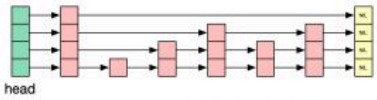
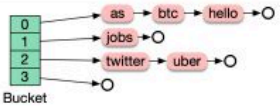
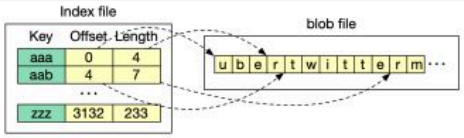
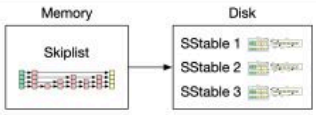
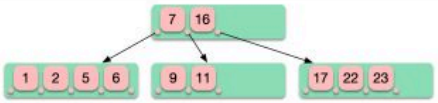
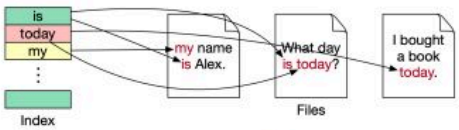
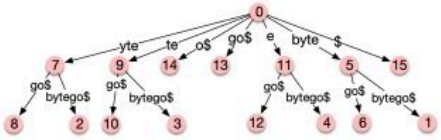
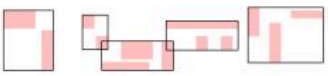
Eight Data Structures That Power Your Databases. Which one should we pick?

The answer will vary depending on your use case. Data can be indexed in memory or on disk. Similarly, data formats vary, such as numbers, strings, geographic coordinates, etc. The system might be write-heavy or read-heavy. All of these factors affect your choice of database index format.

8 Data Structures That Power Your Databases — ByteByteGo.com

Types	Illustration	Use Case	Note
Skiplist		In-memory	used in Redis
Hash index		In-memory	Most common in-memory index solution
SSTable		Disk-based	Immutable data structure. Seldom used alone
LSM tree		Memory + Disk	High write throughput. Disk compaction may impact performance
B-tree		Disk-based	Most popular database index implementation
Inverted index		Search document	Used in document search engine such as Lucene
Suffix tree		Search string	Used in string search, such as string suffix match
R-tree		Search multi-dimension shape	Such as the nearest neighbor

The following are some of the most popular data structures used for indexing data:

- Skiplist: a common in-memory index type. Used in Redis
- Hash index: a very common implementation of the “Map” data structure (or “Collection”)

- SSTable: immutable on-disk “Map” implementation
- LSM tree: Skiplist + SSTable. High write throughput
- B-tree: disk-based solution. Consistent read/write performance
- Inverted index: used for document indexing. Used in Lucene
- Suffix tree: for string pattern search
- R-tree: multi-dimension search, such as finding the nearest neighbor

This is not an exhaustive list of all database index types.

Over to you:

1. Which one have you used and for what purpose?
2. There is another one called “reverse index”. Do you know the difference between “reverse index” and “inverted index”?