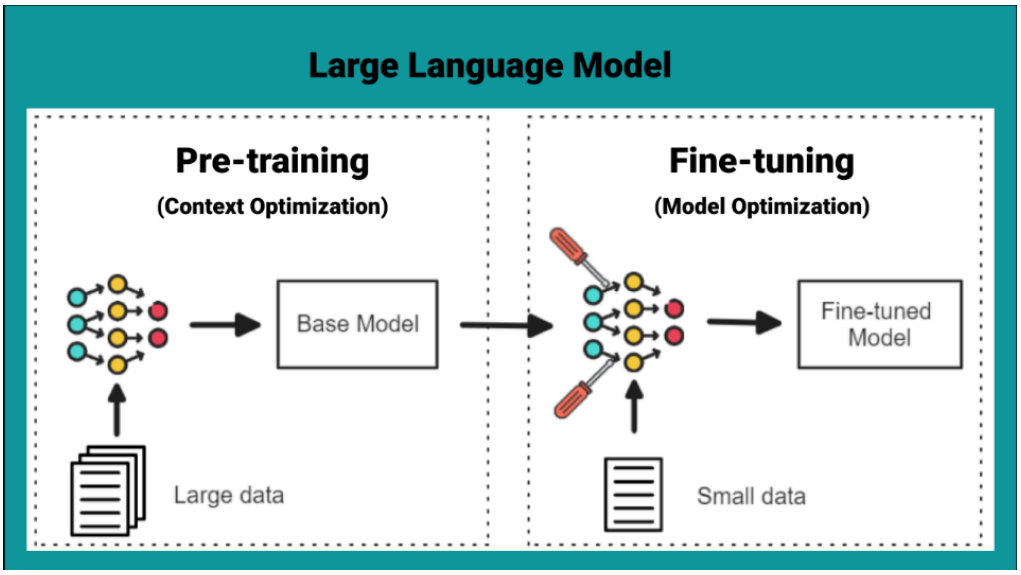
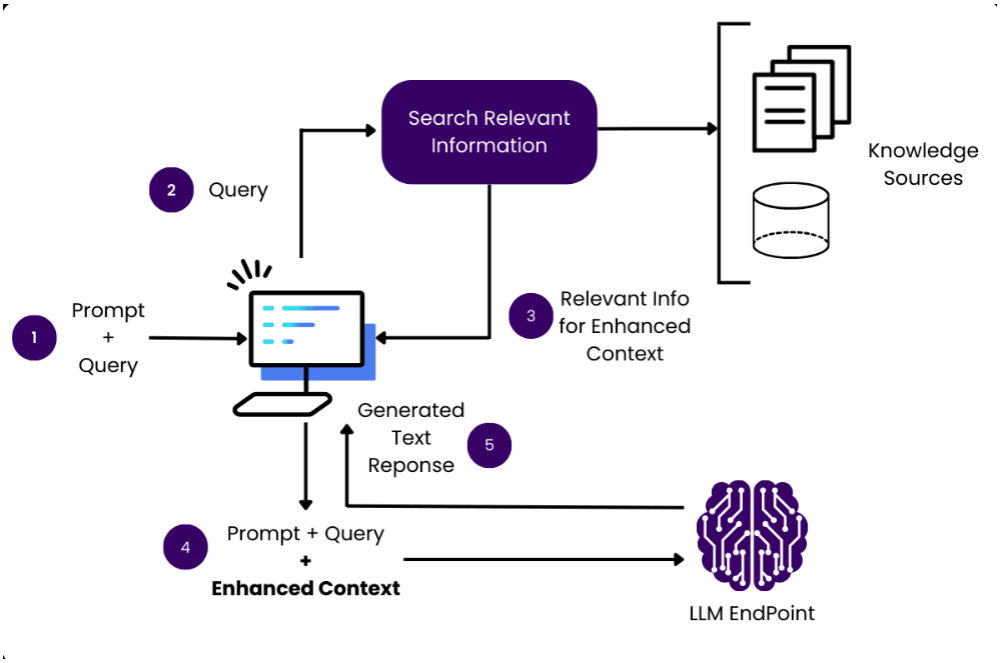
Fine-tuning vs RAG: ইন্ডাস্ট্রি কম্প্যারিজন

১) দুটোর মূল ধারণা (এক লাইনে)

- **Fine-tuning:** মডেলের ওজন/প্যারামিটার আপডেট করে তাকে নতুন আচরণ/স্কিল শেখানো (model পরিবর্তন)। (Microsoft Learn)
- **RAG (Retrieval-Augmented Generation):** মডেল না বদলে, ইনফারেন্সের সময় বাইরের ডকুমেন্ট/নলেজ বেস থেকে তথ্য এনে সেই কনটেক্সটে উত্তর বানানো (knowledge যোগ করা)। (Wikipedia)





২) RAG কীভাবে কাজ করে (Industry pipeline)

RAG সাধারণত ৪ ধাপে বোঝানো হয়: **Indexing → Retrieval → Augmentation → Generation**। (Wikipedia)

1. **Indexing:** ডকুমেন্ট chunk করা + embedding তৈরি + vector store-এ রাখা
2. **Retrieval:** ইউজার প্রশ্নের সাথে similarity করে relevant chunk বের করা
3. **Augmentation:** prompt-এর সাথে retrieved context যোগ করা
4. **Generation:** LLM সেই context দেখে উত্তর তৈরি করে

৩) Fine-tuning কীভাবে কাজ করে (Industry pipeline)

Fine-tuning সাধারণত দুই ধরনের লক্ষ্য নিয়ে হয়:

- **Behaviour/format** শেখানো (যেমন: নির্দিষ্ট টোন, নির্দিষ্ট output schema, tool-usage habit)
- **Task specialization** (যেমন: classification, extraction, domain-style writing)

Microsoft Learn–এ “RAG vs fine-tuning” গাইডে এই পার্থক্যটি স্পষ্টভাবে ব্যাখ্যা করা হয়েছে। (Microsoft Learn)

৪) তুলনা: কোনটা কবে ভালো?

A) যখন **RAG** সাধারণত সেরা পছন্দ

১) তথ্য যদি **private / আপডেটেড / frequently changing** হয়

- কোম্পানির policy, SOP, প্রোডাক্ট প্রাইস, HR নীতি, internal wiki
- এগুলো পরিবর্তনশীল—fine-tuning করলে প্রতিবার আবার train/update লাগতে পারে
- RAG-এ শুধু knowledge base আপডেট করলেই চলে (Wikipedia)

২) **"Citations / source-grounded answer"** দরকার হলে

RAG-এ তুমি retrieved snippet দেখিয়ে "কোথা থেকে বলছি" টাইপ grounding দিতে পারো। এটা enterprise QA, compliance, support-এ খুব গুরুত্বপূর্ণ। (IBM)

৩) দ্রুত **MVP/**পাইলট করতে হলে

RAG তুলনামূলকভাবে দ্রুত দাঁড় করানো যায়—ডকুমেন্ট ইনজেস্ট + রিট্রিভার + prompt। (Microsoft)

RAG-এর বড় সুবিধা **(industry framing):**
"Model retrain ছাড়াই নতুন তথ্য যোগ করা যায়"—এটাই RAG-এর selling point। (Wikipedia)

B) যখন **Fine-tuning** সাধারণত সেরা পছন্দ

১) স্টাইল/টোন/ফরম্যাট "স্থায়ীভাবে" শেখাতে হলে

যেমন:

- সবসময় JSON schema মেনে আউটপুট
- খুব নির্দিষ্ট writing style (legal clause style, brand voice)
- multi-step instruction-following consistent করা

এখানে fine-tuning কাজের, কারণ এটা **model behaviour** বদলায়। (Microsoft Learn)

২) রিট্রিভাল ছাড়াই "দক্ষতা" বাড়াতে হলে

যেমন:

- classification/extraction কাজ খুব high-precision চাই
- latency খুব কম দরকার (retrieval step বাদ দিতে চাই)

৩) যখন কনটেন্ট **“knowledge”** নয়, বরং **“skill”**

উদাহরণ:

- customer support tone policy
- refusal style
- reasoning template
  এইগুলো RAG দিয়ে “তথ্য” হিসেবে ঢোকানো যায়, কিন্তু **consistent habit** বানানো কঠিন; fine-tuning এতে ভালো। (IBM)

৫) ঝুঁকি ও ট্রেডঅফ

RAG-এর **common** সমস্যা

1. **Retrieval miss** হলে উত্তর খারাপ: ভুল chunk এলে hallucination/irrelevance বাড়ে
2. **Chunking/Indexing quality critical**: chunk size, overlap, metadata—সব পারফরম্যান্সে বড় প্রভাব ফেলে
3. **Latency** বাড়ে: retrieval + rerank + generation
4. **Security/Governance**: ডকুমেন্ট leakage, ACL enforcement—enterprise-এ বড় বিষয় (Microsoft)

Fine-tuning-এর **common** সমস্যা

1. ডেটা**/label cost**: ভাল ডেটা বানানো কঠিন
2. **Maintenance**: নতুন policy এলে পুনরায় টিউন/রিটেস্ট
3. **Hallucination** কমবে**—**এটা গ্যারান্টি নয়: knowledge grounding না থাকলে ভুল তথ্য বলতেই পারে
4. **Overfitting/Regressions**: base model-এর general capability নষ্ট হতে পারে (Microsoft Learn)

৬) ইন্ডাস্ট্রিতে সবচেয়ে সাধারণ বাস্তবতা: **Hybrid (RAG + Fine-tuning)**

বেশিরভাগ প্রোডাকশন সিস্টেম একটাকে বাদ দিয়ে আরেকটা নয়—বরং:

- **Fine-tuning** দিয়ে: tone/format/tool-use habit, domain writing style

- **RAG** দিয়ে: fresh/private factual knowledge grounding

এটাই “বিহেভিয়ার + নলেজ” আলাদা করে কন্ট্রোল করার সবচেয়ে বাস্তবসম্মত পথ—Microsoft Learn এই ডিজাইন সিদ্ধান্ত নিয়ে সরাসরি আলোচনা করে। (Microsoft Learn)

`User query → Retriever → Context → (Tuned) LLM → Answer + Citations`