AI Periodic Table Explained: Mapping LLMs, RAG & AI Agent Frameworks

https://www.youtube.com/watch?v=ESBMgZHzfG0

This video introduces an "AI Periodic Table" (0:30) as a way to structure and understand the complex world of AI. It organizes AI concepts into families (groups) and periods (rows), similar to the chemistry periodic table, to help decode AI architectures and products (0:40).

The table is broken down into:

Primitives (Row 1) (1:28):

Prompts (PR) (1:00): Reactive instructions given to an AI (1:54).

Embeddings (EM) (2:39): Numerical representations of meaning used for semantic search (2:50).

Large Language Models (LLMs - LG) (3:30): Stable foundational capabilities like ChatGPT or IBM Granite (3:53).

Compositions (Row 2) (4:56):

Function Calling (FC) (4:27): LLMs calling external tools or APIs (4:30).

Vector Databases (VX) (6:02): Data stores optimized for semantic search and storing embeddings (6:08).

Retrieval Augmented Generation (RAG - RG) (6:34): Orchestrates embeddings, vector databases, and LLMs to generate answers based on retrieved context (6:38).

Guardrails (GR) (7:28): Runtime safety filters and schema validation to ensure appropriate AI output (7:32).

Multi-modal Models (M) (7:57): LLMs that can process images, audio, and text (7:59).

Deployment (Row 3) (5:31):

Agents (AG) (5:16): Use think-act-observe loops to achieve goals by taking actions (5:20).

Fine-tuning (FT) (8:32): Adapting a base model by training it on specific data (8:34).

Frameworks (FW) (9:15): Platforms like LangChain that tie everything together to build and deploy AI systems (9:18).

Red Teaming (RT) (9:31): Adversarial testing to find vulnerabilities in AI systems (9:35).

Small Models (SM) (9:53): Distilled, specialized models that are fast and cheap (9:53).

Emerging (Row 4) (10:31):

Multi-agent Systems (MA) (10:05): Multiple AIs working together to solve complex problems (10:13).

Synthetic Data (SY) (10:56): Using AI to generate training data for AI (10:56).

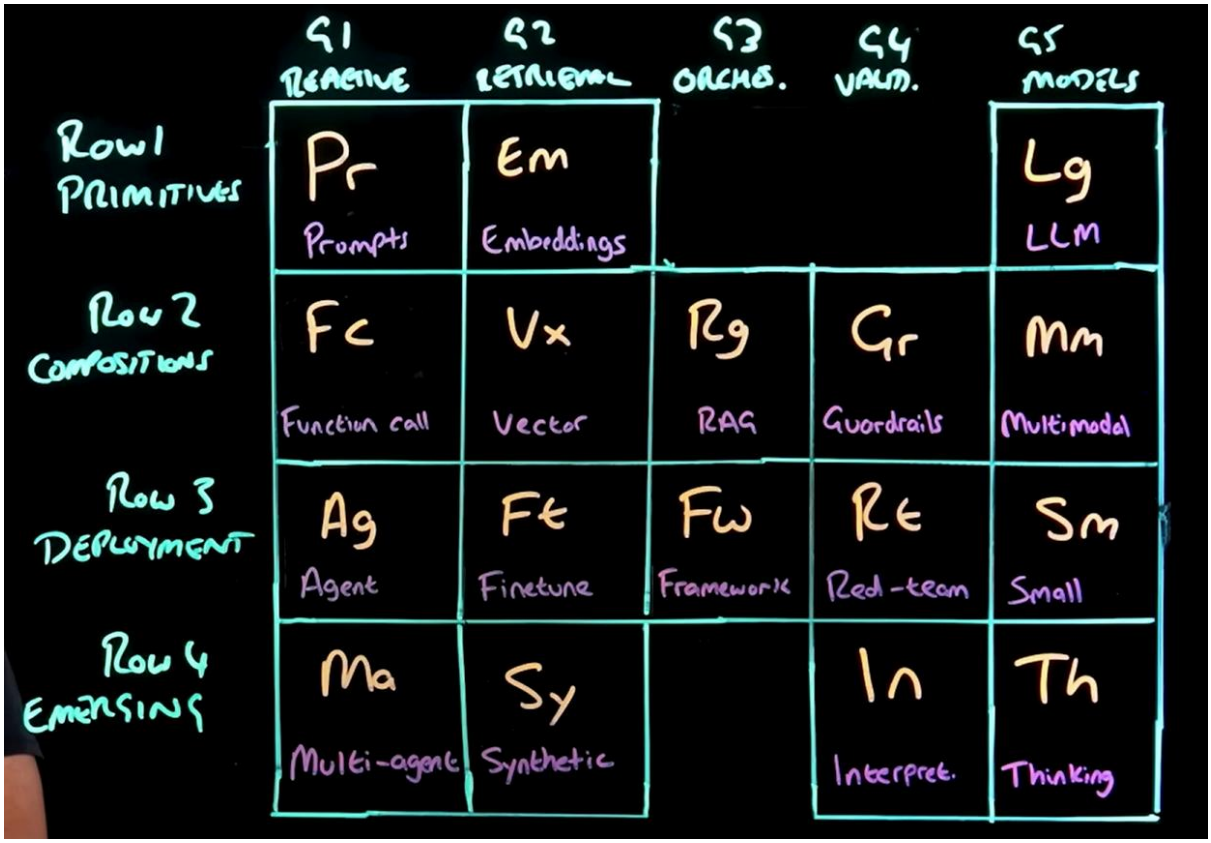
Interpretability (IN) (11:43): Understanding why a model does what it does (11:45).

Thinking Models (TH) (12:06): Models that spend time reasoning before answering, often with chain-of-thought built-in (12:08).

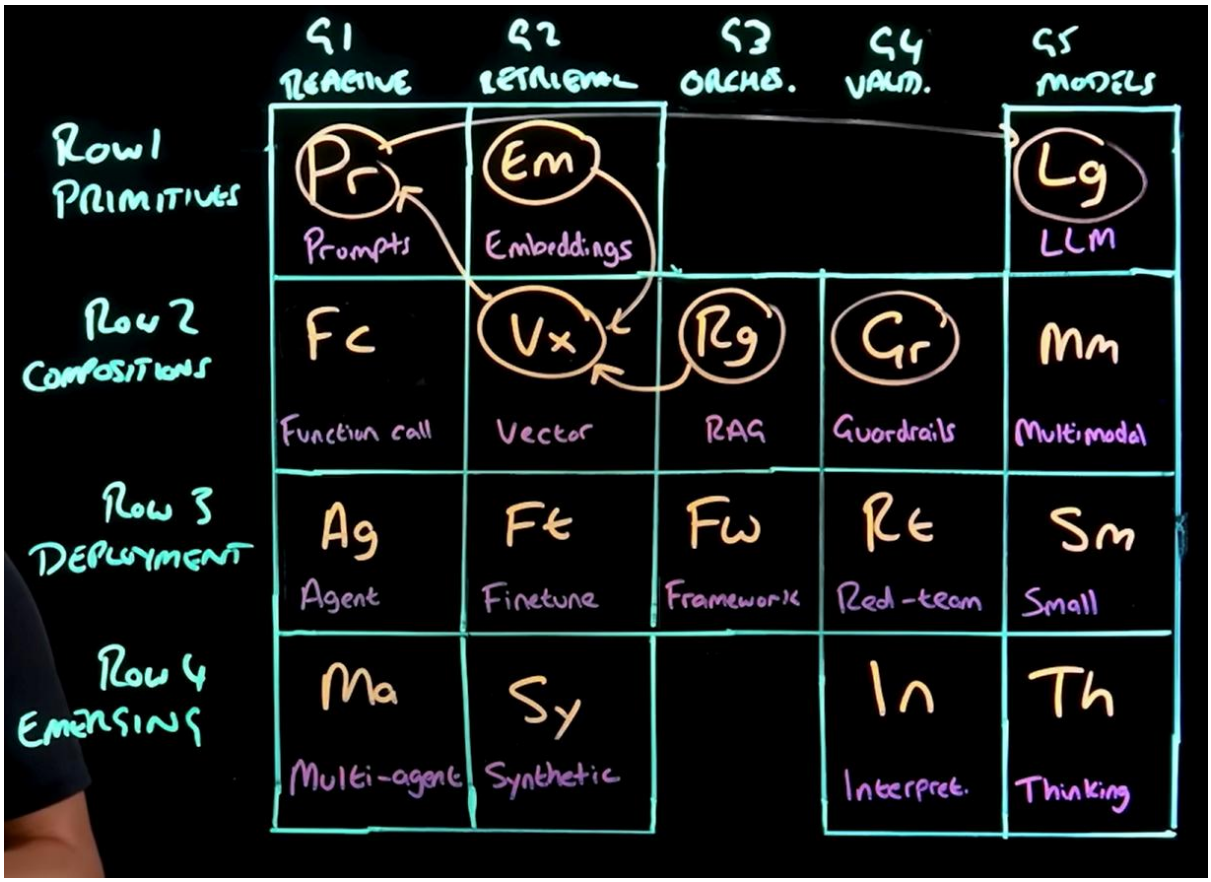
The video demonstrates how these elements combine in "reactions" (12:31) to build AI systems, such as:

A chatbot that knows company documentation (12:48) using embeddings, vector databases, RAG, prompts, LLMs, and guardrails (12:54-14:02).

An agentic loop (14:08) where an agent uses function calling within a framework to achieve a goal (14:11-15:23).

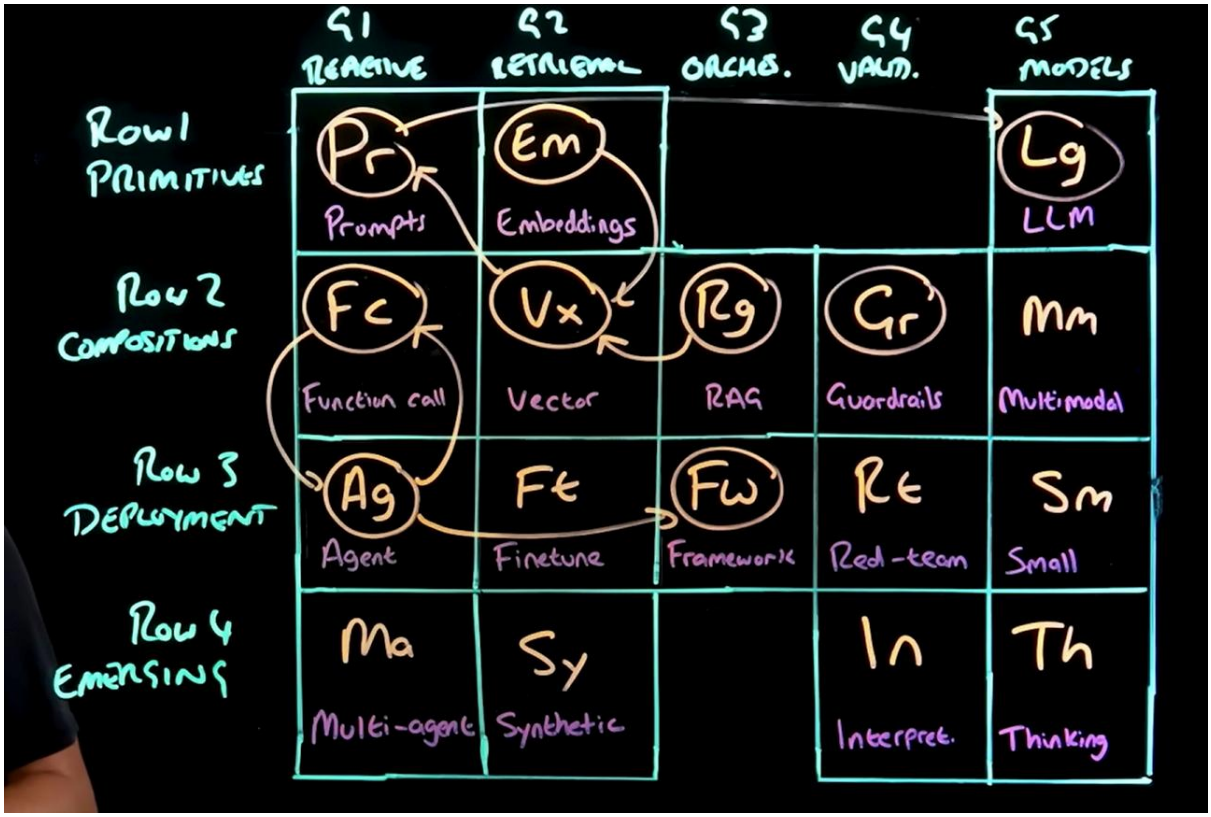


1st chain



2nd chain

Agentic Chain


C1 REACTIVE
C2 RETRIEVAL
C3 ORCHS.
C4 VALID.
C5 MODELS
Row 1 PRIMITIVES
Row 2 COMPOSITIONS
Row 3 DEPLOYMENT
Row 4 EMERGING
Pr
Prompts
Em
Embeddings
Lg
LLM
Fc
Function call
Vx
Vector
Rg
RAG
Gr
Guardrails
Mm
Multimodal
Ag
Agent
Ft
Finetune
Fw
Framework
Rt
Red-team
Sm
Small
Ma
Multi-agent
Sy
Synthetic
In
Interpret.
Th
Thinking

##

AGENTIC AI PERIODIC TABLE

Greg Coquillo
Product Leader at AWS

Attribute

G Goal	WM Working Memory	LT Latency	TR Trigger	AC Action	IS Internal State	CT Cost	HL HITL	ID Identity
E Environment	CF Control Flow	GR Guardrail	LM Long-Term Memory	T Tool	LP Loop	SE State Estimation	PRF Preferences	FR Forgetting

Building Block

FW Framework	CS Code Sandbox	CU Computer Use	IN Instructions	MCP Model Context Protocol	A2A Agent 2 Agent Protocol	V Vector DB	BU Browser Use	LLM Large Language Model
WS Web Search	API API Integration	RAG Retrieval-Augmented Generation	KB Knowledge Base	FN Function Calling	DB Database Access	AUTH Auth & Access	MQ Message Queue	

Characteristic

AU Autonomy	AL Alignment	SA Social Ability	RS Reliability/ Safety	RE Reactivity	MO Modularity	MM Multimodality	AD Adaptability	LE Learning Ability
FN Fine-Tuning Ready								

Activity

PL Planning	DM Decision Making	SI Self-Improvement	EX Executing	EH Error Handling	GF Getting Feedback	R Reasoning	EI Environment Interaction	VS Vision Perception
VC Voice Command	CN Conversation Loop	EXL Exploration						

Pattern

PL Planning	DM Decision Making	SI Self-Improvement	EX Executing	EH Error Handling	GF Getting Feedback	R Reasoning	EI Environment Interaction	VS Vision Perception

Ops

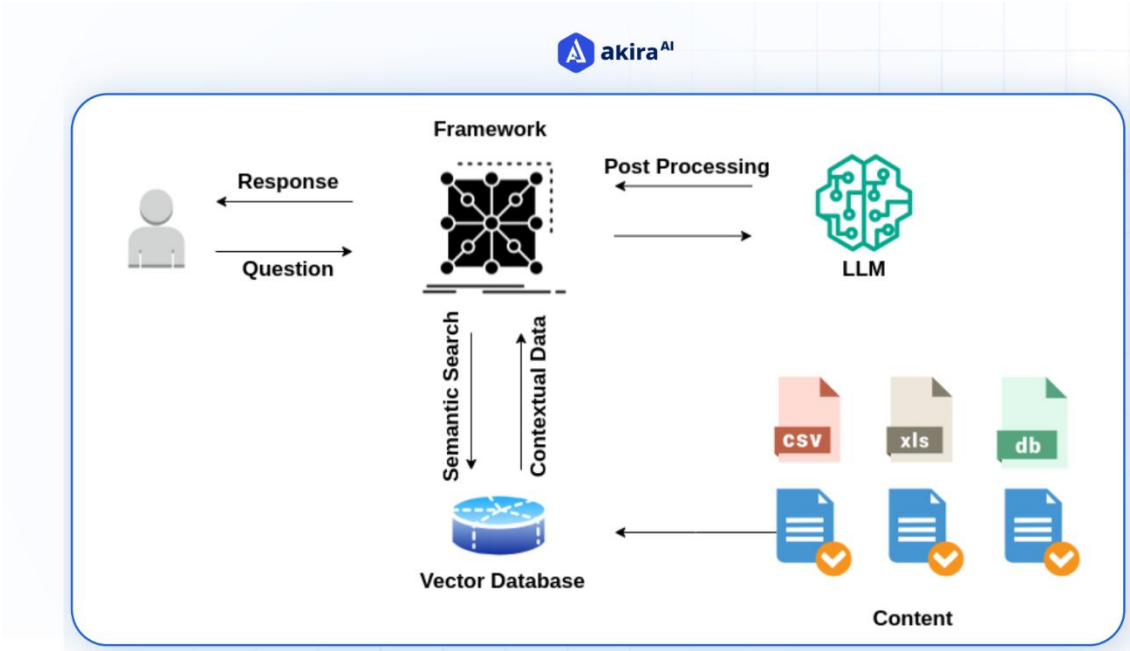
OR Orchestra -tion	RTU Ready To Use	SC Scaling	NC No Code	DP Deployment	LC Low Code	CO Cost Optimization	PC Pro Code	MON Monitoring
LOG Logging	CI Continuous Integration	CD Continuous Delivery						

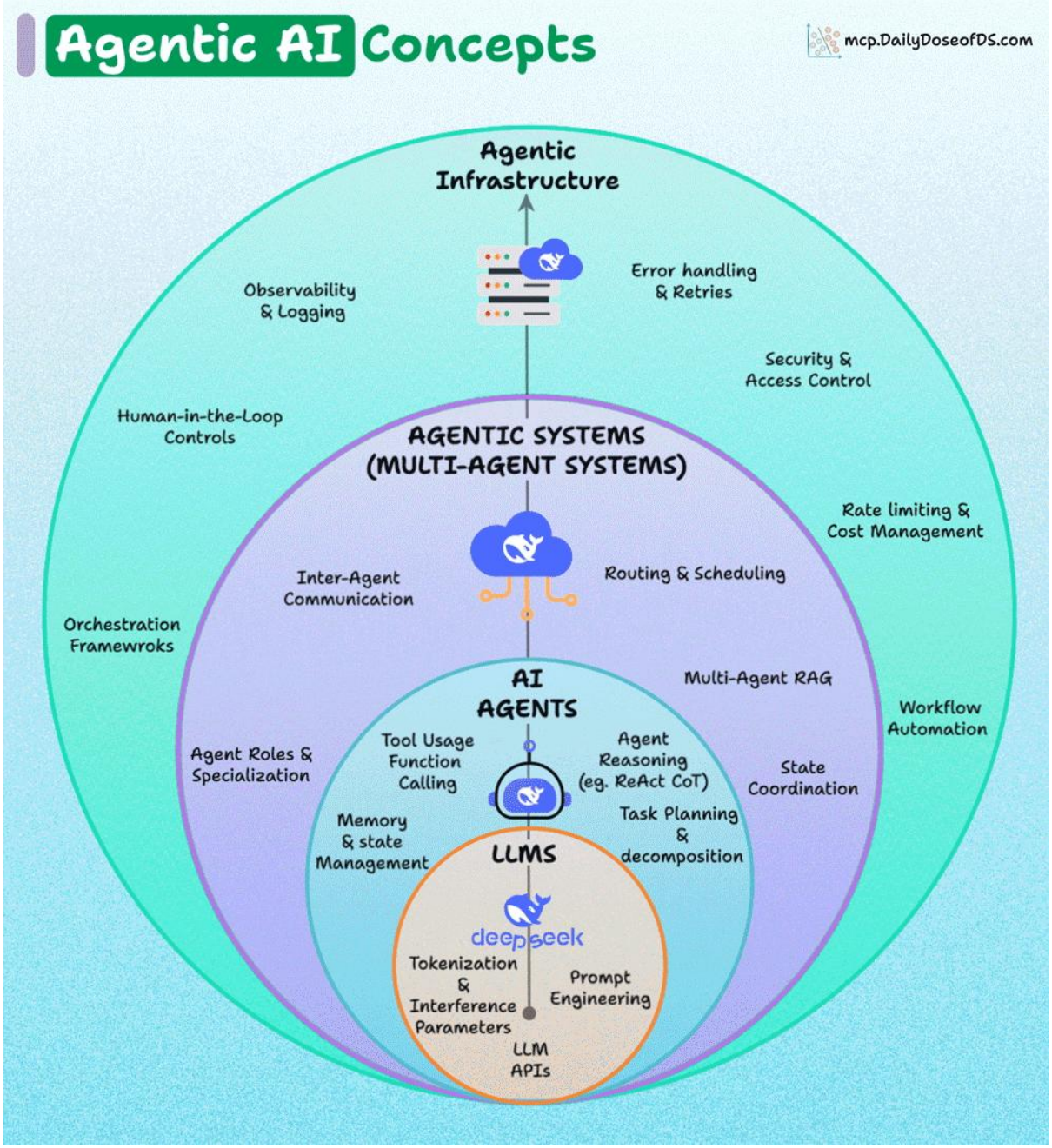
Interface

UI User Interface	CLI Command Line Interface	APIX API Exposure	INTG Integration Layer	UX User Experience

Evaluation

MET Metrics	LOGS Logs & Traces	OBS Observability	QA Quality Assurance	BEN Bench -marking


akira AI
Framework
Response
Question
Post Processing
LLM
Semantic Search
Contextual Data
csv
xls
db
Vector Database
Content



🧠 **AI Periodic Table Explained**

Mapping LLMs, RAG, and AI Agent Frameworks

Think of this diagram as a **systems map of modern AI**, similar to how the chemical periodic table organizes elements.
Here, **rows = maturity layers**, **columns = system capabilities**, and **each cell = a core AI building block**.

📐 **How to Read the Table**

🔹 **Columns (C1 → C5): Capability Evolution**

They represent **how intelligent and autonomous the system becomes** as you move right.

Column	Meaning	What it Enables
C1 – Reactive	Direct responses	Prompt → Answer
C2 – Retrieval	External memory	Knowledge grounding
C3 – Orchestration	Multi-step workflows	Tool + reasoning
C4 – Validation	Safety & control	Guardrails, evals
C5 – Models	Core intelligence	LLMs, multimodal

🔹 **Rows (R1 → R4): System Maturity**

They show **how production-grade and advanced the AI system is**.

Row	Meaning
R1 – Primitives	Atomic building blocks
R2 – Compositions	Combined capabilities
R3 – Deployment	Production systems
R4 – Emerging	Next-gen intelligence

🧩 **Row-by-Row Breakdown**

🟦 **Row 1: Primitives (Foundations)**

These are the **atoms of AI systems**.

Symbol	Name	What it Does
Pr	Prompts	Instructions to LLMs
Em	Embeddings	Text → vectors
Lg	LLM	Core reasoning engine

🔑 *Without these, nothing else exists.*

🟩 **Row 2: Compositions (Power Combos)**

Where real-world usefulness begins.

Symbol	Name	Purpose
Fc	Function Calling	Tool execution
Vx	Vector DB	Semantic search
Rg	RAG	Knowledge-augmented LLMs
Gr	Guardrails	Safety & policy
Mm	Multimodal	Text + image + audio

📌 *This is where most enterprise GenAI apps live today.*

🟧 **Row 3: Deployment (Production AI)**

Scalable, monitored, and autonomous.

Symbol	Name	Why it Matters
Ag	Agent	Autonomous decision-maker
Ft	Fine-tuning	Domain adaptation
Fw	Frameworks	LangChain / CrewAI
Re	Red-teaming	Security & robustness
Sm	Small Models	Cost & latency control

🚀 *This is LLMOps, MLOps, and AgentOps territory.*

🟥 **Row 4: Emerging (Next Frontier)**

Where AI starts behaving like a **system of minds**.

Symbol	Name	Meaning
Ma	Multi-agent	Teams of agents
Sy	Synthetic Data	Self-improving AI
In	Interpretability	Explainable reasoning
Th	Thinking	Chain-of-thought & planning

🧠 *This is where AGI-like behaviors begin to appear.*

🔗 **How It All Connects (Example Stack)**

Enterprise AI Agent Stack

LLM (Lg)

+ RAG (Rg)

+ Vector DB (Vx)

+ Function Calling (Fc)

+ Agent (Ag)

+ Guardrails (Gr)

+ Framework (Fw)

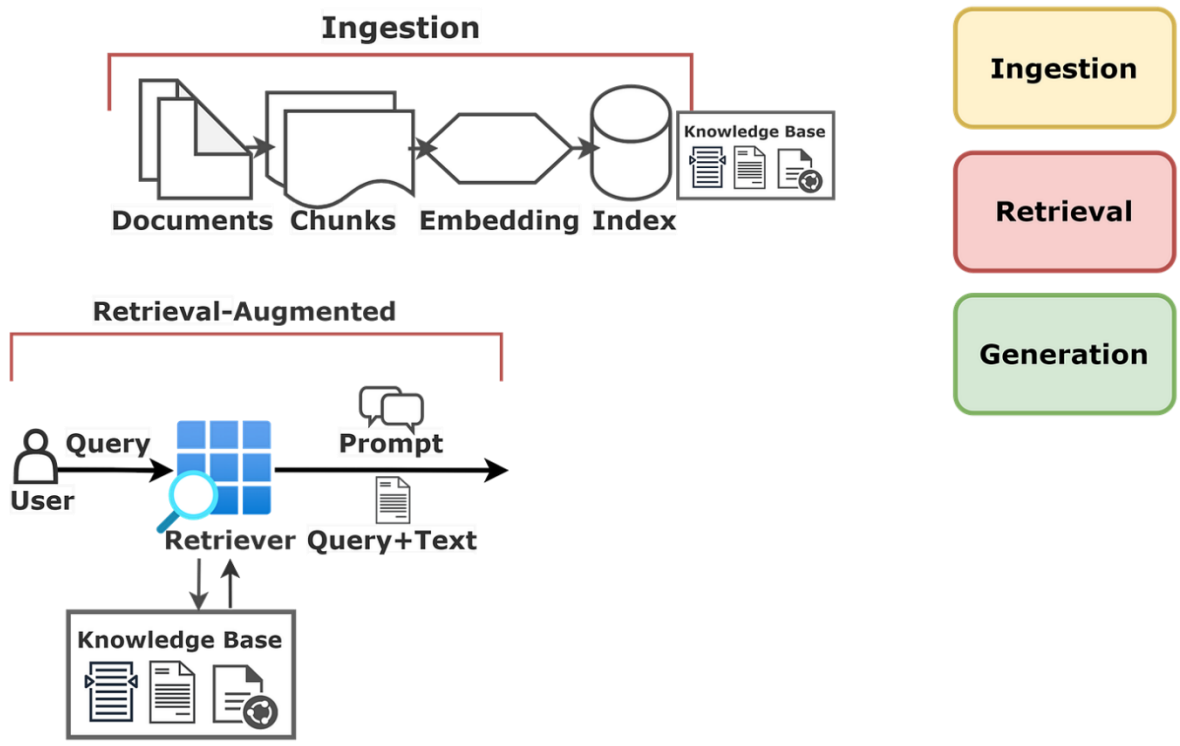
➡️ Result: **Autonomous, safe, production-ready AI**

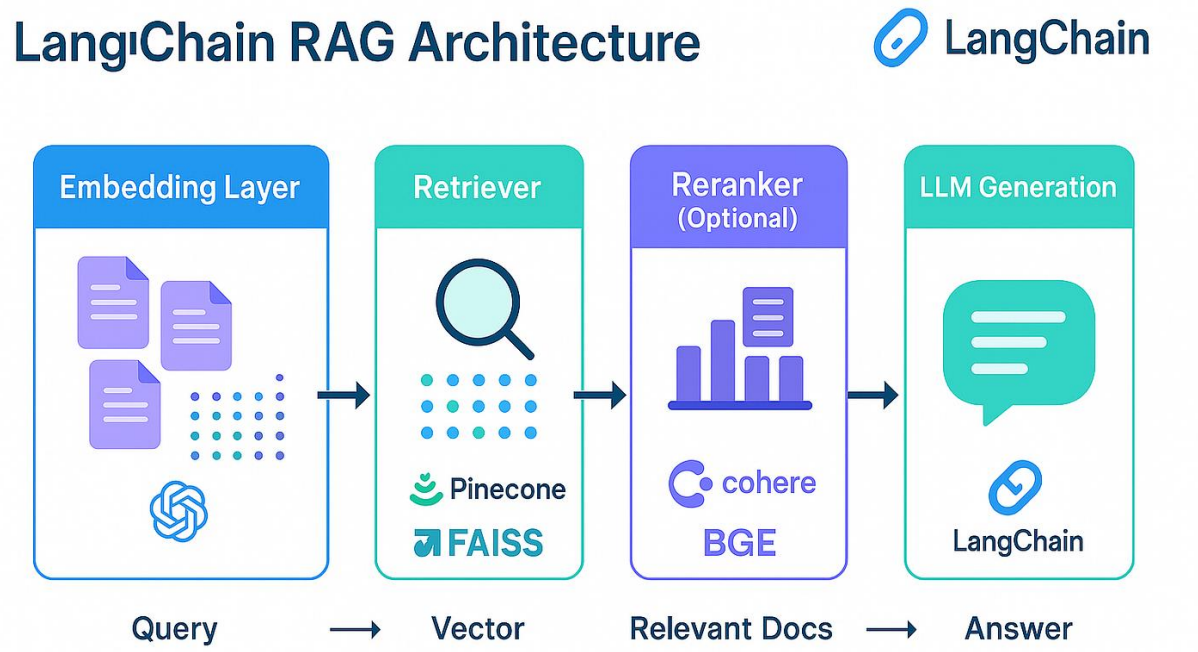
🎯 **Why This Table Matters**

- ✅ Helps **architect AI systems end-to-end**
- ✅ Clarifies **what skill belongs where** (Prompting vs Agent Design)
- ✅ Aligns well with **FAANG / product-company interviews**
- ✅ Perfect mental model for **GenAI, LLMOps, Agentic AI**

##

fist chain





🔗 **First Chain (Core RAG Flow) — Explained Simply**

When you say **“first chain”**, you’re referring to the **basic execution path** drawn with arrows in the diagram.
This is the **minimum viable intelligence chain** behind most RAG systems.

🧠 **The First Chain = Prompt → Retrieval → Grounded Answer**

1️⃣ **Pr — Prompt**

- User asks a question
- Example: *“Explain transformer architecture”*

📌 This is the **entry point** of the system.

2️⃣ **Em — Embeddings**

- The prompt is converted into a **vector representation**
- Captures semantic meaning, not keywords

📌 Enables semantic understanding instead of string matching.

3️⃣ **Vx — Vector Store**

- The embedding is compared against stored vectors
- Retrieves the **most relevant chunks** (documents, PDFs, logs, etc.)

📌 This is your **external memory**.

4️⃣ **Rg — RAG (Retrieval-Augmented Generation)**

- Retrieved context is injected back into the prompt
- Prompt becomes:

Question + Relevant Knowledge

📌 This **grounds the LLM** and reduces hallucination.

5️⃣ **Lg — LLM**

- The LLM generates the final answer
- Uses both:
    - User intent
    - Retrieved factual context

📌 This is where reasoning + language generation happens.

🧩 One-Line Flow (Interview Ready)

Prompt → Embedding → Vector Search → Context Injection (RAG) → LLM Response

---

🏗️ Why This Is Called the "First Chain"

- ✅ Smallest **production-useful** GenAI pipeline
- ✅ Foundation for:
  - Chatbots
  - Search assistants
  - Knowledge Q&A
- ✅ Everything else (Agents, Tools, Guardrails) **builds on top of this**

🔁 What Comes *After* the First Chain

Once this works, teams add:

- **Fc** → tool calling
- **Ag** → autonomous agents
- **Gr** → guardrails & validation
- **Fw** → orchestration frameworks
- **Ma** → multi-agent systems

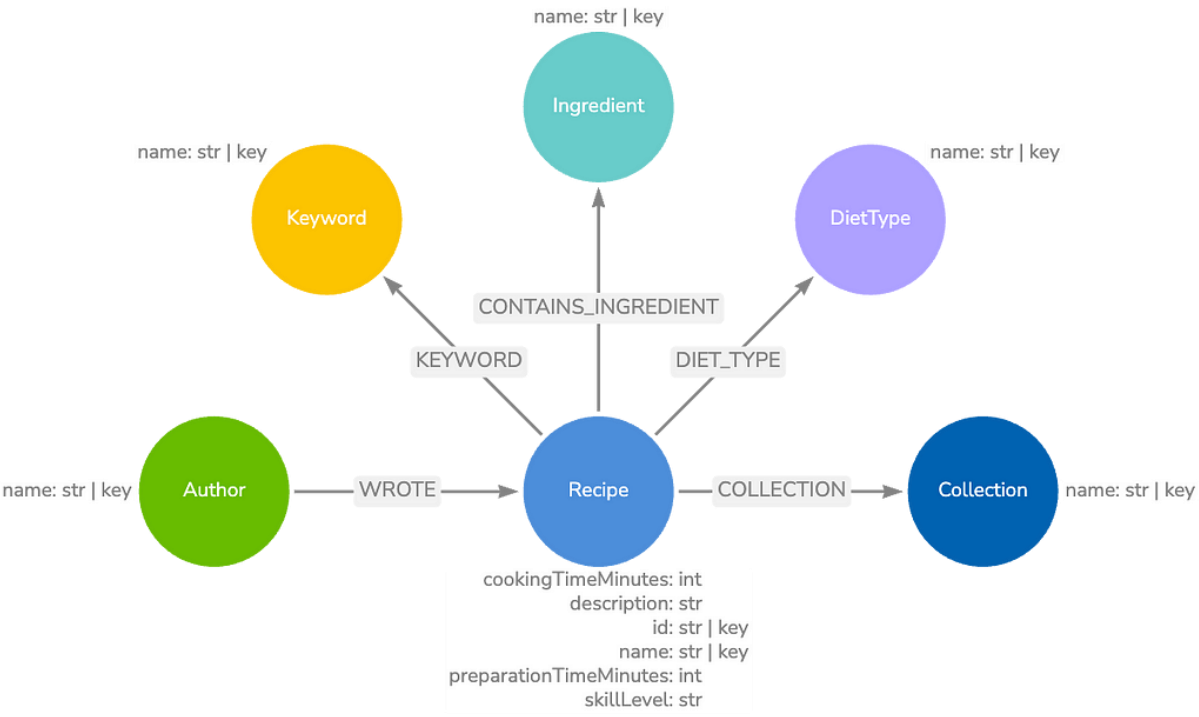
But **all of them still rely on this first chain**.

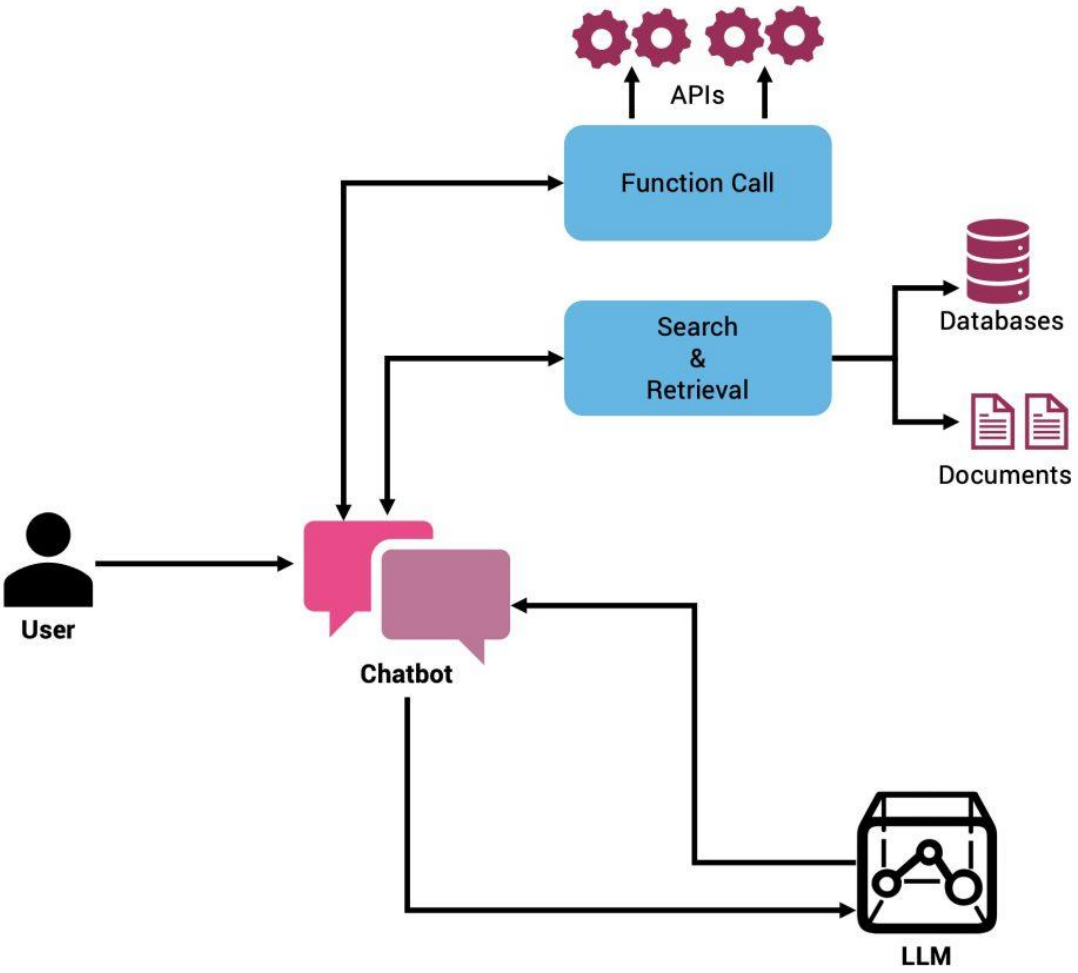
🎯 Line You Can Use

Designed and deployed a core RAG pipeline using prompt embeddings, vector similarity search, and LLM-based grounded generation as the foundational inference chain.

##

2nd chain Agentic loop -Ag,Fc .Fw


name: str | key
Ingredient
name: str | key
Keyword
name: str | key
DietType
CONTAINS_INGREDIENT
KEYWORD
DIET_TYPE
name: str | key
Author
WROTE
Recipe
COLLECTION
Collection
name: str | key
cookingTimeMinutes: int
description: str
id: str | key
name: str | key
preparationTimeMinutes: int
skillLevel: str


APIs
Function Call
Search
&
Retrieval
Databases
Documents
User
Chatbot
LLM

🔁 **Second Chain: Agentic Loop (Ag → Fc → Fw)**

This **2nd chain** builds on the *first RAG chain* and turns a smart chatbot into an **autonomous AI agent** that can **decide, act, observe, and repeat**.

Think of it as **reasoning + action + control loop**.

🧠 **What Makes It "Agentic"?**

An **agent** is not just answering once — it:

- Plans next steps
- Calls tools
- Evaluates results
- Repeats until the goal is achieved

🔗 **The Agentic Loop Explained Step-by-Step**

1️⃣ **Ag — Agent (Decision Maker)**

- Holds the **goal**
- Maintains **state & memory**
- Decides:
    - "Do I need a tool?"
    - "What should I do next?"

📌 This is the **brain + controller**.

2️⃣ Fc — Function Calling (Actions)

- Agent invokes tools:
  - APIs
  - Databases
  - Search
  - Code execution
  - RAG retriever
- Structured inputs & outputs (JSON)

📌 This is **how the agent interacts with the world**.

3️⃣ Fw — Framework (Orchestration Engine)

- Manages the loop:
  - Step ordering
  - Retries
  - Error handling
  - Tool routing
- Examples (conceptually):
  - Agent executor
  - Task planner
  - Workflow engine

📌 This is the **runtime** that keeps the agent alive and stable.

🔁 Full Agentic Loop Flow

User Goal

↓

Agent (Ag) → decides next action

↓

Function Call (Fc) → executes tool

↓

Observation / Result

↓

Framework (Fw) → feeds result back

↓

Agent (Ag) → reassesses & loops

➡️ Loop continues **until goal completion**.

🔁 **How This Extends the First Chain**

First Chain (RAG)	Second Chain (Agentic)
One-shot answer	Multi-step reasoning
Passive	Autonomous
No memory	Stateful
No actions	Tool execution

📌 **RAG answers questions**
📌 **Agents complete tasks**

🧩 **Real-World Example**

"Book a meeting with all stakeholders next week"

Agent flow:

1. Agent checks calendars (Fc)
2. Resolves conflicts (Ag reasoning)
3. Sends emails (Fc)
4. Confirms responses
5. Schedules meeting
6. Stops when done

✔️ No human in the loop.

🏗️ **Typical Production Stack**

LLM

+ RAG (knowledge)

+ Agent (Ag)

+ Function Calling (Fc)

+ Framework (Fw)

+ Guardrails (Gr)

🎯 **Ready Line**

Designed an agentic AI system using function-calling and orchestration frameworks, enabling autonomous multi-step task execution with stateful reasoning loops.

##

AGENTIC AI PERIODIC TABLE

Greg Coquillo
Product Leader at AWS

Attribute

G Goal	WM Working Memory	LT Latency	TR Trigger	AC Action	IS Internal State	CT Cost	HL HITL	ID Identity
E Environment	CF Control Flow	GR Guardrail	LM Long-Term Memory	T Tool	LP Loop	SE State Estimation	PRF Preferences	FR Forgetting

Building Block

FW Framework	CS Code Sandbox	CU Computer Use	IN Instructions	MCP Model Context Protocol	A2A Agent 2 Agent Protocol	V Vector DB	BU Browser Use	LLM Large Language Model
WS Web Search	API API Integration	RAG Retrieval-Augmented Generation	KB Knowledge Base	FN Function Calling	DB Database Access	AUTH Auth & Access	MQ Message Queue	

Characteristic

AU Autonomy	AL Alignment	SA Social Ability	RS Reliability/ Safety	RE Reactivity	MO Modularity	MM Multimodality	AD Adaptability	LE Learning Ability
FN Fine-Tuning Ready								

Activity

PL Planning	DM Decision Making	SI Self-Improvement	EX Executing	EH Error Handling	GF Getting Feedback	R Reasoning	EI Environment Interaction	VS Vision Perception
VC Voice Command	CN Conversation Loop	EXL Exploration						

Pattern

PL Planning	DM Decision Making	SI Self-Improvement	EX Executing	EH Error Handling	GF Getting Feedback	R Reasoning	EI Environment Interaction	VS Vision Perception

Ops

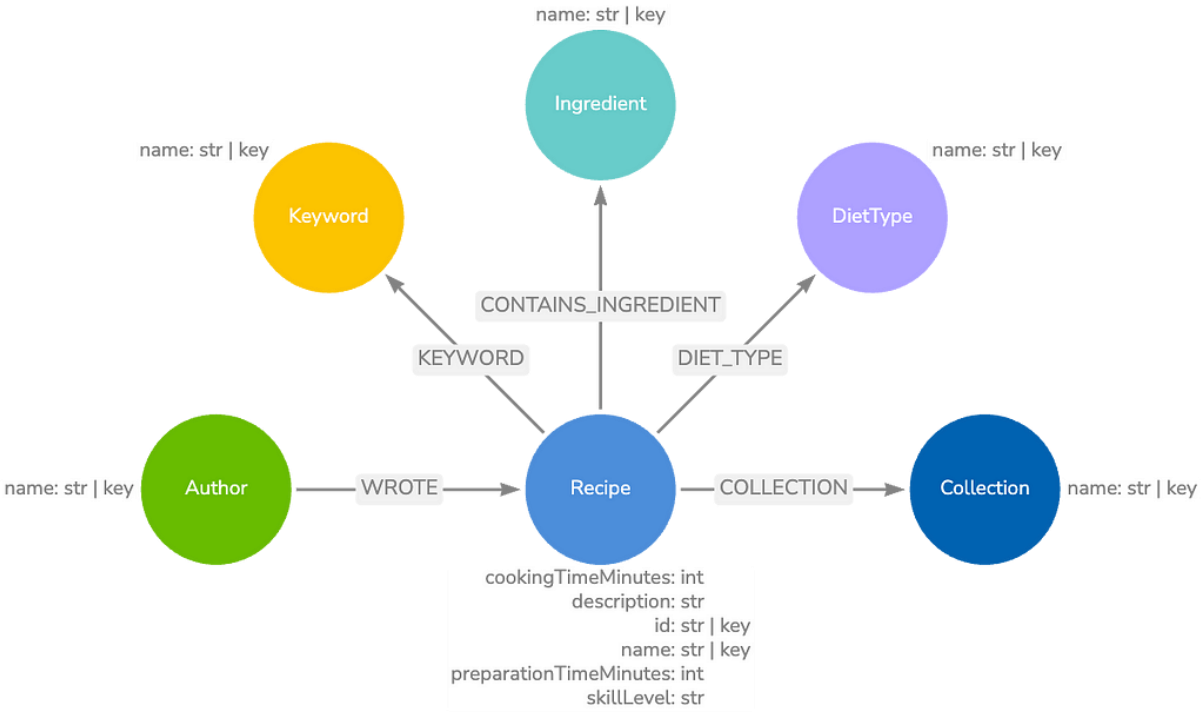
OR Orchestra -tion	RTU Ready To Use	SC Scaling	NC No Code	DP Deployment	LC Low Code	CO Cost Optimization	PC Pro Code	MON Monitoring
LOG Logging	CI Continuous Integration	CD Continuous Delivery						

Interface

UI User Interface	CLI Command Line Interface	APIX API Exposure	INTG Integration Layer	UX User Experience

Evaluation

MET Metrics	LOGS Logs & Traces	OBS Observability	QA Quality Assurance	BEN Bench -marking



Below is a **clean, structured, and interview-ready explanation** of the video content you summarized, with clearer flow and terminology.

🧠 AI Periodic Table — Explained Clearly

The video introduces the **AI Periodic Table** as a **mental model** to understand how modern AI systems are built—similar to how the chemistry periodic table organizes elements.

Instead of chemicals, this table organizes **AI building blocks** into:

- **Rows (Periods)** → system maturity
- **Columns (Families)** → system capabilities

This helps engineers **decode AI architectures, products, and stacks**.

🧩 Structure of the AI Periodic Table

🔹 Rows = Maturity Levels

1. **Primitives** – foundational building blocks
2. **Compositions** – combined capabilities
3. **Deployment** – production & operations
4. **Emerging** – next-generation intelligence

🟦 Row 1: Primitives (Foundations)

These are the **atomic units** of AI systems.

PR – Prompts

- Reactive instructions given to an LLM
- Define behavior, format, and intent
- Example: *"Answer concisely as a data scientist"*

EM – Embeddings

- Numerical vector representations of meaning
- Enable **semantic search & similarity**
- Core to retrieval and memory systems

LG – Large Language Models

- Foundational reasoning engines
- Examples: ChatGPT-like models, enterprise LLMs
- Provide language understanding and generation

📌 *Without Row 1, no AI system exists.*

🟩 **Row 2: Compositions (Power Combinations)**

Where AI becomes **useful and grounded**.

FC – Function Calling

- Allows LLMs to call external tools or APIs
- Enables actions like:
    - Search
    - Database queries
    - Code execution

VX – Vector Databases

- Stores embeddings
- Optimized for semantic similarity search
- Acts as **long-term memory**

RG – Retrieval Augmented Generation (RAG)

- Orchestrates:
    - Embeddings
    - Vector DBs
    - LLMs
- Produces **fact-grounded responses**

GR – Guardrails

- Runtime safety and validation
- Enforces:
    - Output schemas
    - Policy compliance
    - Content safety

MM – Multimodal Models

- Process text, images, audio, video
- Enable richer AI interactions

📌 *Most enterprise GenAI apps live in Row 2.*

🟧 **Row 3: Deployment (Production AI)**

Where AI systems become **autonomous, scalable, and safe**.

AG – Agents

- Use **think → act → observe** loops
- Can:
    - Plan
    - Execute tools
    - Adapt based on results

FT – Fine-tuning

- Trains base models on domain-specific data
- Improves accuracy and tone consistency

FW – Frameworks

- Orchestration layers (e.g., LangChain-like systems)
- Manage:
    - Tool routing
    - State
    - Workflows
    - Retries

RT – Red Teaming

- Adversarial testing
- Identifies:
    - Prompt injection
    - Safety failures
    - Model weaknesses

SM – Small Models

- Distilled, task-specific models
- Faster, cheaper, deployable at the edge

🟥 **Row 4: Emerging (Next Frontier)**

Advanced concepts shaping the future.

MA – Multi-Agent Systems

- Multiple agents collaborate
- Decomposition, delegation, coordination

SY – Synthetic Data

- AI generates training data for AI
- Solves data scarcity & privacy issues

IN – Interpretability

- Explains **why** models behave as they do
- Critical for trust, regulation, and debugging

TH – Thinking Models

- Built-in reasoning time
- Chain-of-thought / planning before answering

⚗️ **"Reactions" — How Elements Combine**

✅ **Example 1: Knowledge-Aware Chatbot**

Uses:

- Prompts (PR)
- Embeddings (EM)
- Vector DB (VX)
- RAG (RG)
- LLM (LG)
- Guardrails (GR)

➡️ Result: **Accurate, grounded enterprise chatbot**

🔁 **Example 2: Agentic Loop**

Uses:

- Agent (AG)
- Function Calling (FC)
- Framework (FW)

Flow:

Goal → Agent decides → Tool call → Observation → Loop

➡️ Result: **Autonomous task-completing AI**

🎯 **Why This Model Is Powerful**

- ✅ Explains **end-to-end AI architecture**
- ✅ Separates hype from real system components

- ✅ Ideal for **system design interviews**
- ✅ Maps directly to **GenAI / LLMOps / Agentic AI**

🧠 **One-Line Summary**

The AI Periodic Table organizes foundational, compositional, deployment, and emerging AI components into a structured framework, enabling systematic design of RAG pipelines, agentic systems, and production-grade AI architectures.

##

🚀 **AI Periodic Table: A Simple Way to Understand Modern AI Systems**

AI systems are becoming increasingly complex — LLMs, RAG, agents, tools, guardrails, multimodal models… it's easy to get lost.

This video introduces a powerful mental model: the **AI Periodic Table** 🧠
Just like the chemistry periodic table, it organizes AI into **foundational elements, compositions, deployment patterns, and emerging capabilities**.

🔹 **Row 1 – Primitives (Foundations)**
• Prompts (PR) – instructions that drive behavior
• Embeddings (EM) – semantic representations
• LLMs (LG) – core reasoning engines

🔹 **Row 2 – Compositions (Where value starts)**
• Function Calling (FC) – tool execution
• Vector Databases (VX) – semantic memory
• RAG (RG) – grounded generation
• Guardrails (GR) – safety & validation
• Multimodal Models (MM)

🔹 **Row 3 – Deployment (Production AI)**
• Agents (AG) – think → act → observe loops
• Fine-tuning (FT) – domain adaptation
• Frameworks (FW) – orchestration (LangChain, etc.)
• Red Teaming (RT) – adversarial testing
• Small Models (SM) – fast & cost-efficient

🔹 **Row 4 – Emerging (Future direction)**
• Multi-Agent Systems (MA)
• Synthetic Data (SY)
• Interpretability (IN)
• Thinking Models (TH)

⚗️ What's powerful is how these elements **combine into "reactions"**:

- A **RAG chatbot** = PR + EM + VX + RG + LG + GR
- An **Agentic system** = AG + FC + FW (looping until the goal is achieved)

👉 This framework makes it much easier to **design, explain, and interview for GenAI systems** — from chatbots to autonomous agents.

🎥 Video: https://www.youtube.com/watch?v=ESBMgZHzfG0

#ArtificialIntelligence #GenerativeAI #LLM #RAG #AgenticAI #LLMOps #SystemDesign #AIArchitecture #DataScience #MachineLearning