Top 20 high-impact RAG & Vector DB interview questions / knowledge points

Here are **20 high-impact RAG & Vector DB interview questions / knowledge points** you can use in class. Each item has a concise **Ask** and **What strong answers include** so students know the bar.

---

1. **Why RAG instead of fine-tuning?**
   Ask: When do you pick RAG vs. (re)training/fine-tuning?
   Strong answers: Need fresh/private data; lower cost & faster iteration; controllable provenance/citations; fine-tune for style/control when knowledge is stable or to compress prompts; often **RAG + light tuning** wins.
2. **Chunking strategy trade-offs**
   Ask: How do you chunk docs and why?
   Strong answers: Structure-aware (headings/sections), sliding windows for continuity, semantic chunking for cohesion; balance chunk length vs. recall; maintain **stable IDs + checksums** for idempotent updates.
3. **Hybrid retrieval vs dense-only**
   Ask: Why combine BM25 + embeddings?
   Strong answers: Hybrid improves recall/robustness on out-of-domain or rare terms; use **RRF fusion**; apply filters/ACLs before fusion; optionally **re-rank** top-N with a cross-encoder.
4. **Embedding model selection & normalization**
   Ask: How do you choose embeddings and handle vector math?
   Strong answers: Dimension vs. latency/cost; multilingual vs. domain-specific; **L2-normalize** for cosine; monitor drift; version embeddings; batch + cache requests.
5. **ANN index choices & tuning**
   Ask: IVF/HNSW/ScaNN/FAISS/pgvector—when and why?
   Strong answers: HNSW for high-recall/low-latency; IVF+PQ for large corpora (memory savings); pgvector for Postgres integration; tune ef/search_k, nprobe, M/efConstruction; warm caches.
6. **Metadata filtering & multi-tenancy**
   Ask: How to enforce tenant/ACL filters at query time?
   Strong answers: Pre-filtering (WHERE tenant_id, doc_type), post-filter as fallback; row-level security or scoped queries; never fetch unpermitted chunks to the LLM; audit logs.
7. **Freshness & invalidation**
   Ask: How do you keep RAG up-to-date?
   Strong answers: Incremental **upserts** keyed by (doc_id, ord); checksums to detect change; background re-embedding jobs; TTLs for volatile sources; invalidation hooks on source updates.

8. **Prompt construction & grounding**
   Ask: How do you stop the model from making stuff up?
   Strong answers: Strict instruction template; **grounding with citations**; include **verbatim snippets** not whole docs; refusal policy when confidence low; JSON schema for answers.
9. **Evaluation of RAG**
   Ask: How do you measure quality reliably?
   Strong answers: Golden sets with references; **success@k**, exact-match/fuzzy-match, **faithfulness** (LLM-judge with spot human review), answer similarity; track per-topic/per-tenant; regression dashboards.
10. **Latency & cost optimization**
    Ask: What are your go-to levers?
    Strong answers: Cache normalized queries; **hybrid first, rerank on small N**; compress prompts; stream outputs; parallel tool calls; smaller router model; early-exit if high confidence; pre-compute embeddings.
11. **Caching layers**
    Ask: What do you cache and keyed by what?
    Strong answers: Retrieval results keyed by (tenant, normalized_query, filters, top_k); LLM responses with prompt hash + context hash; eviction/TTL strategy; warmup popular queries.
12. **Re-ranking models**
    Ask: When does a cross-encoder help and what's the cost?
    Strong answers: Boost precision@k on ambiguous queries; run only on top-N (e.g., 50→10); watch latency; can be replaced by smaller rerankers or Distil-CE for throughput.
13. **Long-context LLMs vs RAG**
    Ask: Would you ever skip retrieval with a 1M-token model?
    Strong answers: RAG still wins for **relevance, cost, citations, ACLs**; long context helps but needs good selection; hybrid: use retriever to curate the long context.
14. **Safety & prompt-injection defenses**
    Ask: How do you protect tools/data?
    Strong answers: Tool **allow-lists**, input sanitation, strip instructions from retrieved content, content scanning, timeouts, **deterministic validators** for tool outputs, isolate secrets.
15. **Idempotent ingestion pipeline**
    Ask: How do you prevent dupes and ensure repeatable loads?
    Strong answers: Deterministic chunker; checksums; upsert by (doc_id, ord); **exact versioning** of embeddings/model; dead-letter queue; retries with backoff; metrics for throughput/errors.
16. **Index maintenance & compaction**
    Ask: What ongoing ops does a vector index need?
    Strong answers: Periodic rebuild/merge/compaction; background re-embedding; distribution re-balance; evaluate recall after parameter changes; snapshot/backup strategy.
17. **Structured retrieval / fielded search**
    Ask: How do you combine vectors with structured filters/sorts?

Strong answers: Two-stage: candidate set by vector, filter/rank by **metadata (SQL)**; or hybrid SQL+vector in pgvector; beware filtering after vector score (leaks irrelevant docs).

18. **Handling numeric/code snippets**
    Ask: Why do some queries fail with embeddings?
    Strong answers: Embeddings struggle with exact numbers/code; use **BM25/keyword** expansion; keep **exact-match index**; consider task-specific encoders.
19. **Observability & tracing**
    Ask: What do you trace and alert on?
    Strong answers: Spans for embed→retrieve→rerank→generate; counters for P50/P95, cache hit rate, tool errors, token usage; error taxonomy; redlines/alerts on drift/latency spikes.
20. **Scaling to 100M+ chunks**
    Ask: What breaks and how do you design for it?
    Strong answers: Shard by tenant or doc family; IVF-PQ/HNSW + memory planning; async pipelines; cold/warm tiers; **index per tenant** vs mega-index trade-offs; pre-filter to shrink candidate set; cost modeling.