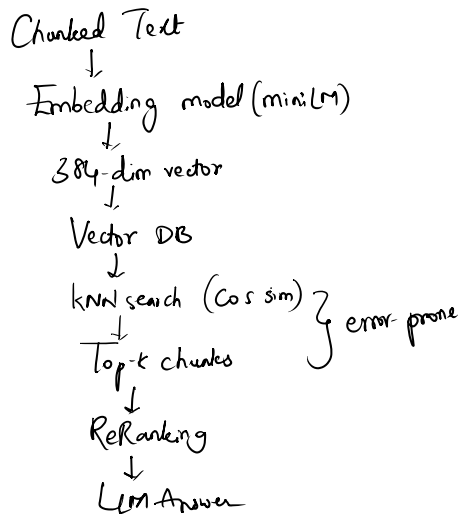


- 1) Embeddings
- 2) Vector db
- 3) kNN
- 4) Semantic similarity
- 5) Retrieval logic



\* Misconception 1:  
(facts)

→ Semantic Similarities

\* Mis-2:

↑ sim = correct answer X  
= related meaning

Mis 3: Vector DB search is exact X  
is approximate ✓  
ANN indices (like HNSW)  
(Top-k)\*\*\*

Mis 4: kNN is just a tool, where you decide:  
i) k value (optimal)  
ii) sim threshold  
iii) filtering logic  
iv) fallback behaviour

\* Improving Search Accuracy:

① Improve Acc by choosing the Better K.

optimal K {  
→ Direct factual — k=1-2  
→ Descriptive — k=3-5  
→ Broad summary — k=5-8  
k=10+

② Sim Threshold:

if sim < 0.55:  
ignore chunk

③ Query Rewriting:

activate different semantic  
regions of an LLM.

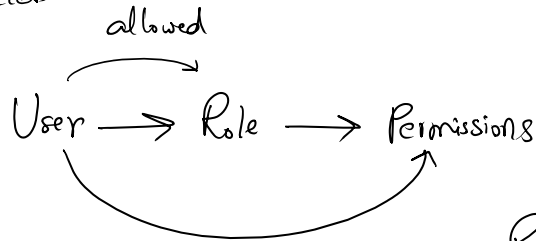
④ Impact on Retrieval chunks: (Debugging skill)

⑤ kNN + Metadata filtering:

→ semantic similarity + {  
dep filtering  
role filter (Rsrc)  
document filter  
type  
} → Improve Precision.

RBAC makes the project real.

Role Based Access  
Control



Eg: Employee

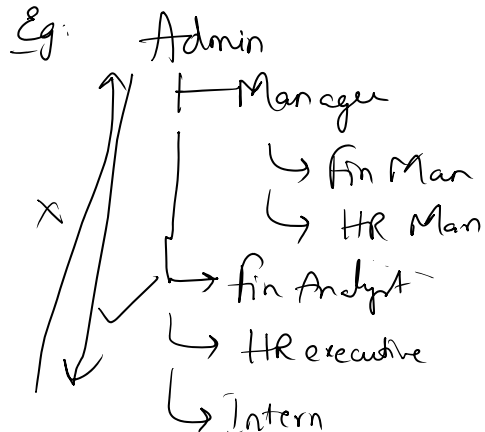
Role: Finance Analyst X

Permissions:

- read\_finance\_docs
- read\_reports

Role hierarchy

Eg:



→ Hierarchy gives us clean permission inheritance.

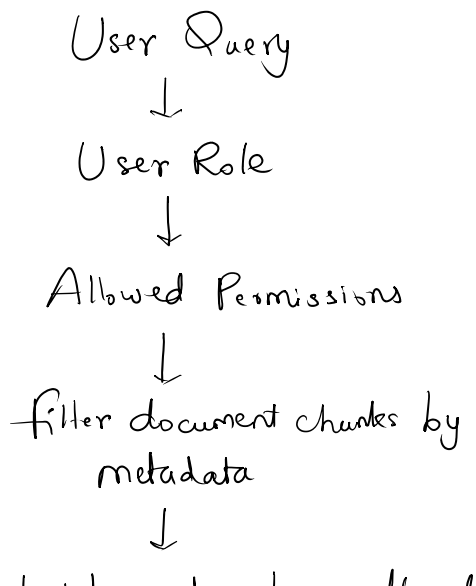
Level

- read — can view content
- write — can create/update
- delete — can remove/delete
- admin — full control

\* Role-Permissions  
json structure:

```
{
  "roles": {
    "admin": {
      "inherits": [ ],
      "permissions": [ "*" ]
    },
    "manager": {
      "inherits": [ ],
      "permissions": [
        "read:finance",
        "read:hr",
        "read:reports"
      ]
    }
  }
}
```

Pipeline:



↓  
kNN search only on allowed  
chunks (top k)  
↓  
LLM Answer

'read: hr',  
"read: reports"  
3,  
fn  
hr-  
int-