ordered by a key (usually the primary key).
When a block becomes full and a new record needs to be inserted, DBMS doesn't rewrite the entire file.
Instead, it creates an **overflow block** linked to the primary block.

So:

- **Primary block** = main block of the file (full)
- **Overflow block** = linked block for new or relocated records

---

🔹 In this question:

- Each block can hold **10 records**
- Each primary block has **one overflow block**
- Overflow blocks are **half full (5 records)**
- Records are **unsorted** inside the primary+overflow pair

Thus, each **primary+overflow pair** effectively holds **10 + 5 = 15 records**.

🧩 2. Sparse Index

🔹 Concept:

A **sparse index** stores **one key-pointer pair per primary block**, not for every record.
Each pointer refers to the first record in a primary block.
→ Much smaller than a dense index.

🔹 Key given details:

- Each index block can hold **100 key-pointer pairs**
- Each index block is **60% full**, so **60 actual entries**
- The index has **multiple levels** (a **multilevel index**):
    - 1st level indexes the data blocks
    - 2nd level indexes the 1st level
    - and so on, until the top level has **only one block**

🧩 3. Calculation of Data File Size

Total records = 3,240,000
Each pair of blocks holds 15 records:

$$\frac{3,240,000}{15} = 216,000 \text{ pairs}$$

Each pair = **2 blocks**, so:

$$216,000 \times 2 = 432,000 \text{ blocks for data file.}$$

🧩 4. Index Structure Calculation

Level 1 (bottom index)

- One entry per primary block → **216,000 entries**
- Each block holds **60 entries**

$$\frac{216,000}{60} = 3,600 \text{ blocks}$$

Level 2

- One entry per block of previous level → **3,600 entries**
- Each block holds **60 entries**

$$\frac{3,600}{60} = 60 \text{ blocks}$$

Level 3

- One entry per block of previous level → **60 entries**
- Each block holds **60 entries**

$$\frac{60}{60} = 1 \text{ block (top level)}$$

✅ **Index total = 3,600 + 60 + 1 = 3,661 blocks**

✅ **Total storage = Data (432,000) + Index (3,661) = 435,661 blocks**

🧩 5. Record Retrieval using the Index

🔹 Concept:

When searching by key:

1. Start from the **root of the index** (top level)
2. Traverse down to the bottom index level
3. Follow pointer to **primary block**
4. Check if record is in **primary** or **overflow block**

Index I/O cost:

There are **3 levels**, so:
3 I/Os for index traversal

Data file I/O cost:

Fraction of records in primary block = 10 / 15 = 2/3

Fraction of records in overflow block = 5 / 15 = 1/3

- If record in primary block → 1 I/O
- If record in overflow block → 2 I/Os (one for primary + one for overflow)

$$\text{Average I/Os for data file} = \frac{2}{3}(1) + \frac{1}{3}(2) = \frac{4}{3}$$

Total Average I/Os

$$\text{Total I/Os} = 3\ (\text{index}) + \frac{4}{3}\ (\text{data}) = \frac{13}{3} \approx 4.33\ \text{I/Os}.$$

✅ **Average I/O cost = 4⅓ per lookup**

🧩 6. Why these assumptions matter

Concept	Role in calculation
Sparse index	Reduces index size (only one pointer per primary block).
60% full index blocks	Reflects typical real-world utilization.
Half-full overflow blocks	Models average insertion performance in sequential files.
Multilevel index	Ensures efficient lookup: (O(\log n)) I/O levels.
I/O cost formula	Weighted average depending on where record resides.

✅ Summary

Step	Concept	Key Result
Data file size	15 records per primary+overflow pair	432,000 blocks
Sparse multilevel index	3 levels, 60% full	3,661 blocks
Average retrieval I/Os	3 (index) + 1.33 (data)	≈ 4.33 I/Os
