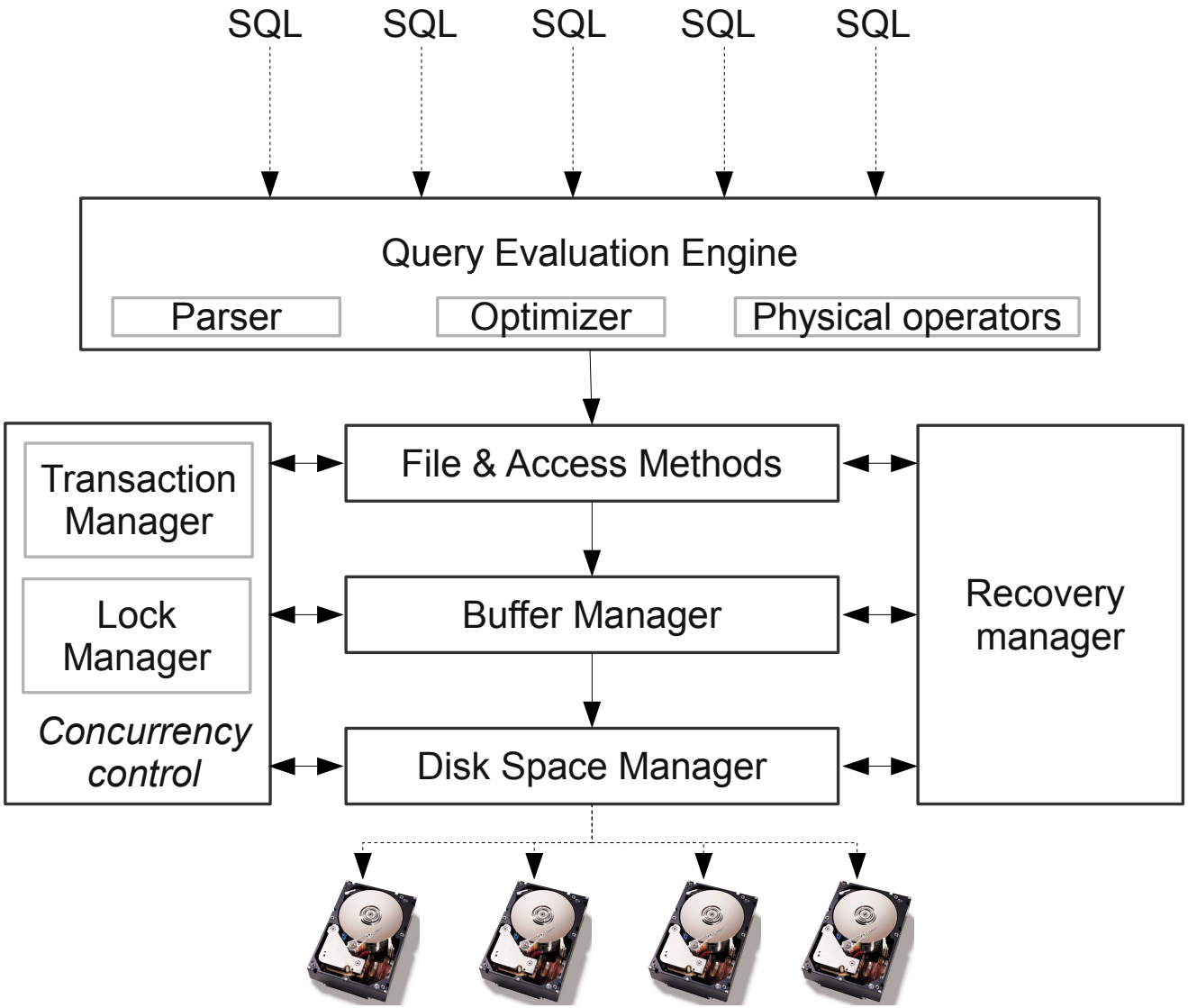
Intermezzo: A typical database architecture

A typical database architecture



A typical database architecture

Main components

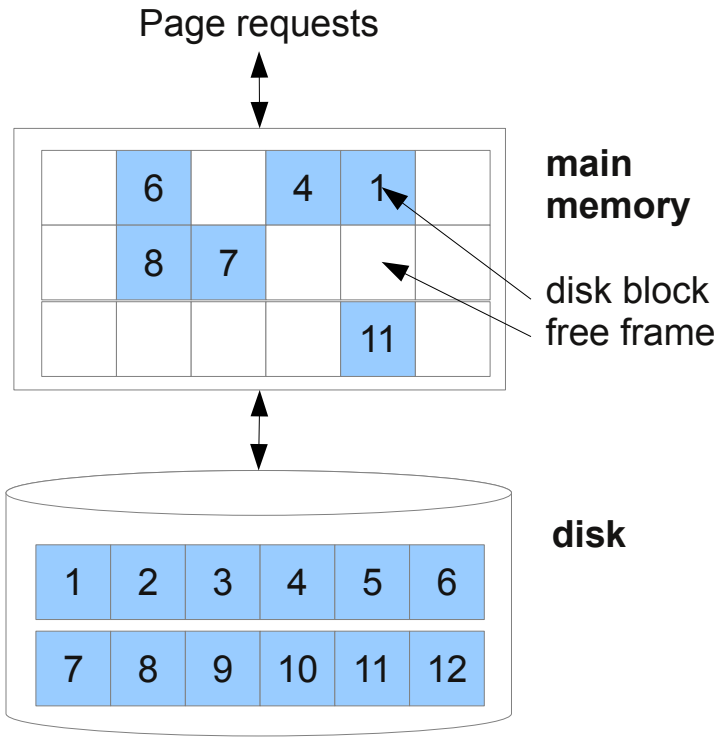
- The lowest layer of the DBMS software deals with management of space on disk, where the data is stored. Higher layers allocate, deallocate, read, and write blocks through (routines provided by) this layer, called the disk space manager or storage manager.
- The buffer manager brings blocks in from disk to main memory in response to read requests from the higher-level layers.
- The file and access methods layer supports the concept of reading and writing files (as collection of blocks or a collection of records) as well as indexes. In addition to keeping track of the blocks in a file, this layer is responsible for organizing the information within a block.
- The query evaluation engine, and more in particular the code that implements relational operators, sits on top of the file and access methods layer.
- The DBMS supports concurrency and crash recovery by carefully scheduling user requests and maintaining a log of all changes to the database. These tasks are managed by the concurrency control manager and the recovery manager.

A typical database architecture

Disk Space Manager

- The disk space manager manages space on disk.
- Abstractly, it supports the concept of a block as a unit of data and provides commands to allocate or deallocate a block and read or write a block.
- A database grows and shrinks when records are inserted and deleted over time. The disk space manager keeps track of which disk blocks are in use. Although it is likely that blocks are initially allocated sequentially on disk, subsequent allocations and deallocations could in general create 'holes.' One way to keep track of block usage is to maintain a list of free blocks. When blocks are deallocated, they are added to the free list for future use.
- The disk space manager hides details of the underlying hardware and operating system and allows higher levels of the software to think of the data as a collection of blocks.
- Although it typically uses the file system functionality provided by the OS, it provides additional features, like the possibility to distribute data on multiple disks, etc.

A typical database architecture



Buffer Manager

- Mediates between external storage and main memory
- Maintains a designated main memory area, called the buffer pool for this task.
- The buffer pool is a collection of memory slots where each slot (called a frame or buffer) can contain exactly one block.
- Disk blocks are brought into memory as needed in response to higher-level requests.
- A replacement policy decides which block to evict when the buffer is full.

A typical database architecture

Buffer Manager (continued)

- Higher levels of the DBMS code can be written without worrying about whether data blocks are in memory or not: they ask the buffer manager for the block, and the buffer manager loads it into a slot in the buffer pool if it is not already there.
- The higher-level code must also inform the buffer manager when it no longer needs a block that it has requested to be brought into memory. That way, the buffer manager can re-use the slot for future requests.
- A buffer whose block contents should remain in memory (e.g., because a routine from a higher-level layer is working with its contents) is called pinned. The act of asking the buffer manager to read a disk block into a buffer slot is called pinning and the act of letting the buffer manager know that a block is no longer needed in memory is called unpinning.
- When higher-level code unpins a block, it must also inform the buffer manager whether it modified the requested block; the buffer manager then makes sure that the change is eventually propagated to the copy of the block on disk.

A typical database architecture

Buffer Manager (continued)

- When the buffer manager receives a block pin request, it checks whether the block is already in memory (because another DBMS component is working on it, or because it was recently loaded but then unpinned). If so, the corresponding buffer is re-used and no disk I/O takes place.
- If not, the buffer manager has to decide a buffer frame to load the block into from disk. If there are no empty frames available, the buffer manager has to select a frame containing a block that is currently unpinned, write the contents of that block back to disk if modifications are made, and load the requested block from disk into the frame.
- The strategy by which the buffer manager chooses the slot to release back to disk is called the buffer replacement policy. Popular policies are FIFO, Least recently used, Clock.

A typical database architecture

Buffer Management in Reality

- Prefetching
  - Buffer managers try to anticipate page requests to overlap CPU and I/O operations.

    Speculative prefetching Assume sequential scan and automatically read ahead.

    Prefetch lists Some database algorithms can inform the buffer manager of a list of blocks to prefetch.
- Page fixing/hating
  - Higher-level code may request to fix a page if it may be useful in the near future (e.g., index pages).
  - Likewise, an operator that hates a page won't access it any time soon (e.g., table pages in a sequential scan).
- Multiple buffer pools
  - E.g., separate pools for indexes and tables.