DBMS process

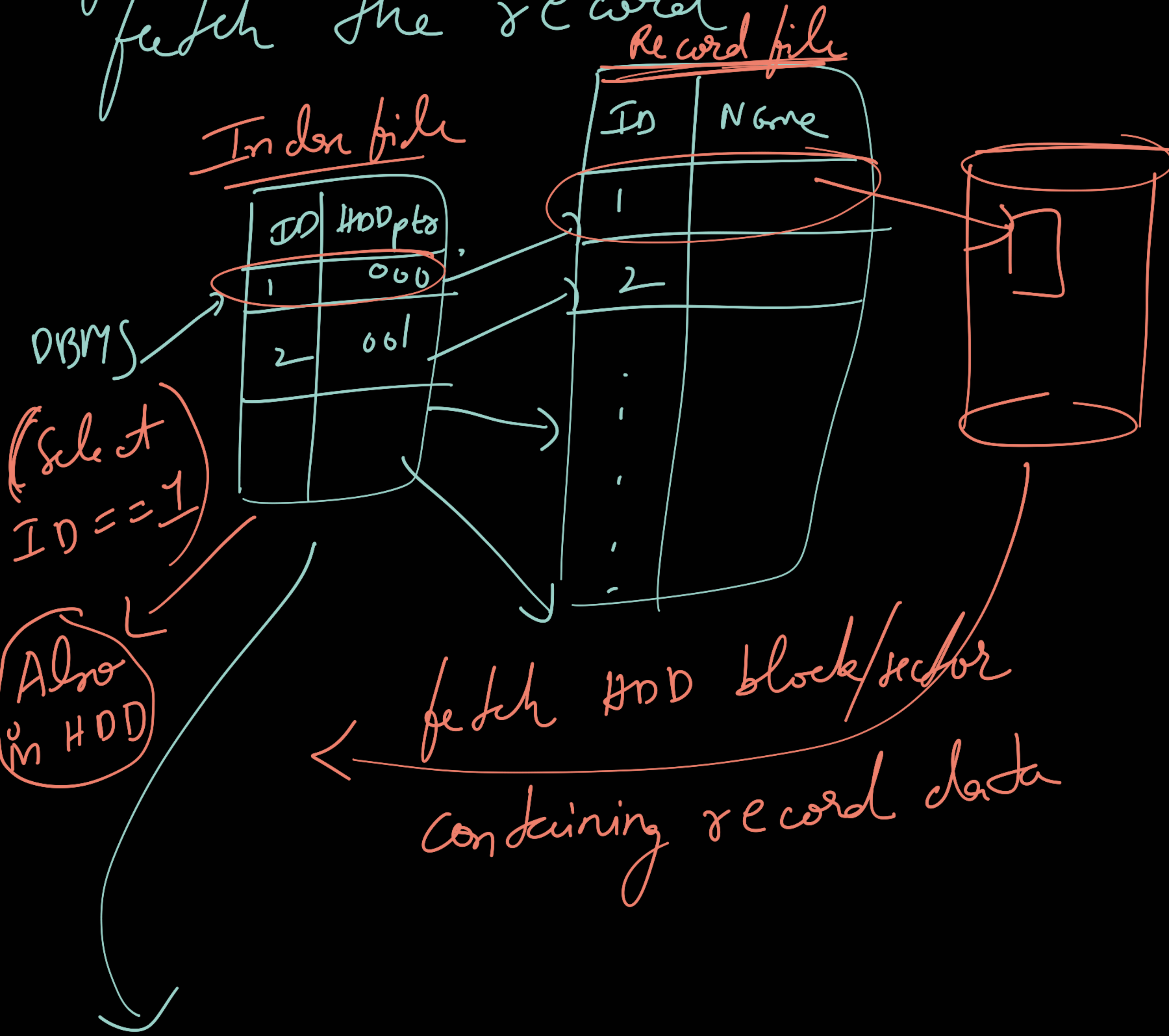
⇓

check index file

check which rows satisfies query

⇓

goto actual HDD blocks to fetch the record



How to make the searching in index file faster

- Single level index
  - Binary
  - Secondary
  - Clustered
- Multi-level
  - Static → can't grow in levels
  - Dynamic
    - B tree
    - B^+ tree

* Levels can grow or shrink

* Note B, B^+ tree is data structure for storing Index file

Yes, that's correct! In a DBMS, the **index files** use data structures like B-Trees or B+ Trees for efficient lookups. However, the **actual files containing the table's data records** are managed by the DBMS in conjunction with the underlying operating system's file system. The data structure and storage mechanism for these files depend on both the DBMS's implementation and the OS file system.

1. **How DBMS Stores Records**

- **Page-Based Storage**: Most DBMSs organize data records into **pages** (fixed-size chunks of data, typically 4KB, 8KB, or more). These pages are written to and read from disk as part of the database's physical storage model.
- **Heap Storage (Unordered)**: By default, many DBMSs store table rows in **heap files**. In heap storage:
  - Rows are written to available space in no specific order.
  - Retrieval without an index requires a full table scan.
- **Clustered Storage (Ordered)**: If a clustered index is present, the records are stored in the same physical order as the index, ensuring efficient range-based access.

2. **Role of the OS File System**

The actual low-level storage and management of files containing database pages or records depend on the operating system's file system. For example:

- In **Windows**, the DBMS may store files on an NTFS (New Technology File System) partition.
- In **Linux**, it could use ext4, XFS, or other file systems.

Example Workflow

1. A query is issued with a condition (e.g., `SELECT * FROM users WHERE id = 101`).
2. The DBMS uses the **index file** (stored as a B/B+ Tree) to quickly locate the page containing `id = 101`.
3. Using the pointer provided by the index, the DBMS retrieves the **record** from the data file stored on disk.

The OS file system is responsible for:

- Allocating and managing disk blocks for the DBMS files.
- Handling low-level operations like reading/writing blocks to disk.
- Ensuring file integrity and enabling features like journaling.

3. DBMS Abstraction from File System

- While the OS manages the physical files, the DBMS implements its own file abstraction and access methods to handle database pages, transactions, and recovery mechanisms. This abstraction ensures database operations are consistent, efficient, and recoverable, regardless of the underlying file system.

4. Why DBMS Abstracts Storage

- **Performance Optimization**: DBMSs use their own algorithms to manage data, including caching, prefetching, and page replacement policies, to optimize disk I/O.
- **Transaction Management**: DBMSs implement logging and recovery mechanisms (like Write-Ahead Logging) to ensure ACID properties, which are independent of the OS file system.
- **Portability**: By abstracting the file system, DBMSs can work across different operating systems and file systems with minimal changes.