QUICKWIT

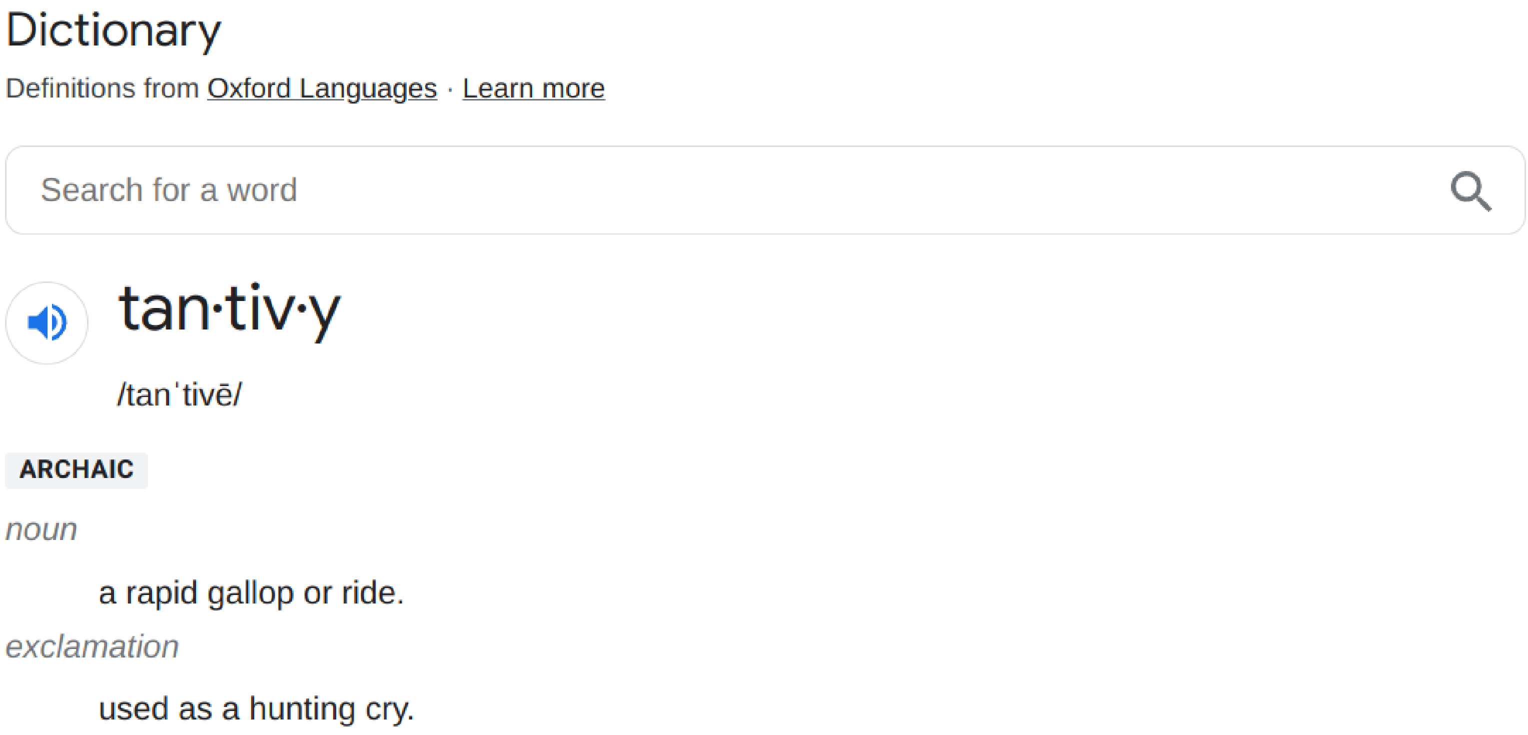
Intro to Tantivy

A full-text search engine library written in Rust

Hello 👋

- Adrien Guillo (@guilload)
- Software engineer
- Co-founder at Quickwit
- Two-time Recurser (Fall '14, Summer 1'17)

What is Tantivy?



Dictionary

Definitions from Oxford Languages · Learn more

tan·tiv·y

/tanˈtivē/

ARCHAIC

noun

a rapid gallop or ride.

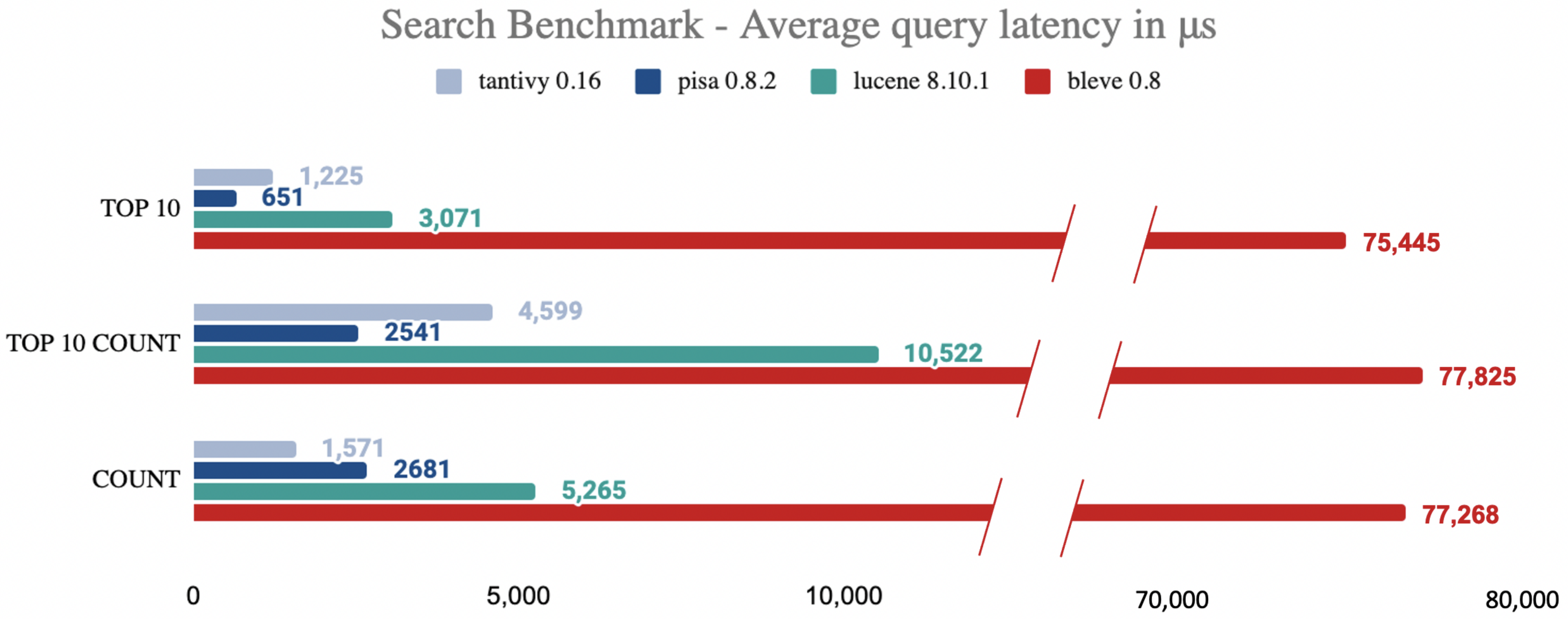
exclamation

used as a hunting cry.

What is Tantivy?

- Full-text search library inspired by Lucene
- Fully-featured (phrase queries, faceting, ranking, aggregations, ...)
- Mature and production-ready
- Open-source (MIT)

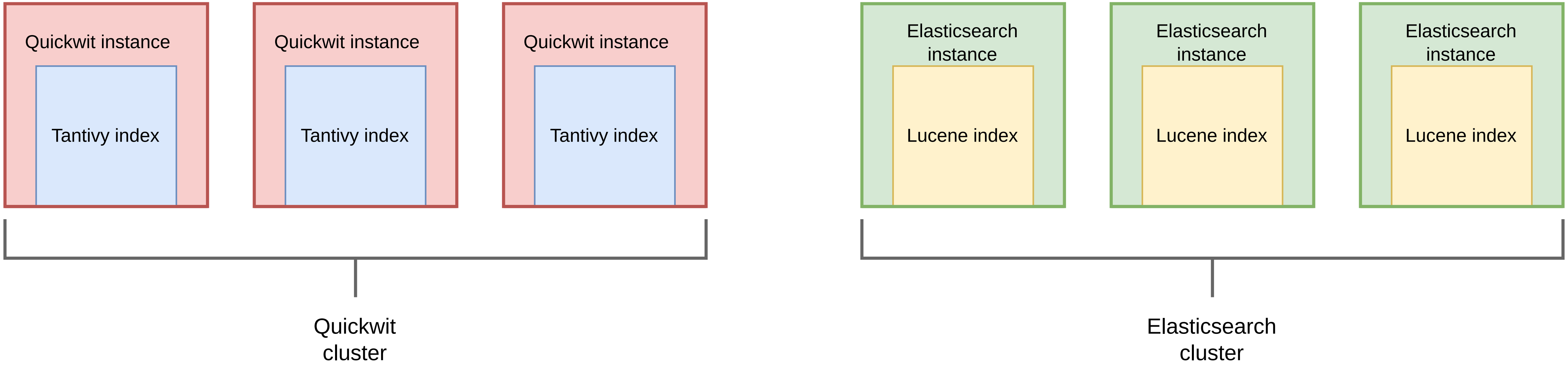
Tantivy is fast!



Benchmark source

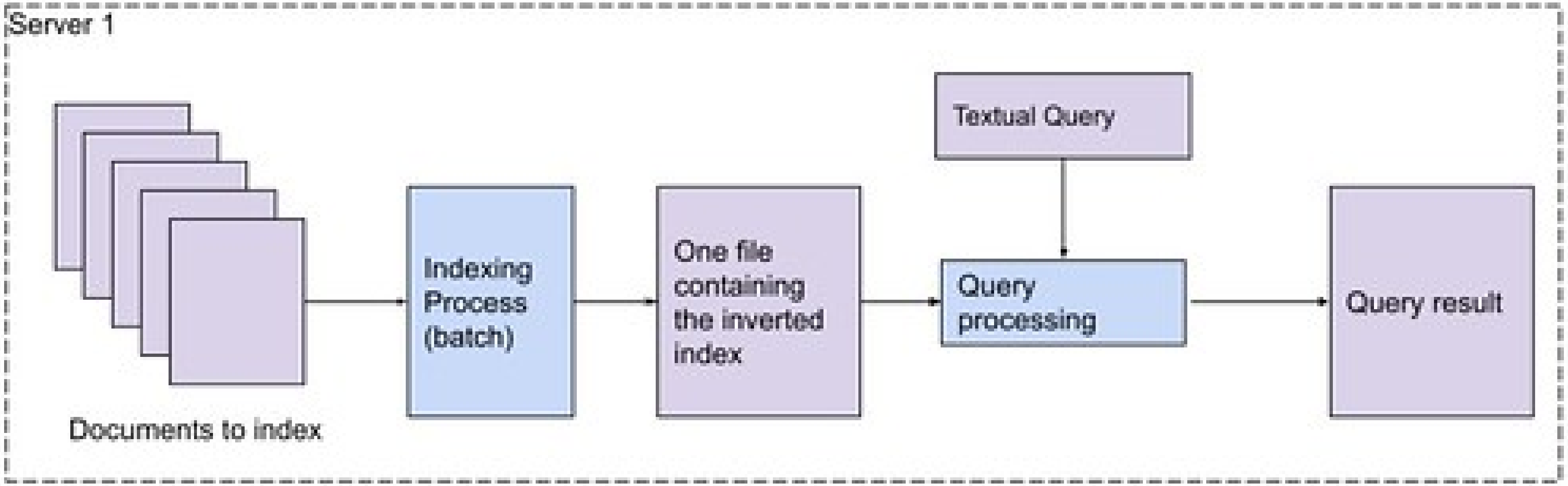
“Merely” a library

- Provides the building blocks for implementing search functionalities
- NOT a distributed search engine



How does Tantivy work?

Search architecture



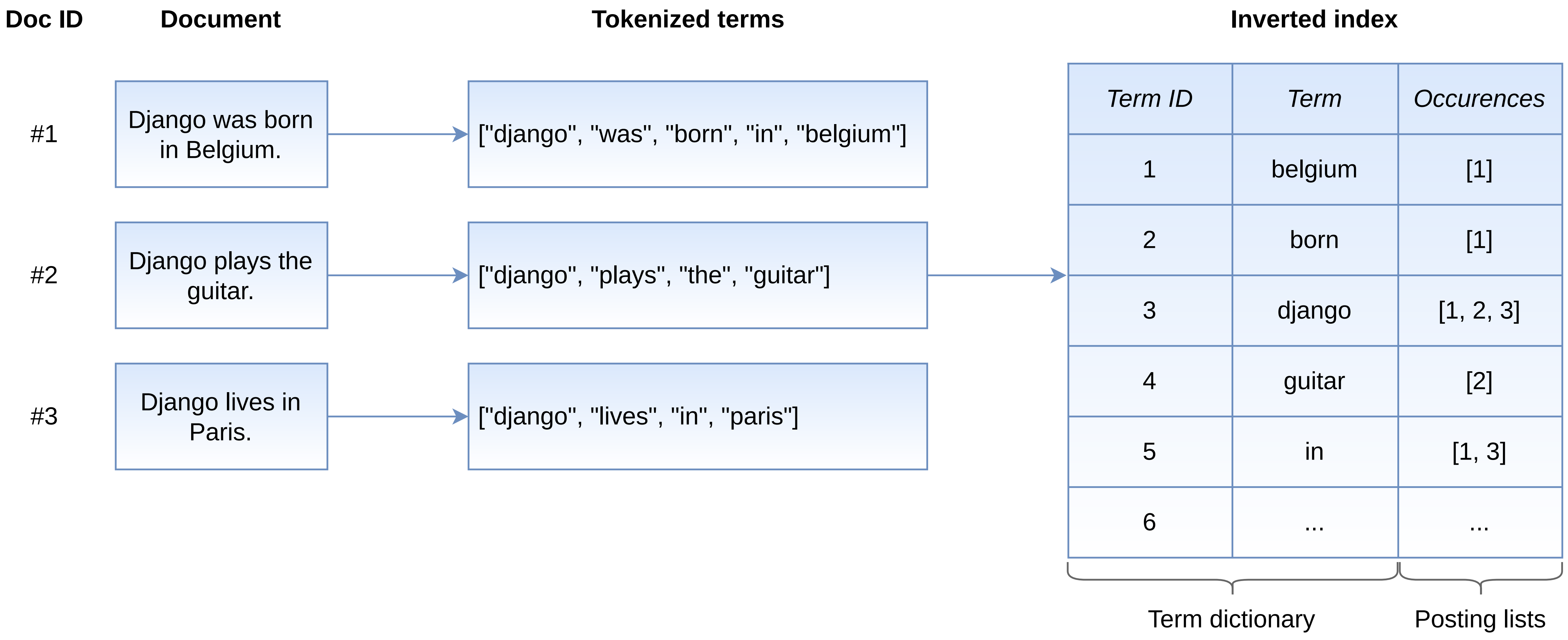
Anatomy of a search index

What are the data structures that compose a search index?

Anatomy of a search index

- Inverted index

Inverted index



Doc ID	Document	Tokenized terms
#1	Django was born in Belgium.	["django", "was", "born", "in", "belgium"]
#2	Django plays the guitar.	["django", "plays", "the", "guitar"]
#3	Django lives in Paris.	["django", "lives", "in", "paris"]

Inverted index

Term ID	*Term*	*Occurences*
1	belgium	[1]
2	born	[1]
3	django	[1, 2, 3]
4	guitar	[2]
5	in	[1, 3]
6	...	...

Term dictionary

Posting lists

Anatomy of a search index

- Inverted index
    - Posting lists

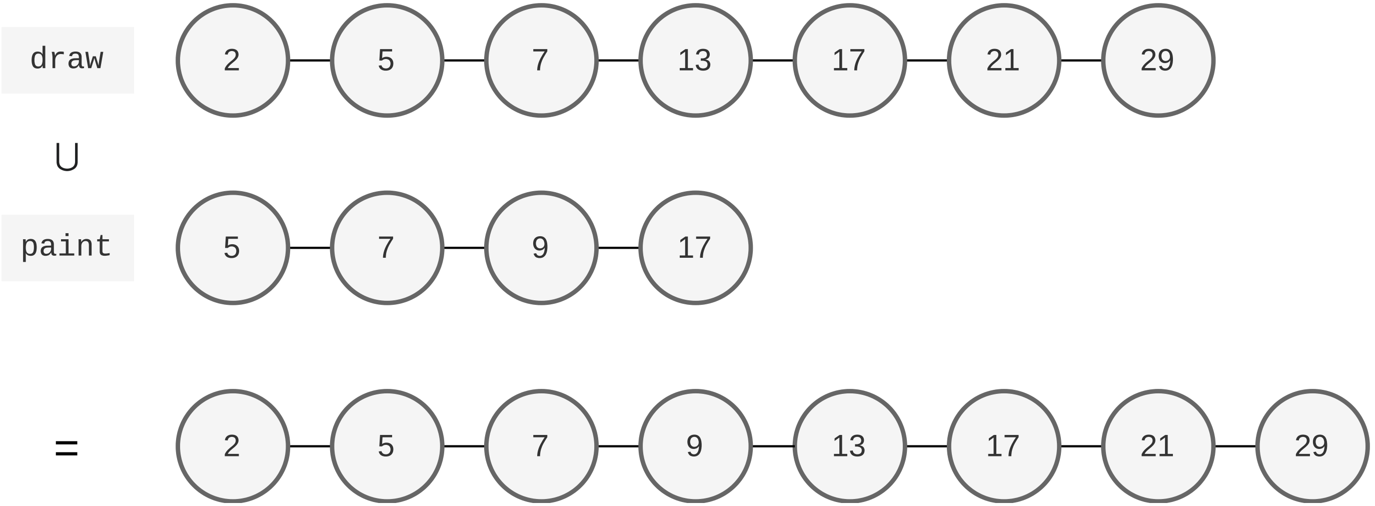
Posting list

- **Sorted** list of document IDs
- Efficient data structure for processing boolean queries

"django" → [1, 2, 3]

OR boolean query

- `draw OR paint`
- union of posting lists



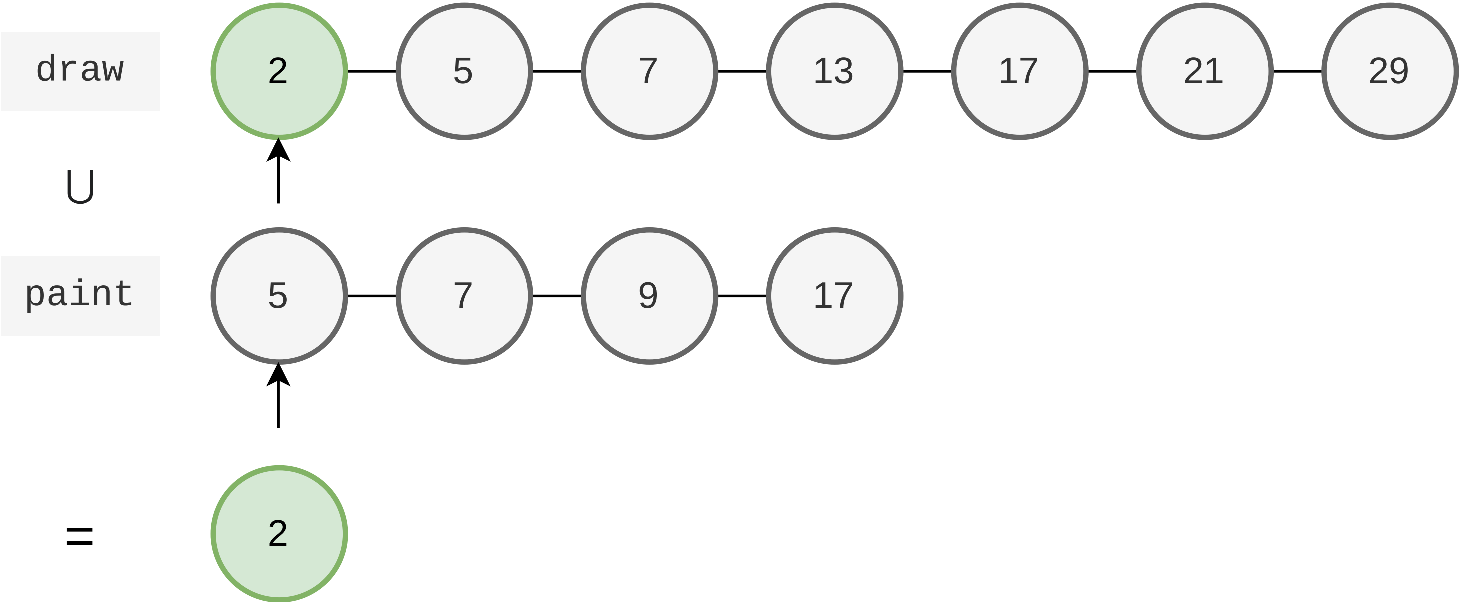
draw 2 5 7 13 17 21 29

∪

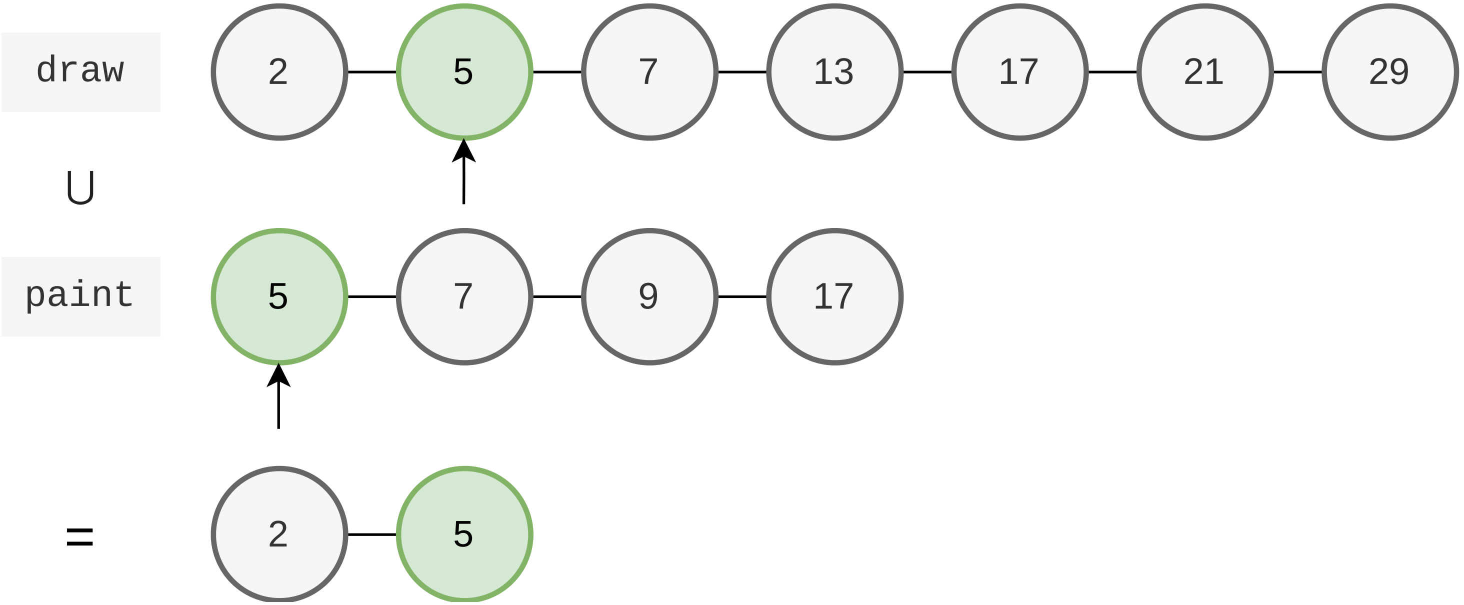
paint 5 7 9 17

= 2 5 7 9 13 17 21 29

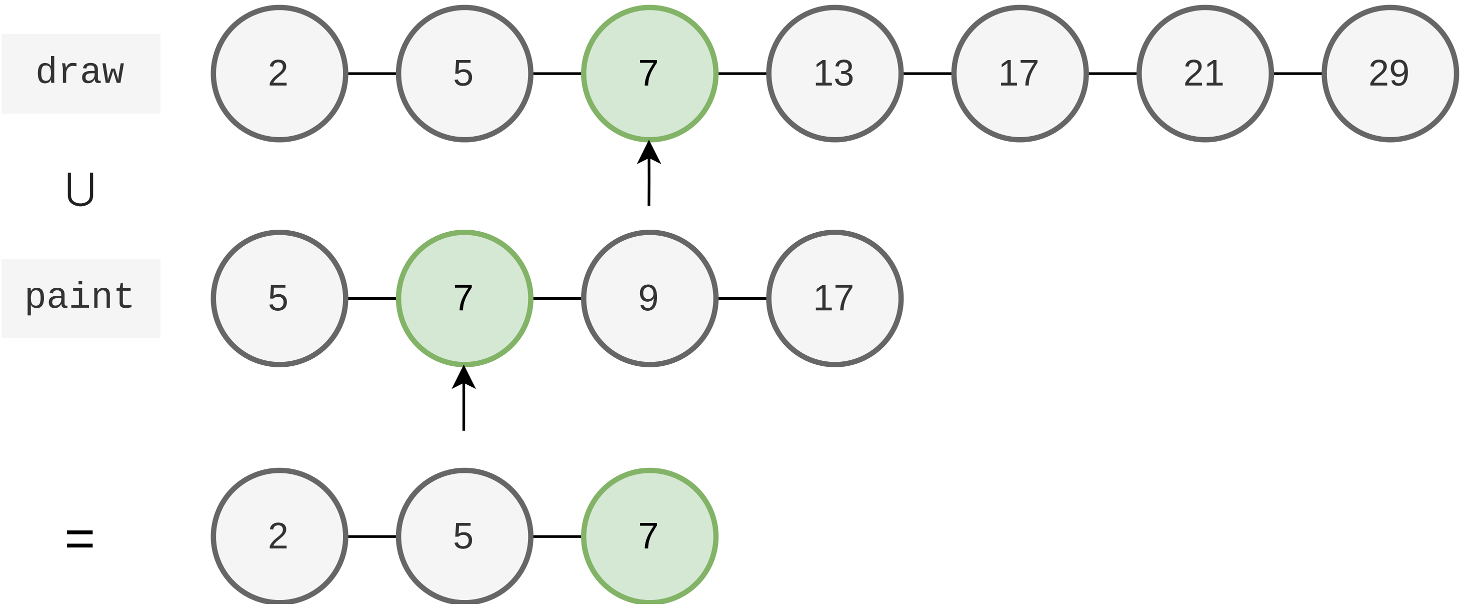
OR boolean query



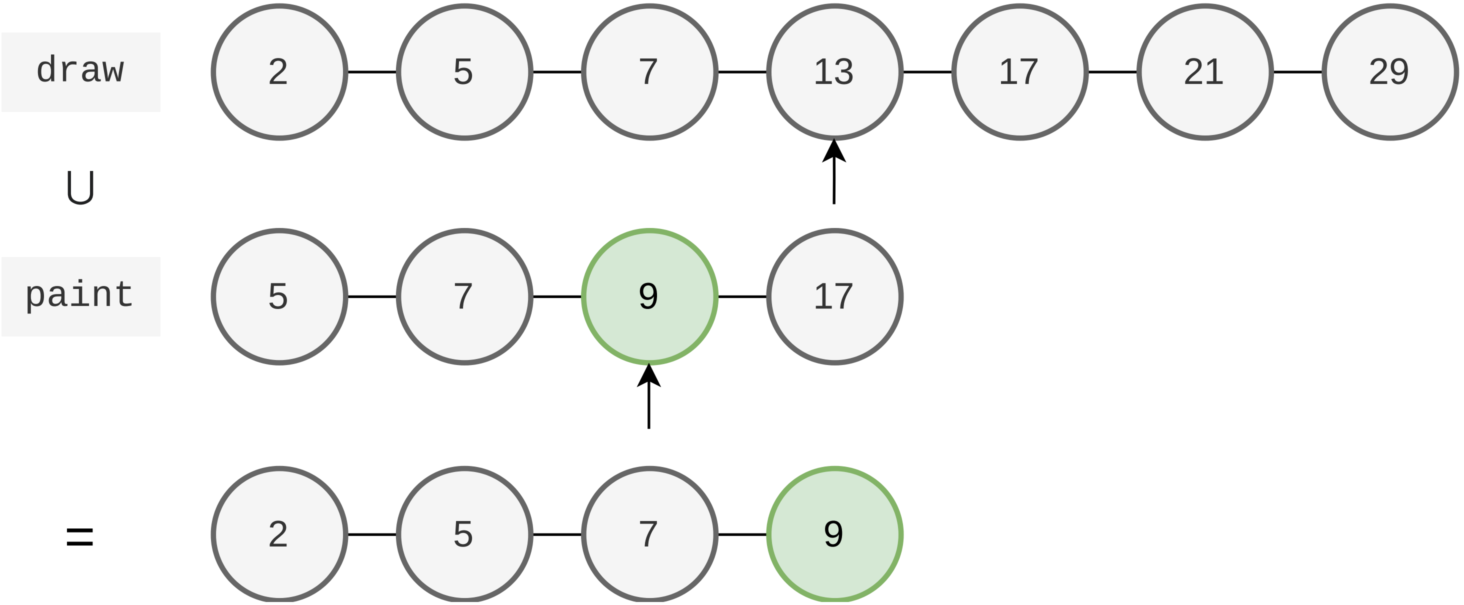
OR boolean query



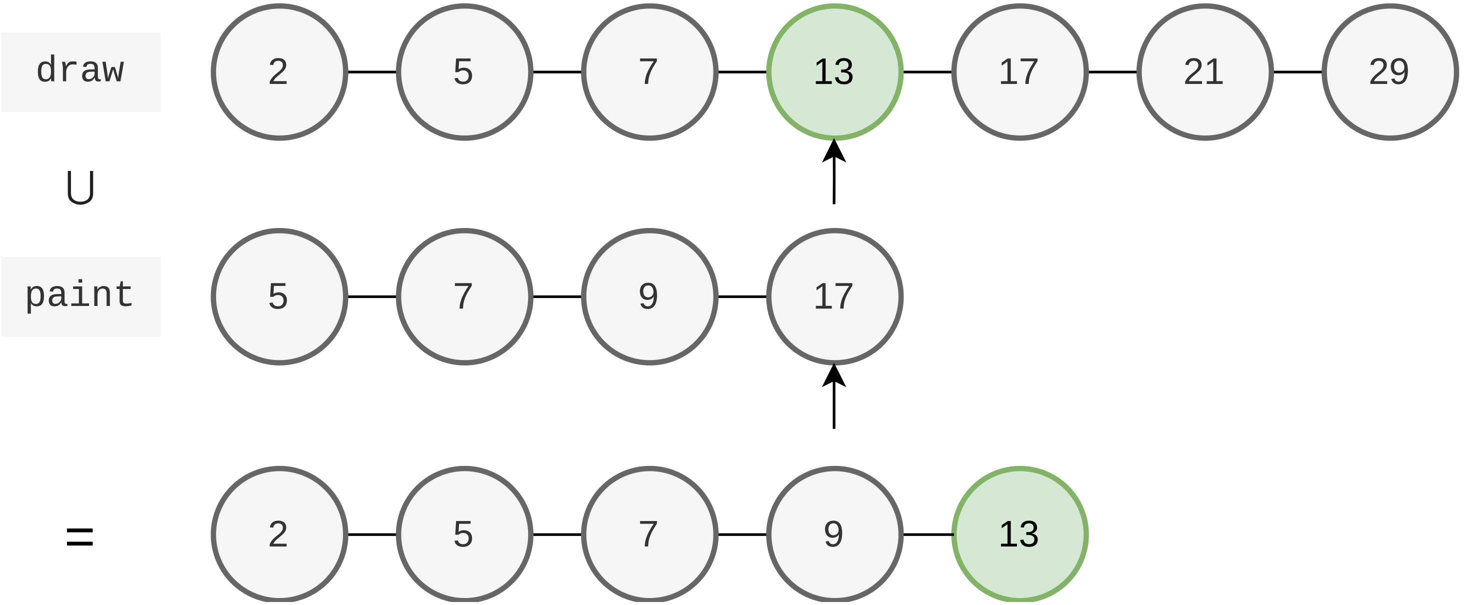
OR boolean query



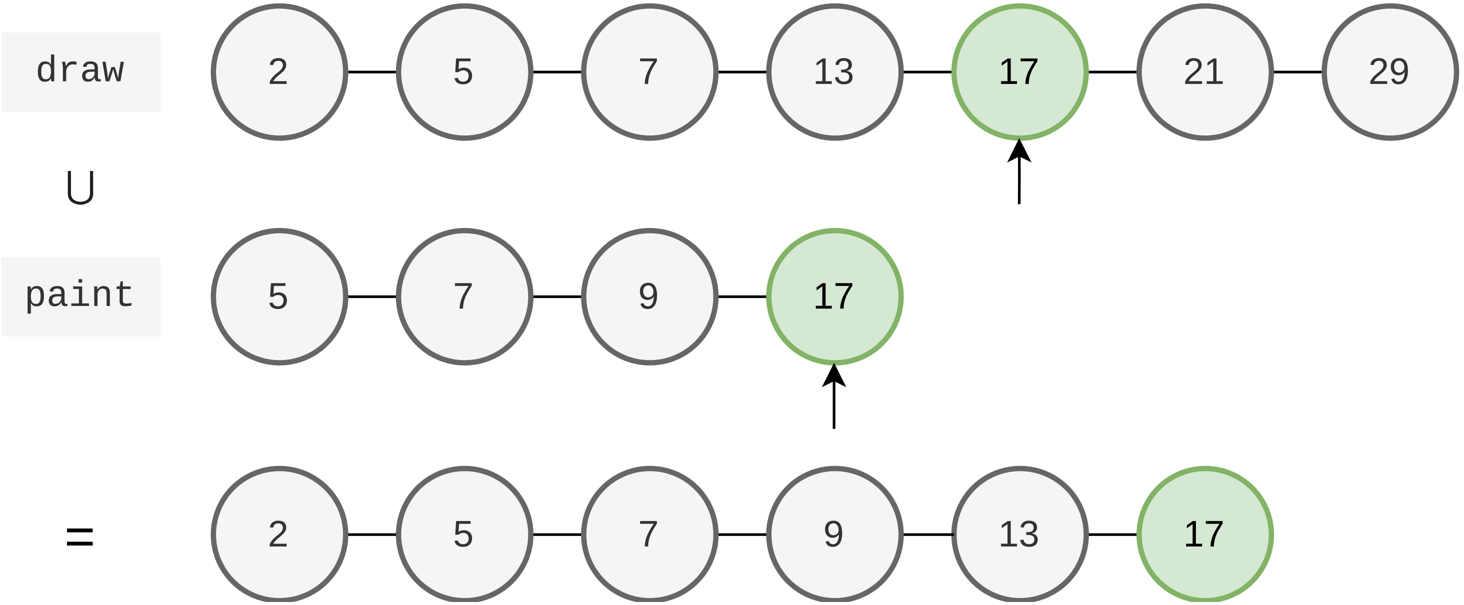
OR boolean query



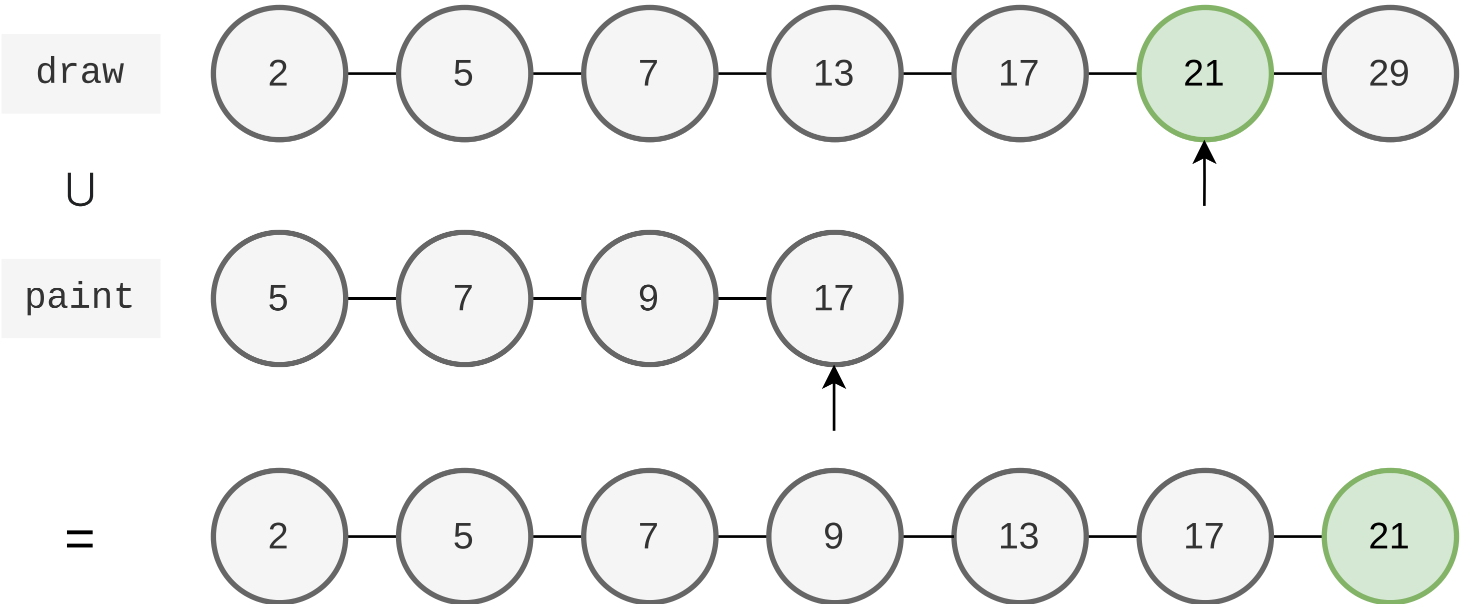
OR boolean query



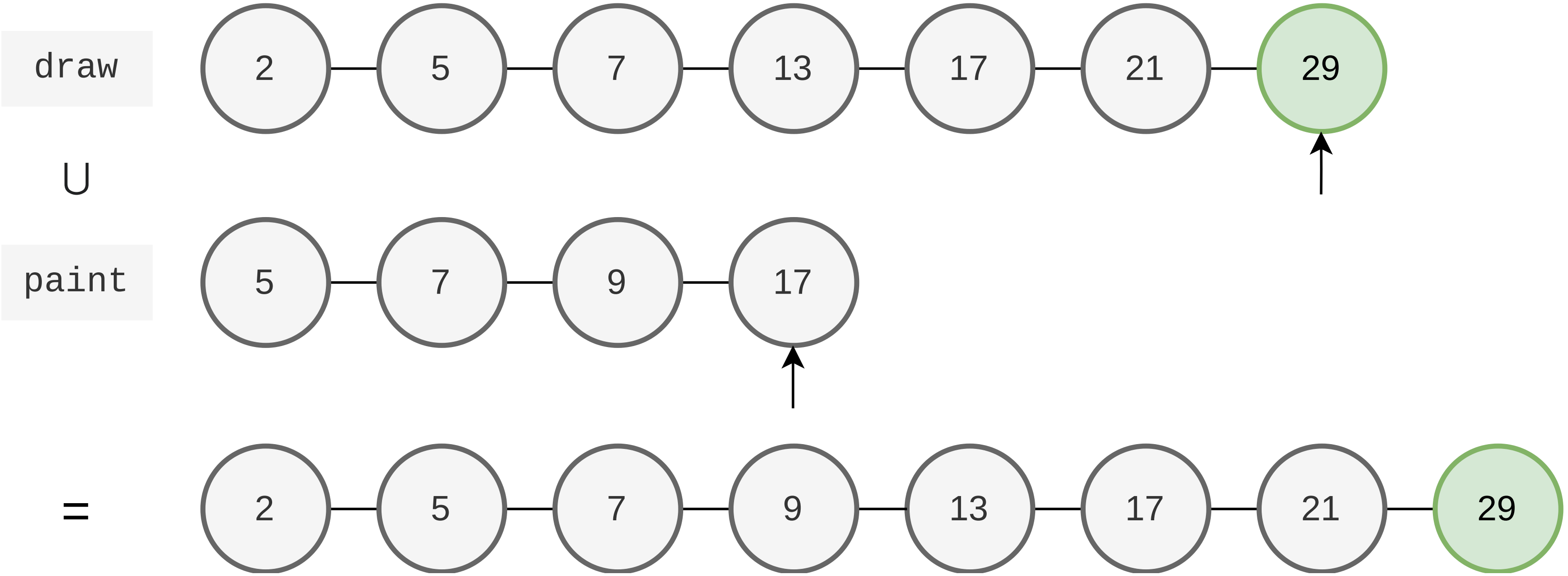
OR boolean query



OR boolean query

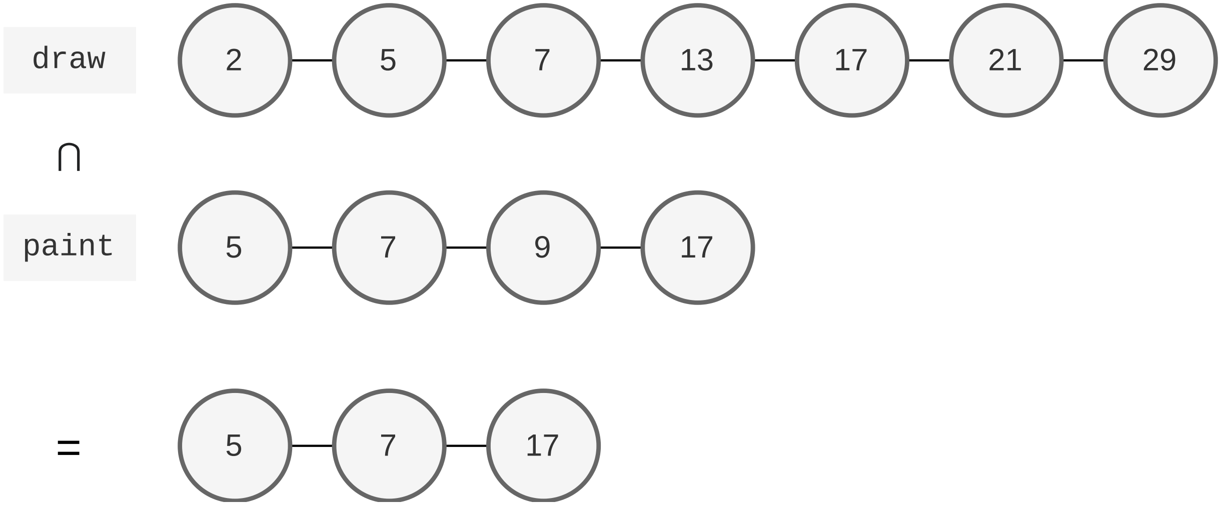


OR boolean query



AND boolean query

- `draw AND paint`
- intersection of posting lists



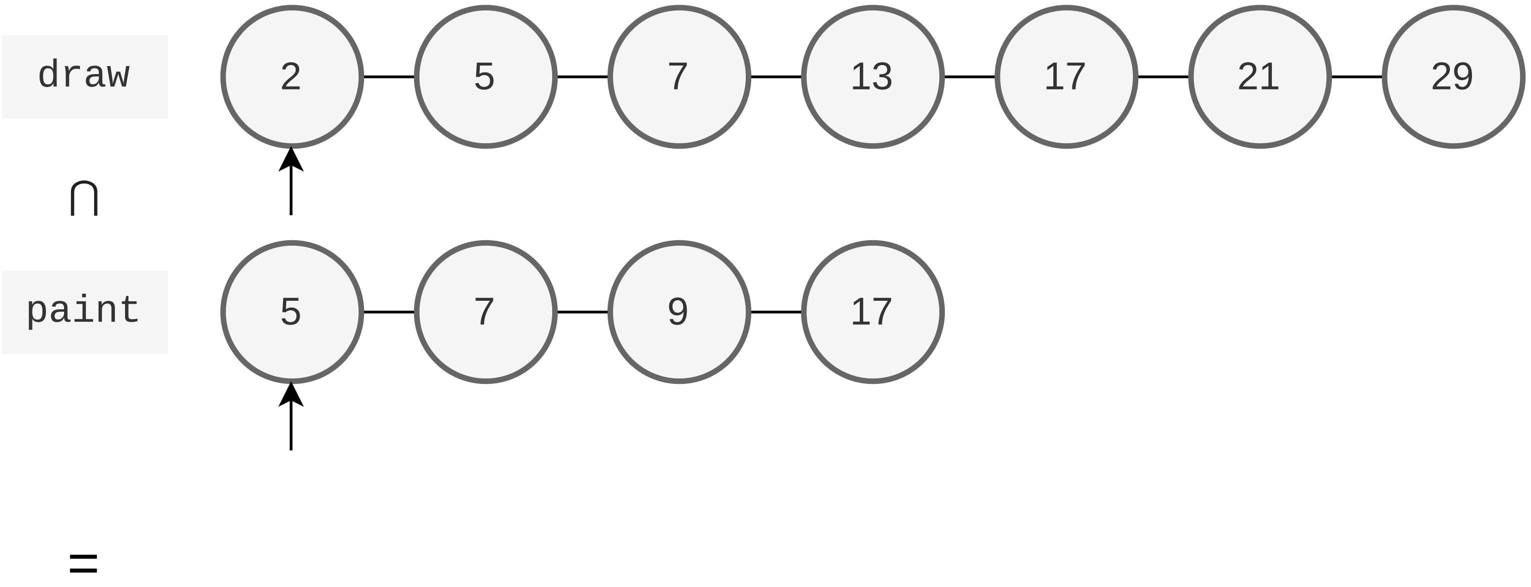
draw 2 5 7 13 17 21 29

∩

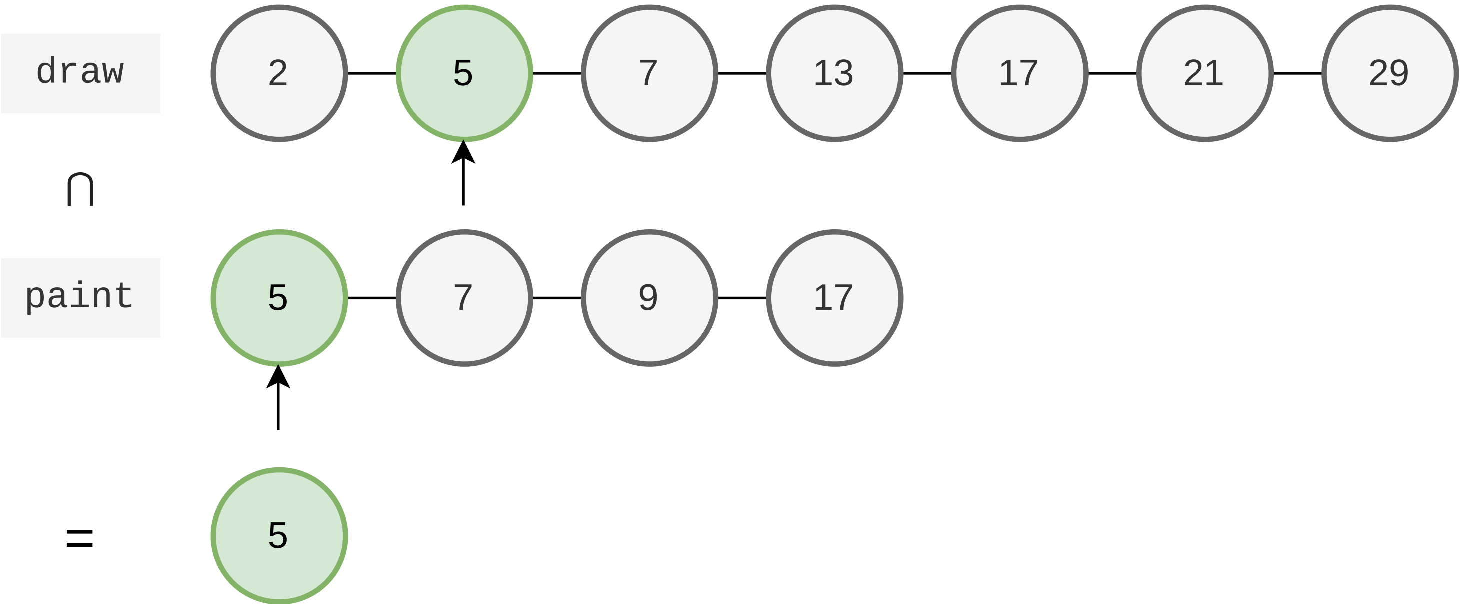
paint 5 7 9 17

= 5 7 17

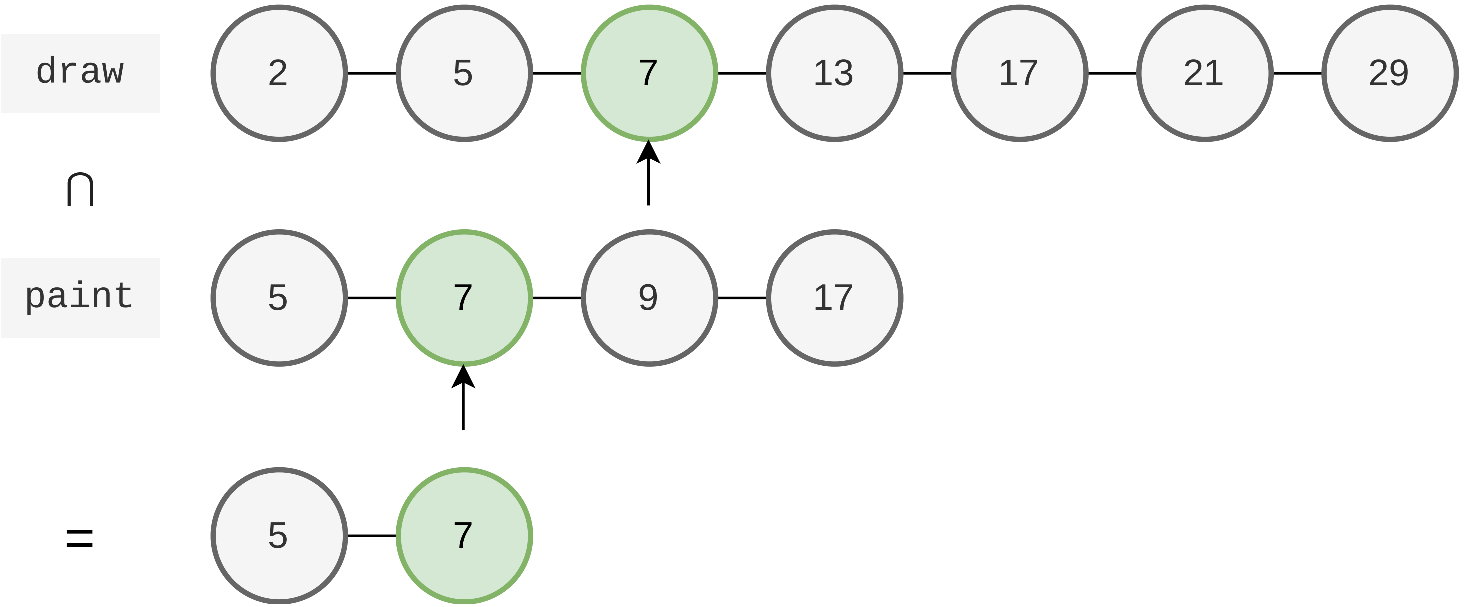
AND boolean query



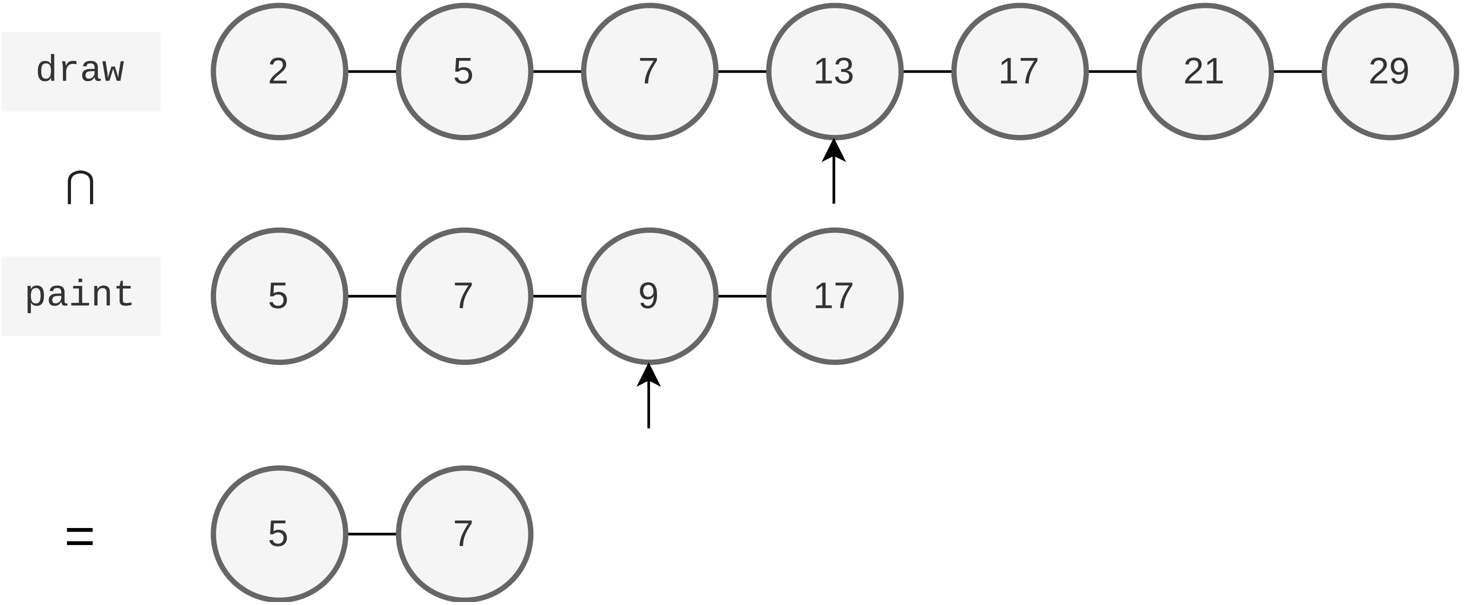
AND boolean query



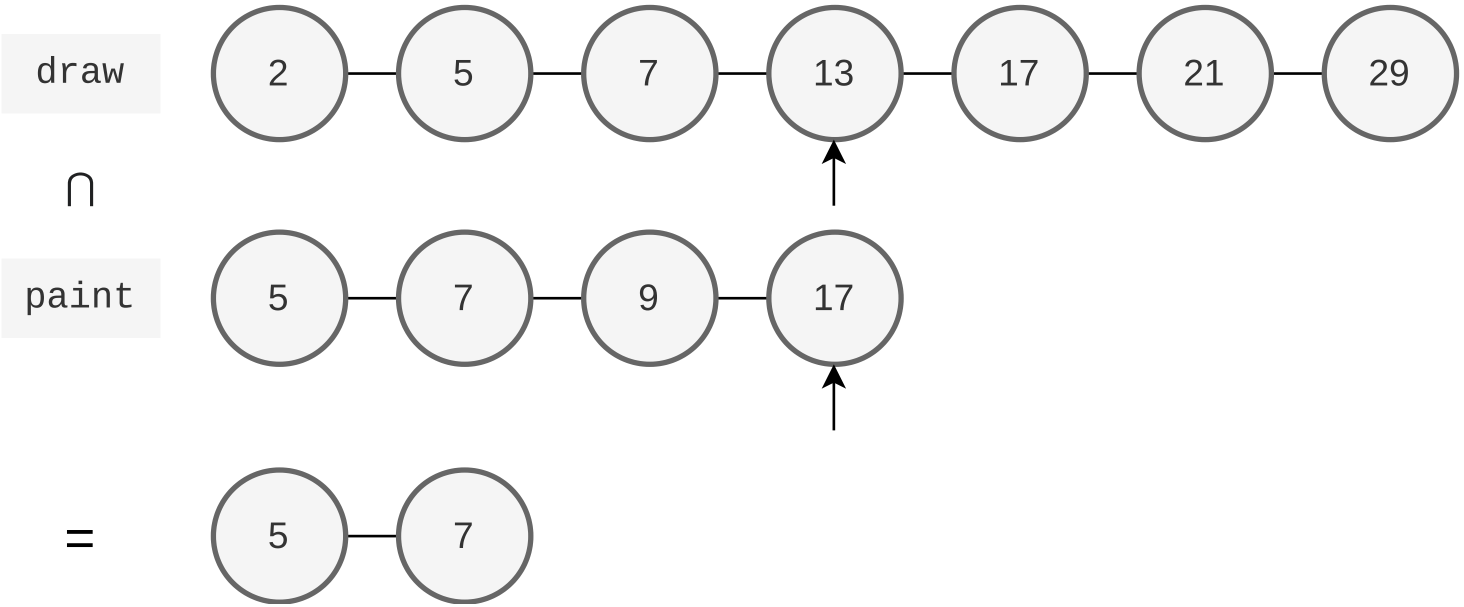
AND boolean query



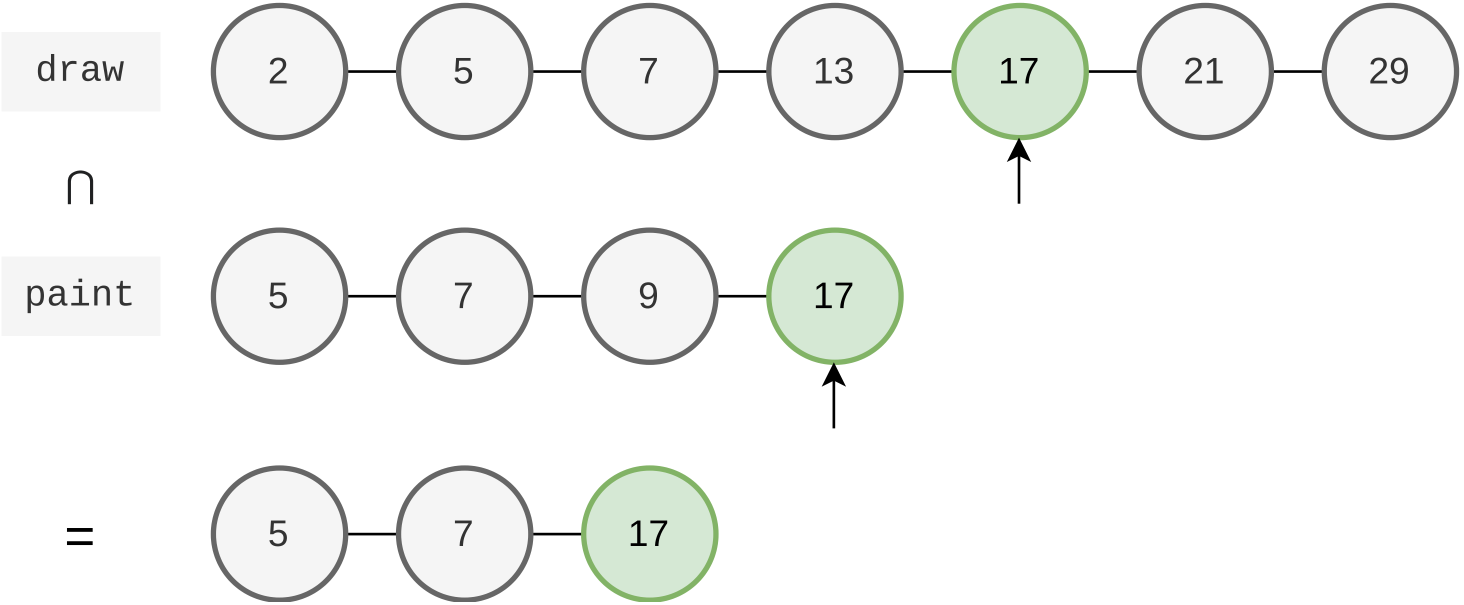
AND boolean query



AND boolean query



AND boolean query



Posting lists compression

Delta encoding

- Document IDs are large numbers
- but differences between subsequent document IDs in posting lists can be small
- especially true for frequent terms

Example:

[12, 17, 21, 36, , ...] → [12, (17 - 12), (21 - 17), (36 - 21), ...] → [12, 5, 4, 15, ...]

Bit packing

- Bit packing reduces the number of bits necessary to serialize a sequence of integers
- Performs better than dictionary-based compression schemes

Bit packing algorithm

1. Group integers into blocks of constant size
2. Find `b` such that all integers in the block are stricly smaller than 2^b
3. Concatenate `b` least significant bits of each integer

Bit packing example 1

Number	Binary representation (`b = 3`)
4	100
7	111
3	011
2	010

4 integers → 12 bits

Bit packing example 2

Number	Binary representation (`b = 4`)
4	100
9	1001
3	011
2	010

4 integers → 16 bits

Bit packing at the speed of light

- SIMD instructions
- Operate on 4 integers at a time
- 4 billion integers/second!

Anatomy of a search index

- Inverted index
    - Posting lists
    - Term dictionnary

Term dictionnary

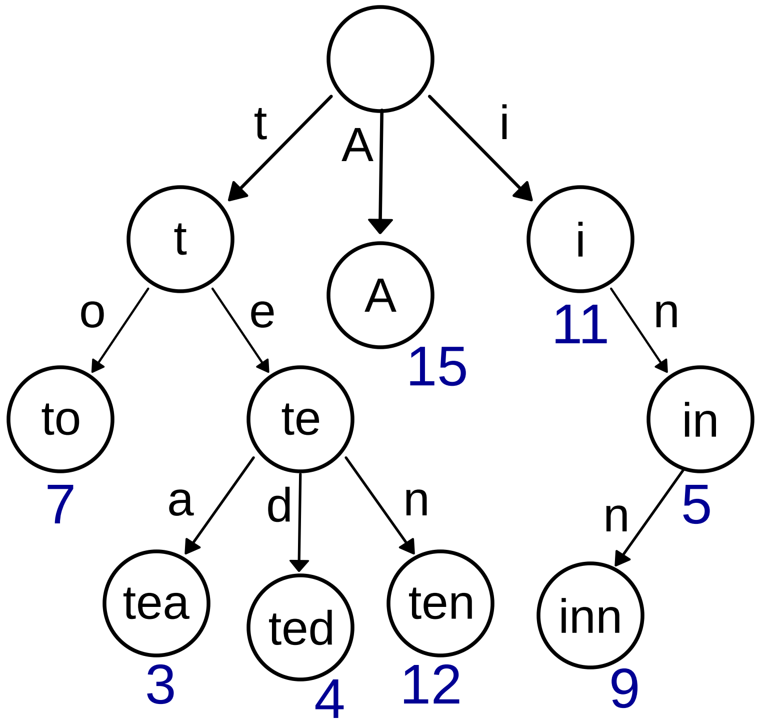
- A mapping from term to an offset in the postings file
  - `&[u8]` → `TermInfo`

```
//
// Code sample extracted from `tantivy/src/postings/term_info.rs`.
//
pub struct TermInfo {
    /// Byte range of the posting list within the postings (`.idx`) file.
    pub postings_range: Range<usize>,
}
```

Term dictionnary implementation

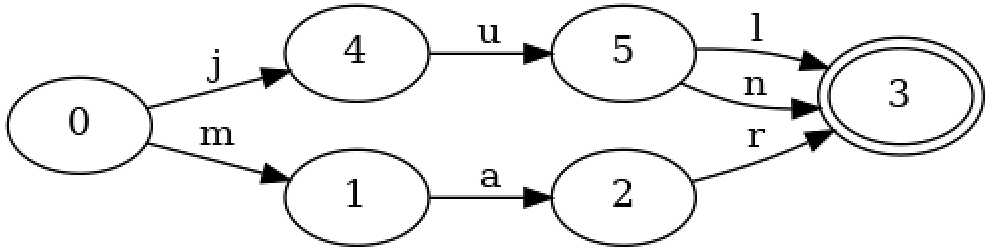
- Hashmap family?
  - fast lookups 😃
  - one or two random I/O accesses 😃
- Trie family?
  - more expensive lookups 😢
  - more random I/O accesses 😢
  - sorted range of keys 😃
  - compress well 😃
  - intersection with automaton (Levenshtein, regex) 😃

Trie



Finite state transducer (FST)

- More compact than a trie
- More CPU intensive
- Supports same operations as a trie
- Based on the fst crate from @BurntSushi (author of ripgrep)



Anatomy of an invertex index

- Inverted index
  - Posting lists
  - Term dictionary
  - Sufficient?

Phrase search

	`washington state`	`"washington state"`
Washington is a US state.	✓	
Washington State University is in Pullman.	✓	✓

Positional indexing

- Record the position of each term within the document
- Enable phrase and proximity search
- Increase substantially the size of the index

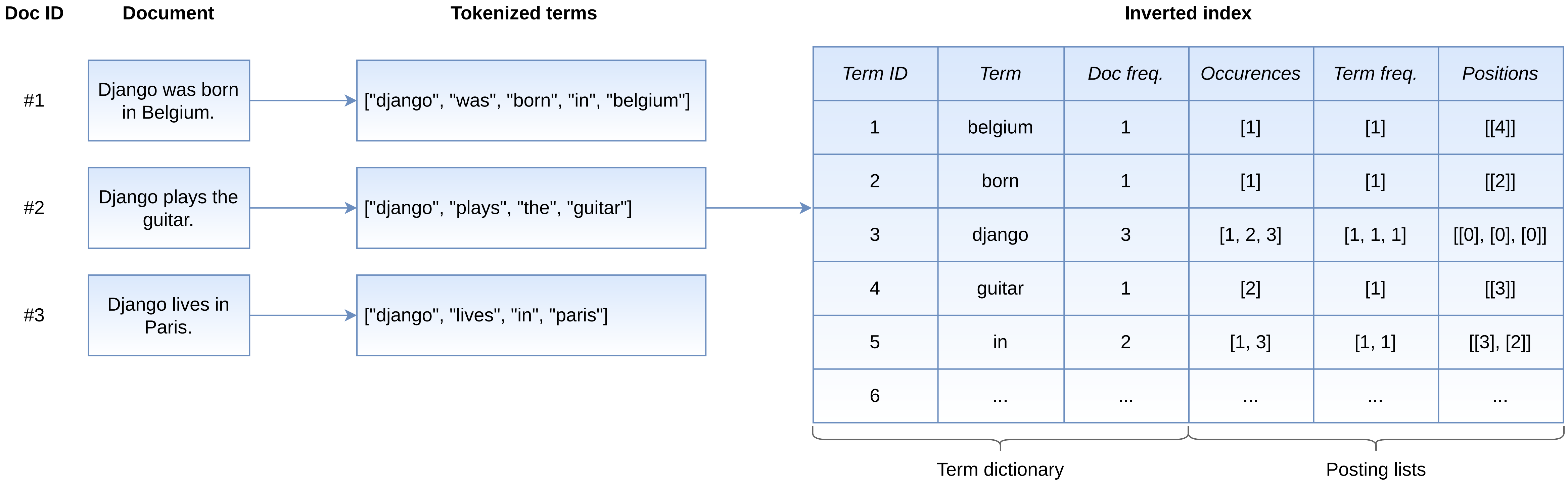
Scoring

- Probabilistic scoring functions (tf–idf, BM25)

Probabilistic scoring functions

- Term frequency
  - # occurences of term in document
  - stored in posting lists
- Document frequency
  - # documents where term occurs
  - stored in term dictionnary

Inverted index with frequencies and positions



Doc ID	Document	Tokenized terms
#1	Django was born in Belgium.	["django", "was", "born", "in", "belgium"]
#2	Django plays the guitar.	["django", "plays", "the", "guitar"]
#3	Django lives in Paris.	["django", "lives", "in", "paris"]

Inverted index

Term ID	*Term*	*Doc freq.*	*Occurences*	*Term freq.*	*Positions*
1	belgium	1	[1]	[1]	[[4]]
2	born	1	[1]	[1]	[[2]]
3	django	3	[1, 2, 3]	[1, 1, 1]	[[0], [0], [0]]
4	guitar	1	[2]	[1]	[[3]]
5	in	2	[1, 3]	[1, 1]	[[3], [2]]
6	...	...	...	...	...

Dictionary with doc frequencies and positions

```
//
// Code sample extracted from `tantivy/src/postings/term_info.rs`.
//
pub struct TermInfo {
   /// Number of documents in the segment containing the term.
   pub doc_freq: u32,

   /// Byte range of the posting list within the postings (`.idx`) file.
   pub postings_range: Range<usize>,

   /// Byte range of the positions of this term within the positions (`.pos`) file.
   pub positions_range: Range<usize>,
}
```

Incremental indexing

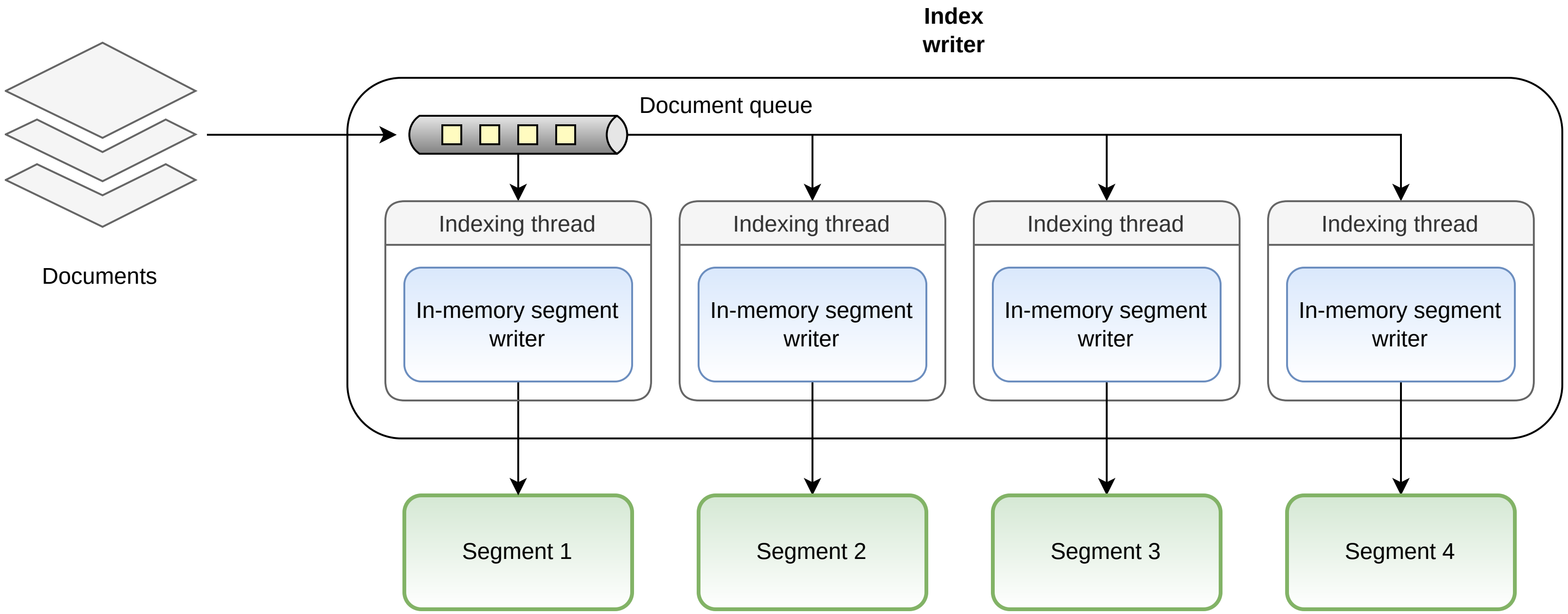
- Inverted index is a very a compact data structure not designed for in-place updates

Segments

- User allocates a RAM budget (e.g. 500MB)
- Append new documents to in-memory buffer
- When the budget is reached, serialize documents to disk in the index format

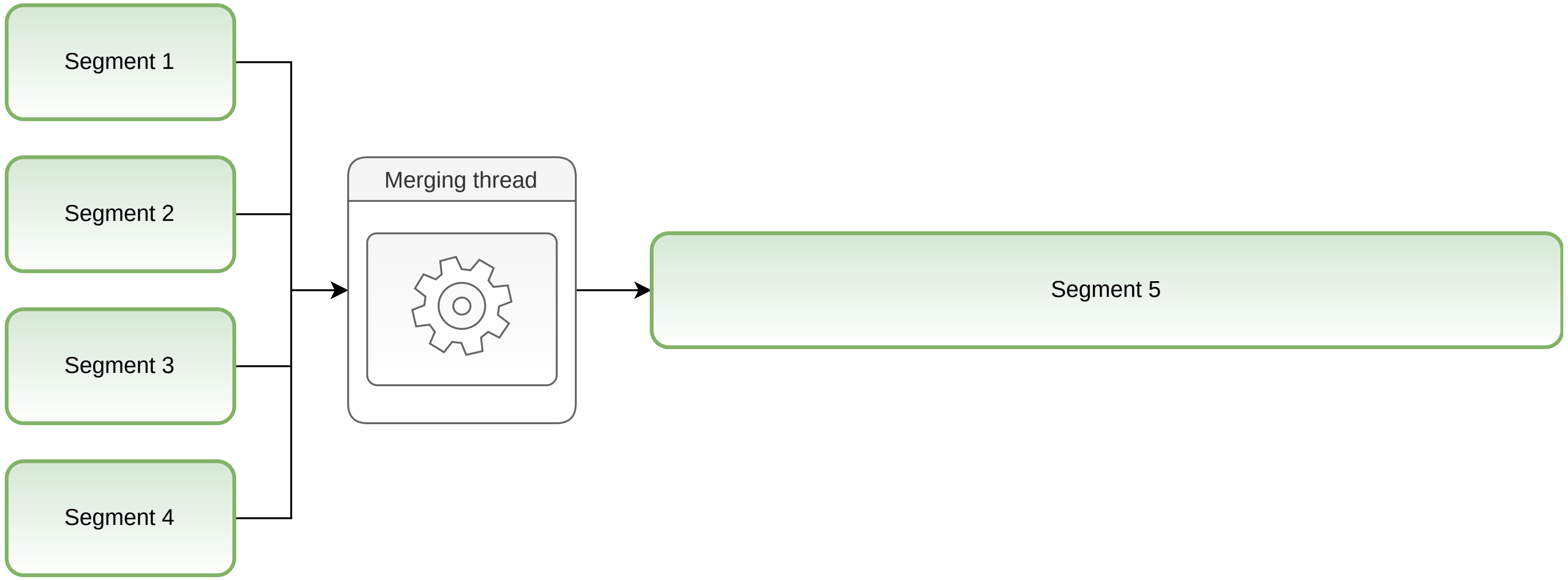
→ This **small, immutable, and independent** index is called a **segment.**
→ An index is actually a **collection of segments.**

Multithreaded index writer



Segment merges

- Merge multiple segments into a larger one
- Costly operation
- Trade-off between time to search and write amplification



Let's index Wikipedia!

Indexing Wikipedia

- 5M+ Wikipedia articles (JSON)
- 8.9 GB (uncompressed)

```
{
  "url": "https://en.wikipedia.org/wiki?curid=48689927",
  "title": "The Princess and the Clown",
  "body": "The Princess and the Clown (French: La princesse aux clowns) is a 1924 French silent film directed by André Hugon and starring Huguette Duflos, Charles de Rochefort and Magda Roche."
}
```

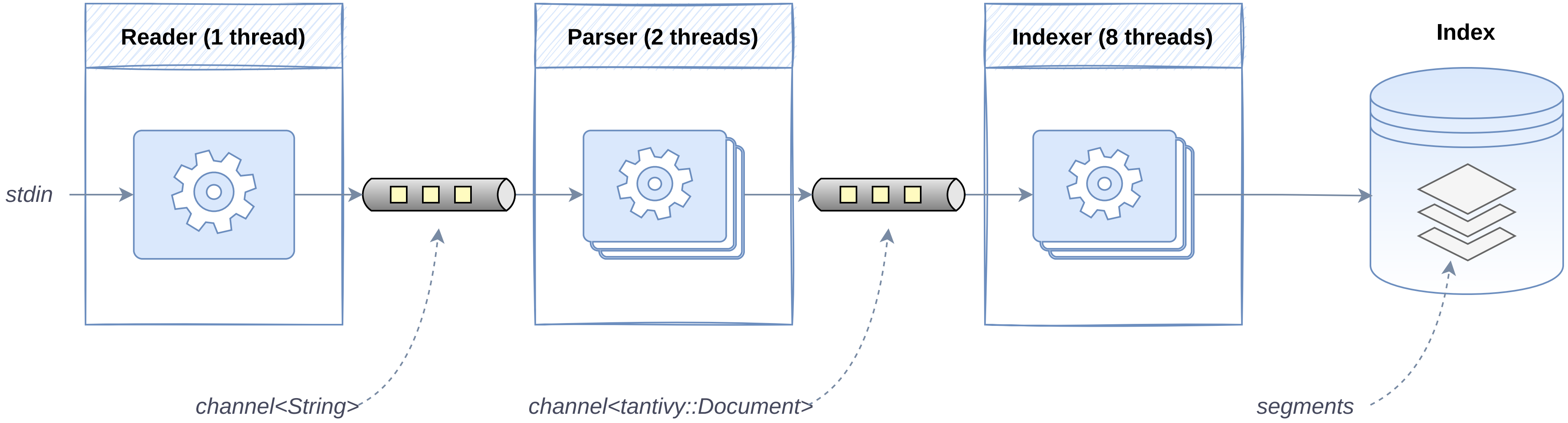
Indexing Wikipedia

1. `create_index()`

Indexing Wikipedia

1. `create_index()`
2. `append_docs()`

Indexing pipeline



Indexing Wikipedia

1. `create_index()`
2. `append_docs()`
3. `search_index()`

Indexing Wikipedia

1. `create_index()`
2. `append_docs()`
3. `search_index()`
4. `merge_segments()`

Indexing Wikipedia

1. `create_index()`
2. `append_docs()`
3. `search_index()`
4. `merge_segments()`
5. `run_gc()`

Common gotchas ⚠️

- Committing too often
- Acknowledging writes before committing
- Calling sync APIs in async runtime

Join the community!

- GitHub
- Discord

Questions?

Feedback?

adrien@quickwit.io