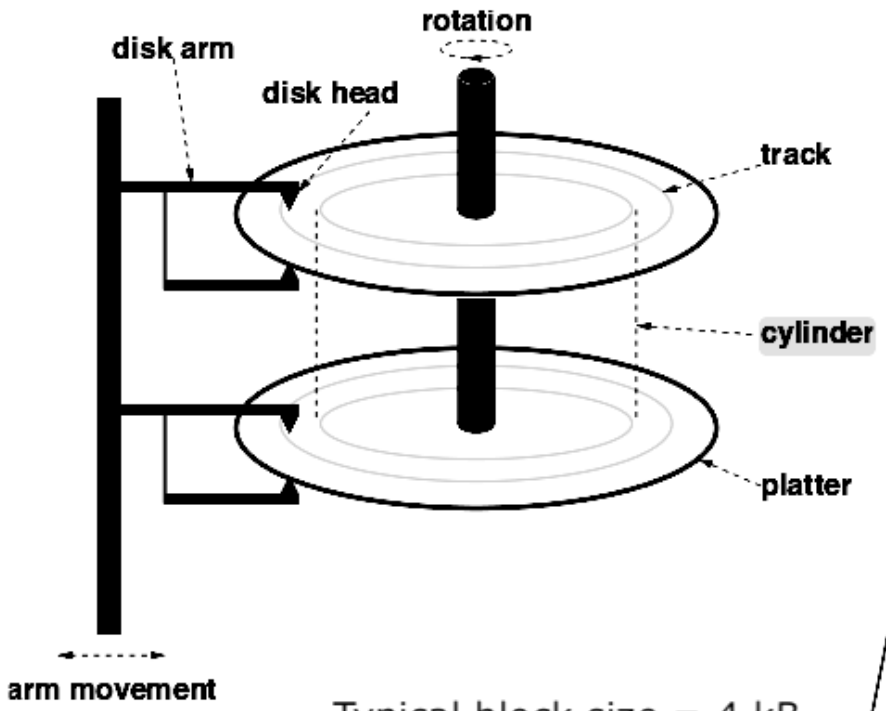
How does a hard disk work?



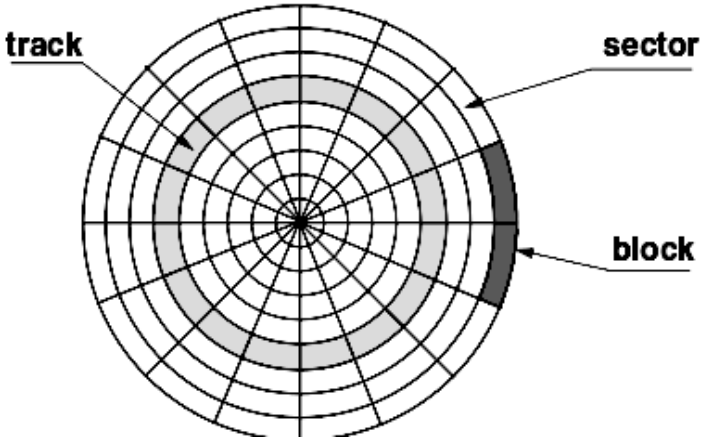
Typical block size = 4 kB

Access Time =

Seek Time (0-10 ms) +

Rotational Time (0-3 ms) +

Transfer time (.02 ms per 8K)



File Organization

- The database is stored as a collection of *files*. Each file is a sequence of *records.* A record is a sequence of fields.
- One approach:
  - assume record size is fixed
  - each file has records of one particular type only
  - different files are used for different relations

Storage Access

- A database file is partitioned into fixed-length storage units called **blocks**. Blocks are units of both storage allocation and data transfer.
- Database system seeks to minimize the number of block transfers between the disk and memory. We can reduce the number of disk accesses by keeping as many blocks as possible in main memory.
- **Buffer** – portion of main memory available to store copies of disk blocks.
- **Buffer manager** – subsystem responsible for allocating buffer space in main memory.

Directory

File	Index Block
File-1	22

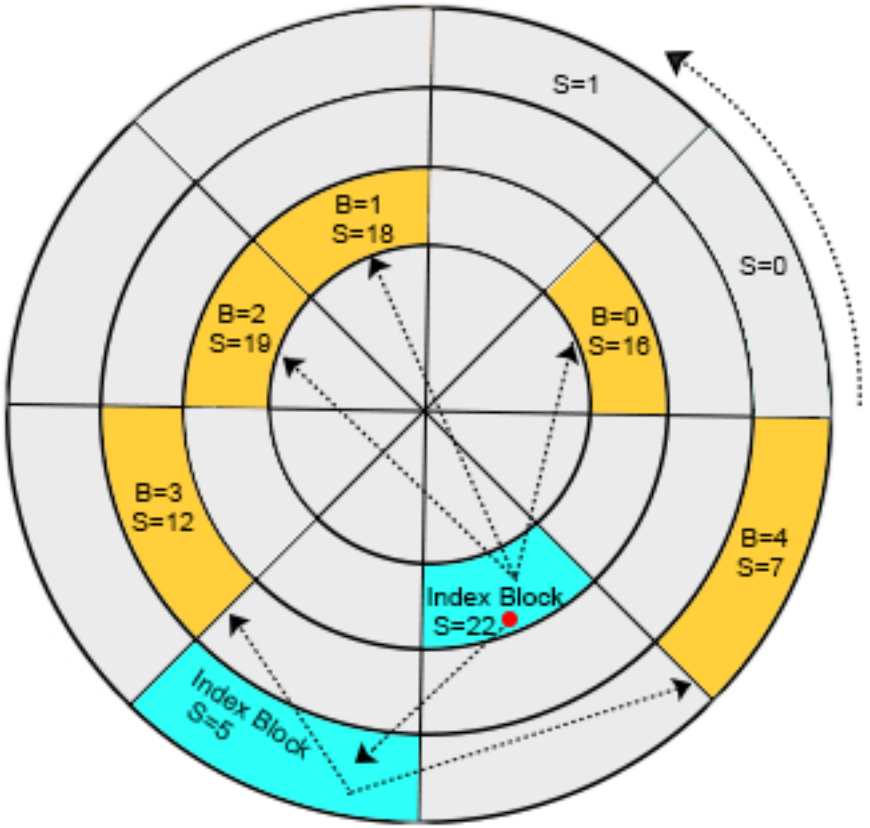
S= Sector
B= Block

File-1 blocks
File-2 blocks
Index blocks
Filled Blocks
By other files

0
1
2
3
4

File-1
with 5 blocks

One index block capability, to represent 3 disk-blocks



Hard Disk

THINK ABOUT A QUERY S.T. "FIND ALL BANK ACCOUNTS OWNED BY ROSS" - THIS QUERY ACCESSES LOT & SMALL PROPORTION OF DATA AND IT WOULD BE INEFFICIENT FOR THE SYSTEM TO READ EVERY ACCOUNT RECORD AND CHECK IT OWNER - IDEALLY, THIS SYSTEM SHOULD BE ABLE TO LOCATE THE FEW RECORDS DIRECTLY -

↳ WE USED ADDITIONAL → INDEXES DATA STRUCTURES TO DO THAT

Chapter 11: Indexing and Hashing

INDEXES WORK SIMILARLY TO ANALYTICAL INDEXES AT THE END OF A BOOK - IF WE WANT TO LEARN ABOUT A PARTICULAR TOPIC (SPECIFIED BY A WORD/WORDS), WE CAN SEARCH FOR SUCH WORDS IN THE ANALYTICAL INDEX, FIND THE PAGES WHERE THEY OCCUR, AND THEN READ THE PAGES TO FIND THE INFORMATION WE ARE LOOKING FOR

↳ WORDS ARE SORTED

↳ INDEX IS MUCH SMALLER THAN THE BOOK so you have to read less pages

Basic Concepts

DO NOT CONFUSE IT WITH RELATION KEY

- Indexing mechanisms used to speed up access to desired data.
  - E.g., author catalog in library — NAME OF THE AUTHOR → BOOK AND SHELF WHERE THE BOOKS OF THE AUTHOR ARE STORED
- **Search Key** - attribute or set of attributes used to look up records in a file.
- An **index file** consists of records (called **index entries**) of the form

E.G. AUTHOR NAME ← search-key | pointer(s) → IDENTIFIERS OF THE DISC BLOCKS WHERE THE INFO. LIES

THEN, WE CAN TRANSFER JUST THESE BLOCKS TO MAIN MEMORY

- Index files are typically much smaller than the original file
- Two basic kinds of indices:
  - **Ordered indices:** search keys are stored in sorted order

SO THEY CAN BE TRANSFERRED TO MAIN MEMORY MORE EFFICIENTLY THAN ENTIRE TABLES

THUS, AN INDEX IS BUILT OVER A SET OF TABLE COLUMNS.

Index Evaluation Metrics

- Access types supported efficiently. E.g.,
  - records with a specified value in the attribute (point queries)
  - or records with an attribute value falling in a specified range of values (range queries).
- Access time
- Insertion time
- Deletion time
- Space overhead

WHENEVER YOU MODIFY THE CONTENT OF A TABLE, YOU MUST MODIFY THE CONTENT OF THE INDEXES ACCORDINGLY

ALSO, IT IS NOT OBVIOUS THAT AN INDEX DEFINED OVER A TABLE WILL SPEED UP THE INVOLVED QUERIES; IT DEPENDS ON MANY FACTORS TAKEN INTO ACCOUNT BY THE OPTIMIZER –

→ E.G., TABLE SO SMALL THAT IT CAN BE LOADED ENTIRELY IN MEMORY: IT IS TYPICALLY BETTER TO WORK DIRECTLY ON THE TABLE –