RAG FUNDAMENTALS 03

What is Chunking?

When you build a search system that uses AI to answer questions from documents, you need to break those documents into smaller pieces called "chunks", before uploading them into a vector store / knowledge base. The AI then searches through these chunks to find relevant information

Why Chunking Matters:

- **How you split your documents dramatically affects how well your system finds the right answers.**
- LLMs have limited context windows
- Retrieval can find/return only most relevant pieces of documents
- Gives LLM focused information without overwhelming with irrelevant context

---

The Example Text (Used in All Examples)

None

```
# The Kepler Space Telescope

The Kepler space telescope was a space observatory launched by NASA to discover
Earth-size planets orbiting other stars. It was named after astronomer Johannes Kepler.

## Mission Overview

Kepler's sole instrument was a photometer that continually monitored the brightness
of approximately 150,000 main sequence stars. It looked for periodic dimming,
known as transits, caused by planets crossing in front of their host star.

## Major Discoveries

During its nine-year mission, Kepler observed 530,506 stars and confirmed 2,662
planets. A key finding was that 20-50% of stars visible to the naked eye likely
have small, rocky planets similar in size to Earth.

## End of Operations

The mission ended in 2018 when the spacecraft ran out of fuel. The telescope was
deactivated, but its data continues to yield new discoveries.
```

Strategy 1: Fixed Size Splitting

Technical name: Recursive Character Splitter / Fixed Token Count

How it works: Cut the text every [N] words (or characters), no matter what.

Example Output:

Chunk	Content	Problem
1	"The Kepler space telescope was a space observatory... ## Mission Overview Kepler's sole instrument was a"	❌ Cuts mid-sentence
2	"photometer that continually monitored the brightness... caused by planets crossing in"	❌ Starts and ends mid-sentence
3	"front of their host star. ## Major Discoveries During its nine-year mission..."	❌ Mixes unrelated sections

Best for: When speed matters more than quality, or you have a very tight budget.

Strategy 2: Header-Based Splitting

Technical name: Markdown Header Chunking

How it works: Looks for headings (#, ##, etc.) and keeps everything under each heading together.

Example Output:

Chunk	Content
1	**# The Kepler Space Telescope** — "The Kepler space telescope was a space observatory... named after Johannes Kepler."
2	**## Mission Overview** — "Kepler's sole instrument was a photometer... planets crossing in front of their host star."
3	**## Major Discoveries** — "During its nine-year mission, Kepler observed 530,506 stars..."
4	**## End of Operations** — "The mission ended in 2018 when the spacecraft ran out of fuel..."

✅ **Perfect!** Each section stays intact.
❌ **Limitation:** Doesn't work if your document has no headers (like chat transcripts or emails).

Strategy 3: Meaning-Based Splitting

Technical name: Semantic Chunking

How it works: AI measures how "similar" each sentence is to the next. When the topic changes significantly (e.g., from "how it works" to "what it discovered"), it creates a new chunk.

Example Output:

Chunk	**Why Grouped Together**
"The Kepler space telescope was a space observatory... It looked for periodic dimming, known as transits."	AI sees "telescope," "instrument," and "monitor" as related concepts
"During its nine-year mission, Kepler observed 530,506 stars and confirmed 2,662 planets..."	Numbers and discoveries = new topic
"The mission ended in 2018 when the spacecraft ran out of fuel..."	"End," "deactivated" = termination topic

✅ **Benefit:** Groups ideas together even if the document has no headers.

Strategy 4: AI-Decided Splitting

Technical name: LLM-Based Chunking / Agentic Chunking

How it works: The AI reads the entire input text and decides the best places to split based on logic and flow.

> ⚠️ **Important: Why NOT to ask AI to output the full text with split markers**
>
> - Costly (duplicates all input tokens as output)
> - Slow (massive output generation)
> - Hallucination risk (AI may alter the original text)
> - Output token window limitations

The Correct 3-Phase Process:

PHASE 1: Pre-Split with Index Tags (No AI needed)

Split text into small fixed-size pieces (~50 tokens) and add numbered tags:

None

```
<chunk_0>The Kepler space telescope was a space observatory launched by NASA to
discover Earth-size planets orbiting other stars.</chunk_0>

<chunk_1>It was named after astronomer Johannes Kepler. Kepler's sole instrument
was a photometer that</chunk_1>

<chunk_2>continually monitored the brightness of approximately 150,000 main
sequence stars in a fixed field of view.</chunk_2>

<chunk_3>It looked for periodic dimming, known as transits, caused by planets
crossing in front of their host star.</chunk_3>

<chunk_4>During its nine-year mission, Kepler observed 530,506 stars and
confirmed 2,662 planets.</chunk_4>

<chunk_5>A key finding was that 20-50% of stars visible to the naked eye likely
have small, rocky planets similar in size to Earth.</chunk_5>

<chunk_6>The mission ended in 2018 when the spacecraft ran out of fuel.</chunk_6>

<chunk_7>The telescope was deactivated, but its data continues to yield new
discoveries.</chunk_7>
```

PHASE 2: AI Decision (Minimal output — just numbers!)

Ask AI: *"Return ONLY the chunk indices where semantic splits should occur."*

AI Response (entire output):

```
None
split_after: 0, 3, 5, 7
```

✅ AI outputs only ~10 tokens, not the entire document!

PHASE 3: Programmatic Merge (No AI needed)

Based on `split_after: 0, 3, 5, 7`, merge chunks:

Final Chunk	Source	Content
1	chunk_0	"The Kepler space telescope was a space observatory launched by NASA to discover Earth-size planets orbiting other stars."
2	chunks 1-3	"It was named after astronomer Johannes Kepler. Kepler's sole instrument was a photometer that continually monitored the brightness of approximately 150,000 main sequence stars... planets crossing in front of their host star."
3	chunks 4-5	"During its nine-year mission, Kepler observed 530,506 stars and confirmed 2,662 planets. A key finding was that 20-50% of stars visible to the naked eye likely have small, rocky planets..."
4	chunks 6-7	"The mission ended in 2018 when the spacecraft ran out of fuel. The telescope was deactivated, but its data continues to yield new discoveries."

✅ **Result:** AI decided semantically optimal split points, original text preserved exactly, low token cost.

✅ **Best for:** Messy documents, transcripts, or text without clear structure.

💡 **Pro Tip: Combining with Summarization**

AI-Decided chunking may create chunks that **exceed your vector store's maximum chunk size** (e.g., 8,000 tokens). When this happens, you can combine it with **Summarization**:

1. AI decides optimal semantic boundaries → creates logical chunks
2. If a chunk is too large → apply Summarization to condense it
3. Result: Semantically coherent chunks that fit within size limits

This hybrid approach gives you the **best of both strategies**: intelligent splitting + guaranteed size compliance.

Strategy 5: Summarization / Structured Output

Technical name: Structured Output / Q&A Format / Propositional Indexing

How it works: Instead of keeping the original text, extract pure facts as question-answer pairs or other structured formats (JSON, key-value pairs).

Example Output:

Question	Answer
What was the Kepler Space Telescope?	A NASA observatory designed to discover Earth-size planets.
What instrument did Kepler use?	A photometer that monitored star brightness.
How many planets did Kepler confirm?	2,662 planets.
How many stars did Kepler observe?	530,506 stars.
Why did Kepler's mission end?	The spacecraft ran out of fuel in 2018.

✅ **Best for:** FAQ systems, databases, when users ask specific questions.
❌ **Downside:** Loses the story and context between facts.

Enhancement Strategies (Add to Any Chunking Method Above)

These enhancements can be **combined with any chunking strategy** to boost search quality:

Enhancement	What It Does	Example	Cost	Quality Boost
Add Document Summary	Prepends document title, topic, and context to each chunk so it stands alone	"[Source: NASA Kepler Telescope Report, 2009-2018, exoplanet detection] The mission ended in 2018..."	💲 💲 Medium	⭐⭐⭐ High
Add Search Keywords	Prepends likely search queries and keywords before each chunk	"kepler fuel end mission 2018 shutdown why did kepler stop | The mission ended in 2018..."	💲 💲 💲 High	⭐⭐⭐⭐ Very High
Both Combined	Maximum search accuracy — chunks have context + optimized keywords	"[Source: Kepler Report] kepler mission end fuel 2018 | The mission ended in 2018..."	💲 💲 💲 High	⭐⭐⭐⭐⭐ Elite

Why Use Enhancements?

The Problem: A chunk saying "The mission ended in 2018" is useless if the AI doesn't know *which* mission.

Before (Ambiguous chunk):

> "The mission ended in 2018 when the spacecraft ran out of fuel."

After (With Document Summary):

> **[Source: NASA Kepler Space Telescope mission (2009-2018), exoplanet detection]**
> "The mission ended in 2018 when the spacecraft ran out of fuel."

✅ Now a search for "fuel" or "Kepler" returns a chunk that makes sense on its own and also has higher RAG accuracy / bigger chance to retrieve the chunk from the vector store.

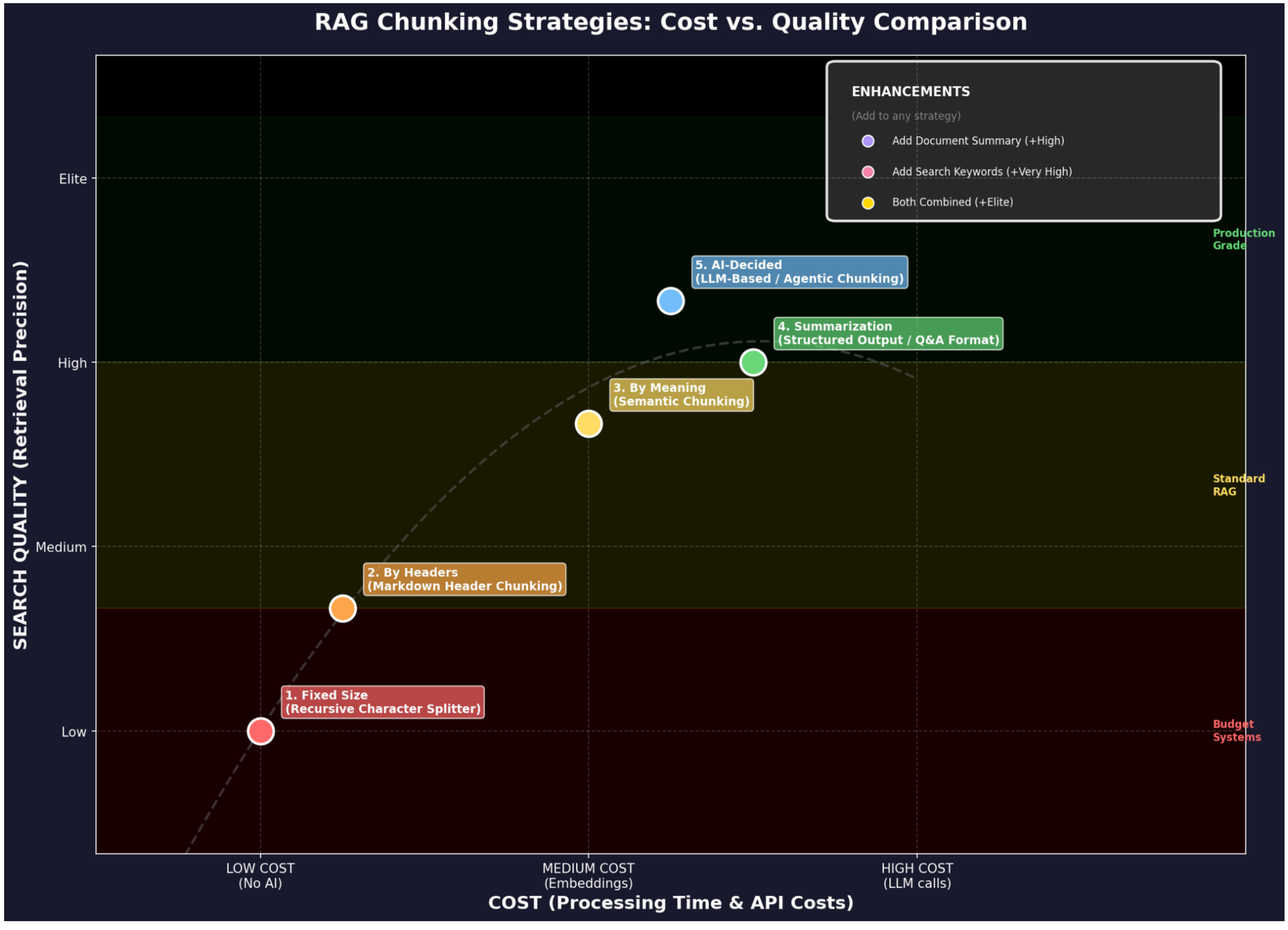
Chunking Strategies Comparison

	Strategy	Technical Name	What It Does	✅ Pros	❌ Cons	Cost	Retrieval Quality & LLM Generation Accuracy
1	**Fixed Size**	Recursive Character Splitter / Fixed Token Count	Cuts text every X characters, ignoring meaning	Fast, simple, predictable	Often cuts sentences in half; loses context	💲 Low	⭐ Low
2	**By Headers**	Markdown Header Chunking	Splits at document headings #, ##, etc. (like chapter titles)	Keeps topics and sections together naturally	Only works if the document / URL is properly structured and has headers. Length of information under each header must comply with vector store max chunk size.	💲 Low	⭐⭐ Medium
3	**By Meaning**	Semantic Chunking	AI detects when the topic changes and splits there	Smart grouping by ideas	Slower; needs a lot of tuning to work well. Length of semantic group must comply with vector store max chunk size.	💲 💲 Medium	⭐⭐⭐ High
4	**AI-Decided**	LLM-Based / Agentic Chunking	AI reads entire text and decides where logical breaks are	Handles messy text /structure perfectly well	Requires AI processing. Length of information in each calculated chunk must comply with vector store max chunk size.	💲 💲 Medium	⭐⭐⭐⭐ Very High
5	**Summarization**	Structured Output / Q&A Format	Converts text into question-answer pairs or structured data	Perfect for clear and concise types of answers / data. Option to influence summary length -> perfectly complies with	Loses the story/narrative; can hallucinate. Not ideal for long lists of information into single pairs.	💲 💲 💲 High	⭐⭐⭐⭐ Very High

	Strategy	Technical Name	What It Does	✅ Pros	❌ Cons	Cost	Retrieval Quality & LLM Generation Accuracy
				vector store max chunk size.			

Key Takeaways

1. **No single strategy is best** — Choose based on your document type and budget.
2. **Strategies can be combined** — Most production systems use 2-3 strategies together.
3. **Start simple, improve later** — Begin with Header-Based or Fixed Size, then upgrade.
4. **Test with real questions** — The best strategy depends on how your users actually search.
5. **Context matters** — Adding source summaries or keywords to chunks often gives the biggest quality boost.


RAG Chunking Strategies: Cost vs. Quality Comparison
ENHANCEMENTS
(Add to any strategy)
Add Document Summary (+High)
Add Search Keywords (+Very High)
Both Combined (+Elite)
5. AI-Decided
(LLM-Based / Agentic Chunking)
4. Summarization
(Structured Output / Q&A Format)
3. By Meaning
(Semantic Chunking)
2. By Headers
(Markdown Header Chunking)
1. Fixed Size
(Recursive Character Splitter)
Production Grade
Standard RAG
Budget Systems
Elite
High
Medium
Low
SEARCH QUALITY (Retrieval Precision)
LOW COST
(No AI)
MEDIUM COST
(Embeddings)
HIGH COST
(LLM calls)
COST (Processing Time & API Costs)